

YAKUNIY DAVLAT ATTESTATSIYASI

S1. Operatsion tizim faniga kirish, uning maqsadi va vazifalari.

Operatsion tizim faniga kirish

Operatsion tizim (OS) - bu dasturiy ta'minot bo'lib, kompyuter yoki boshqa raqamli qurilmalarni boshqarish, undan foydalanish va resurslarni boshqarish uchun xizmat qiladi. Ushbu fan operatsion tizimlarning tuzilishi, funksiyalari, turlari va ulardan foydalanish usullari haqida o'rganamiz.

Operatsion tizimning maqsadi va vazifalari

Operatsion tizimning asosiy maqsadi - kompyuter yoki raqamli qurilmani samarali va xavfsiz ishlashini ta'minlash. Buning uchun u quyidagi vazifalarni bajaradi:

1. Resurslarni boshqarish: Protsessor, xotira, disk, tashqi qurilmalar kabi resurslardan samarali foydalanishni ta'minlash.
2. Foydalanuvchi interfeysi: Foydalanuvchilarga qurilmadan foydalanish uchun qulay foydalanuvchi interfeysi taqdim etish.
3. Xavfsizlik va himoya: Tizimni noqonuniy kirish va buzilishlardan himoya qilish.
4. Ilovalarni boshqarish: Turli dasturlarni boshqarish, ularning ishini muvofiqlashtirish va samarali ishlatish.
5. Fayllar va kataloglarni boshqarish: Fayl tizimini boshqarish, fayllar va kataloglarni odam tushuna oladigan nomenklatura bilan boshqarish.
6. Kengaytirish imkoniyatlari: Keyinchalik turli yangi qurilmalar va dasturiy ta'minotni qo'shish imkoniyatini ta'minlash.

Operatsion tizim ushbu vazifalarni bajarish orqali kompyuter yoki qurilmadan samarali foydalanish va ishlashini ta'minlaydi.

S2. Operatsion tizim klassifikatsiyasi.

Operatsion tizim klassifikatsiyasi

Operatsion tizimlar quyidagi asosiy turlardan iborat:

1.Birjoylik (Single-tasking) OS - Faqat bitta dasturni bajarish mumkin. Masalan, CP/M, MS-DOS.

2.Ko'p vazifalik (Multi-tasking) OS: Bir vaqtning o'zida bir nechta dasturlarni bajarish mumkin. Masalan, Windows, Linux, macOS.

3.Ko'p foydalanuvchili (Multi-user) OS - Bir vaqtning o'zida bir nechta foydalanuvchilar tizimdan foydalanishi mumkin. Masalan, UNIX, Linux, Windows Server.

4.Yopiq (Embedded) OS - Maxsus qurilmalarda o'rnatiladi, masalan, mobil telefonlar, o'yin konsollari, avtomobillar. Masalan, Android, iOS, Windows CE.

5.Real vaqt (Real-time) OS - Tezkor javob qaytarish talab qilinadigan tizimlar uchun mo'ljallangan. Masalan, Linux RT, FreeRTOS, QNX.

6.Tarmoq (Network) OS - Turli kompyuterlar va qurilmalarni birlashtirib, yagona tarmoqni boshqarish uchun mo'ljallangan. Masalan, Windows Server, Linux, macOS Server.

Bu klassifikatsiya asosiy turdagi operatsion tizimlarni ajratib ko'rsatadi, ammo amaliyotda ular o'rtasida uyg'unlashma ham bo'lishi mumkin.

Operatsion tizimlar turli xil usullarda klassifikatsiya qilinishi mumkin, ammo ular odatda quyidagi asosiy kategoriyalarga bo'linadi:

1. Tizimning qurilishi bo'yicha:

- **Monolitik tizimlar:** Bu tizimlar, barcha tizim funksiyalarini bitta integratsiya bo'lib to'plashgan holatda ishlaydi. Bularning yozilishi odatda qiyinroq, ammo ularga tezkor murojaat mumkin.
- **Muduli tizimlar:** Bu tizimlarda, tizim funksiyalari modullarga bo'linadi. Har bir modulning o'z funksiyasi bor, va ularga murojaat qilish ason.

2. Ishlovchi tizim bo'yicha:

- **Kompyuter uchun tizimlar:** Masalan, Windows, macOS, Linux.
- **Mobil qurilmalar uchun tizimlar:** Masalan, iOS, Android.
- **Embadded tizimlar:** Kichik, umumiy ravishda boshqa qurilmalar uchun maxsus tizimlar, masalan, IoT qurilmalari yoki avtomobillar.

3. Interfeys bo'yicha:

- **Komand line tizimlar:** Foydalanuvchilar bilan interaktsiyani yozma shaklida qilish imkonini ta'minlaydi. Masalan, DOS, Unix shell.
 - **Grafikali tizimlar:** Foydalanuvchilar interfeys yordamida grafik interfeysi orqali boshqarishadi. Masalan, Windows, macOS.
4. **Tarmoq tizimlari bo'yicha:**
- **P2P tizimlar:** Bir xil martalik qurilmalar o'rtasida aloqalarni tashkil etish uchun ishlatiladi.
 - **Klient-server tizimlar:** Boshqaruv markazi bo'lgan tarmoq tizimlari.
5. **Xavfsizlik bo'yicha:**
- **Xavfsizlikni boshqarishda yuqori tizimlar:** Masalan, tarmoq uskunalari uchun maxsus tizimlar yoki banklar uchun maxsus tizimlar.
 - **Oddiy tizimlar:** Odatiy ma'lumotlar uchun ishlatiladi.
6. **Qurilish turi bo'yicha:**
- **Xususiy, o'zi-ta'minlangan tizimlar:** Bitta xususiy qurilmani yoki ma'lum bir kompyuterni ishlatish uchun yaratilgan tizimlar.
 - **Umumiy tarmoq tizimlari:** Boshqaruv markazi bo'lmagan va ko'pgina qurilmalar uchun ishlatiladigan tizimlar.

Bu klassifikatsiya, operatsion tizimlarning turli xil xususiyatlari asosida aniqlanadi va ularni ajratib turishga yordam beradi.

S3. BIOS tizimi.

BIOS, "Basic Input/Output System"ning qisqartmasi hisoblanadi. Bu, kompyuter qurilmalari uchun juda muhim tizim asosiy dasturi hisoblanadi. BIOS, kompyuter ishlab chiqarilganida o'rnatiladi va uning boshlash tizimi tugagandan so'ng avtomatik ravishda ishga tushadi.

BIOS, quyidagi vazifalarni bajaradi:

1. **Qurilmani boshlash:** BIOS, kompyuter yoki boshqa qurilmani boshlash jarayonida asosiy rol o'ynaydi. Qurilma yoqtirilgandan so'ng, BIOS qurilma qismlarini (masalan, protsessor, xotira, klaviatura, ekran va boshqalar) tekshiradi va ularni dasturga tayyorlash jarayonini boshlaydi.
2. **Ishlovchi tizim ma'lumotlarni saqlash:** BIOS, operatsion tizim o'rnatilgandan oldin kompyuter bilan bog'liq ma'lumotlarni saqlash uchun xotirani ishlatadi. Bu, tarmoq kartalari, disk sifatida, protsessor tomonidan o'qilgan ma'lumotlar, qurilma o'rnatilgan jihatlari va boshqalar bo'lishi mumkin.
3. **Bootning protsessini boshlash:** BIOS, kompyuter yoki boshqa qurilmani yoqib, operatsion tizimni o'rnatishni boshlash uchun qurilma yoki bootable dastur yoki qurilma yuklash vositasi bo'lishi mumkin bo'lgan joylarni izlaydi. BIOS, qurilma yoki operatsion tizimni yuklash maqsadida kerakli ma'lumotlarni topib oladi va uni ishga tushiradi.
4. **Qurilma sozlamalari:** BIOS, foydalanuvchi tomonidan qurilma sozlamalari va konfiguratsiyasi o'zgartirish imkonini beradi. Bu, qurilma tomonidan ishlatiladigan vositalar, o'zgaruvchilar, protsessor sozlamalari, disk o'chirishlar va boshqalar to'g'risidagi sozlamalar uchun mumkin.
5. **Diagnostika va xato to'g'rilash:** BIOS, kompyuter yoki qurilmaning ishlashida yuzaga kelgan xatoliklarni aniqlaydi va qanday xatoni tuzatish kerakligini aniqlash uchun imkoniyatlar ta'minlaydi.
6. **Foydalanuvchi interfeysi:** BIOS, foydalanuvchi interfeysini taqdim etadi, qo'llanuvchining sozlamalarni o'zgartirish va qurilmani boshqarish imkonini beradi.

BIOS asosiy vazifalari:

1. **Dastlabki o'z-o'zini tekshirish (POST):** Kompyuter yoqilganda, BIOS dastlabki o'z-o'zini tekshirish jarayonini amalga oshiradi. Bu jarayonda BIOS qurilmalarning ishlashini va xotiraning to'g'ri yuklashini tekshiradi.
2. **Qurilmalarni initiallashtirish:** POST jarayonidan so'ng, BIOS qurilmalarni, masalan, CPU, xotira, disk, klaviatura, monitor va boshqalarni initiallashtiradi.
3. **Tizimni yuklab olish:** BIOS operatsion tizimni yuklab olish uchun mas'ul. U MBR (Master Boot Record) yoki UEFI (Unified Extensible Firmware Interface) kabi tizimlardan operatsion tizimni yuklaydi.

4. **BIOS sozlamalari:** Foydalanuvchilarga BIOS sozlamalarini o'zgartirishga imkon beradi. Bu sozlamalar tizimning asosiy parametrlarini, masalan, qurilmalar, boot ketma-ketligi, xavfsizlik va boshqalarni o'z ichiga oladi.
5. **Qo'shimcha funksiyalar:** Ayrim BIOS talablarga mos keladigan qo'shimcha funksiyalarni taqdim etishi mumkin, masalan, diagnostika vositalari, xavfsizlik yechimlar va hokazo.

BIOS kompyuterning dastlabki ishga tushirish va qurilmalarni boshqarish uchun kritik ahamiyatga ega. U operatsion tizim yuklanadigan vaqtgacha kompyuterning asosiy funksiyalarini ta'minlaydi.

S4. Operatsion tizim tarkibiy qismlari:

Operatsion tizim quyidagi asosiy tarkibiy qismlardan iborat:

1. **Kernel (Yodro)** - Operatsion tizimning eng asosiy va muhim qismi. Kompyuter resurslarini boshqarish, jarayonlarni boshqarish, xotiradan foydalanish va qurilmalarni boshqarish kabi asosiy vazifalarni bajaradi. Masalan, Linux kerneli, Windows NT kerneli.
2. **Foydalanuvchi interfeysi (User Interface - UI)** - Foydalanuvchilarga kompyuter bilan interaktiv holda ishlash uchun grafik yoki matn asosidagi interfeysni taqdim etadi. Masalan, Windows Explorer, GNOME, KDE, macOS Finder.
3. **Fayl tizimi (File System)** - Fayllar va kataloglarni saqlash, boshqarish va ularga kirish imkonini beradi. Masalan, FAT32, NTFS, ext4, HFS+.
4. **Qurilmalar drayverlari (Device Drivers)** - Turli qurilmalar (printer, skaner, videokarta va h.k.) bilan o'zaro muloqot qilish uchun zarur bo'lgan dasturiy ta'minotdir. Operatsion tizimning ushbu qurilmalar bilan to'g'ridan-to'g'ri muloqot qilishiga imkon beradi.
5. **Xizmatlar (Services)** - Asosiy tizim funksiyalarini yoki ilovalar tomonidan foydalaniladigan funksiyalarni ta'minlaydi. Masalan, tarmoq xizmatlari, xavfsizlik xizmatlari, print xizmatlari.
6. **Utility dasturlar** - Foydalanuvchilarga qurilmani boshqarish, sozlash, diagnostika qilish va boshqa amallarni amalga oshirish uchun yordam beradi. Masalan, Command Prompt, Control Panel, Task Manager. Ushbu tarkibiy qismlar bir-biri bilan uzviy bog'liq holda operatsion tizimning muvaffaqiyatli ishlashini ta'minlaydi.

S5. Interfeys va uning turlari.

Interfeys va uning turlari

Interfeys - bu ikkita tizim yoki qurilma o'rtasidagi o'zaro aloqa va muloqot uchun mo'ljallangan vositadir. Operatsion tizimlar va ulardagi ilovalar uchun quyidagi asosiy interfeys turlari mavjud:

1. **Grafik foydalanuvchi interfeysi (Graphical User Interface - GUI)** - Foydalanuvchilarga grafikalar, ikonalar, oynalar, menyular, panellar va boshqa vizual elementlardan foydalanish imkonini beradi. Masalan, Windows, macOS, GNOME, KDE.
2. **Matn asosidagi foydalanuvchi interfeysi (Command Line Interface - CLI)** - Foydalanuvchilar tomonidan text-base buyruqlar kiritish orqali tizim bilan o'zaro munosabatda bo'lish imkonini beradi. Masalan, Windows Command Prompt, Linux/Unix Bash, macOS Terminal.
3. **Veb interfeysi (Web Interface)** - Veb-brauzerlar orqali tizim bilan o'zaro munosabatda bo'lishga imkon beradi. Masalan, veb-sahifalar, veb-ilovalar, veb-xizmatlar.
4. **Mobil interfeysi (Mobile Interface)** - Mos ravishda mobil qurilmalar uchun mo'ljallangan interfeys. Masalan, Android, iOS, Windows Phone interfeyslari.
5. **Ovoz asosidagi interfeys (Voice Interface)** - Foydalanuvchilar ovoz buyruqlari orqali tizim bilan ishlashlariga imkon beradi. Masalan, Siri, Alexa, Google Assistant.
6. **Kataklashgan interfeys (Augmented Reality Interface)** - Fizik olamni virtual ma'lumotlar bilan boyitish orqali foydalanuvchilarga yangi interaktiv tajriba taqdim etadi. Masalan, Google Glass, Microsoft HoloLens.

Har bir interfeys turi, foydalanuvchilar va kompyuter tizimlari o'rtasidagi muloqotni osonlashtirish uchun maqsadga muvofiq rivojlangan va ishlab chiqilgan. Har bir turi o'zining afzalliklari va cheklovlari mavjud bo'lganligi uchun, foydalanuvchi tomonidan talab qilingan interfeys turi amalga oshiriladi.

S6. Operatsion tizimlarni qurish asoslari.

Operatsion tizimni qurish asoslari quyidagi asosiy jarayonlarga asoslanadi:

1. **Qo'llanuvchi talablari va maqsadlarini aniqlash:** Bir operatsion tizimni qurishdan oldin, qo'llanuvchilar tomonidan talab qilingan funksiyalarni va maqsadlarni aniqlash juda muhimdir. Bu, operatsion tizimning qanday tarmoqlarda, qanday qurilmalarda, qanday turdagi dasturlar uchun qulayliklar, xavfsizlik talablari va boshqalar bo'lishi kerakligini aniqlashga yordam beradi.
2. **Platformani tanlash:** Operatsion tizimni qurishda, qo'llaniladigan qurilmaning platformasi (masalan, kompyuter, telefon, IoT qurilmalari)ni tanlash juda muhimdir. Bu, qurilmaning xususiyatlari, qurilma ustida ishlovchi dasturlarning mavjudligi va boshqalar kabi omillar ni hisobga oladi.
3. **Yadro va tizim arxitekturani tanlash:** Operatsion tizimning yadro va arxitekturasi, uni boshqarishning asosiy usuli va funksiyalari bilan mos keladi. Klassik monolitik yadro, mikroservisli yadro, va distributiv yadro kabi turli yadro arxitekturalar mavjud. Tizim arxitekturasi, operatsion tizimning faoliyatini va ishlaydigan qurilma turlarini belgilaydi.
4. **Xavfsizlikni ta'minlash:** Operatsion tizimni qurishda xavfsizlikning katta e'tibori berilishi zarur. Bu, foydalanuvchilar ma'lumotlarini himoya qilish, tarmoq xavfsizligini ta'minlash, avtomatik yangilanishlar orqali xavfsizlik yopiq kodini ta'minlash kabi ko'plab jarayonlarni o'z ichiga oladi.
5. **Tarmoq bilan aloqani tuzish:** Bugungi kunda operatsion tizimlar, tarmoq bilan yuqori darajada integratsiya qilingan. Shuning uchun, operatsion tizimni qurishda tarmoq protokollari (masalan, TCP/IP), tarmoq uskunalari (WiFi, Ethernet), VPN, ishlab chiqarish, tarmoq xavfsizlik sozlamalari kabi elementlar xususiy e'tibor talab etadi.
6. **Dasturlash va tizim bo'yicha xizmatlar:** Operatsion tizimni qurishda, dasturlash va tizim bo'yicha xizmatlarni ta'minlash juda muhimdir. Bu xizmatlar, dasturlash interfeyslari, tarmoq boshqarish, fayl tizimlari, yuklash va qurish tizimlari, xizmat ishga tushirish va boshqalar kabi asosiy funksiyalarni o'z ichiga oladi.

Bu asoslar operatsion tizimni qurishda ko'p qatlamli holatda e'tibor berilishi kerak bo'lgan omillar. Har birini kuzatib borish, operatsion tizimning foydalanuvchi talablari va maqsadlari bo'yicha to'liq yoki qaror qabul qilish uchun juda muhimdir.

S7. EHM arxitekturası

EHM arxitekturası avlodlari

Elektron hisoblash mashinalarining (EHM) arxitekturası tarixiy rivojlanish jarayonida quyidagi asosiy avlodlarni o'z ichiga olgan:

1. **Birinchi avlod (1940-lar)** - Bu tarkibiga tezkor xotira qurilmasi ham bor bo'lgan bazaviy "EHM"dir. Vakuum lampali (elektron lampali) EHM larga asoslangan. ENIAC, UNIVAC I kabi mashhur namunalarga ega. Juda katta o'lchamdagi, unchalik ishonchli bo'lmagan, ko'p xato qiluvchi tizimlar.
2. **Ikkinchi avlod (1950-lar)** - Bu 1 avlod mashinasidan tarkibida tashqi xotira qurilmasi borligi bilan faqlanadi. Tranzistorli texnologiyaga o'tish. UNIVAC II, IBM 700/7000 seriyasidagi EHM lar. Birinchi avlodga nisbatan ancha ishonchli, tezkor va arzonroq.
3. **Uchinchi avlod (1960-lar)** - Bu 2 avlod mashinasidan tarkibida axborot almashinuv qurilmasi borligi bilan farqlanadi. Integarl sxemali texnologiyalarga o'tish. IBM System/360, CDC 6600 kabi yirik kompyuterlar. Kichik o'lchamdagi, ishonchli, katta tezlikdagi tizimlar. Operatsion tizimlar, ma'lumotlar bazasi, dasturlash tillari paydo bo'lgan.
4. **To'rtinchi avlod (1970-lar - 1980-lar)** - Bu 3 avlod mashinasidan har biri parallel ravishda ishlay oladigon 2 va undan ortiq protsesorlari borligi bilan farqlanadi. Apple II, IBM PC kabi shaxsiy kompyuterlar paydo bo'lgan. Grafik interfeys, dasturlash muhitlari, tarmoqlar rivojlangan.
5. **Beshinchi avlod (1980-lar - hozirgi vaqt)** - Bu 4 avlod mashinasidan intellektual interfeys borligi bilan farq qiluvchi, universal sun'iy tafakkur mashinasi. Parallel va tarqatilgan arxitekturalar, ulkan integarl sxemalar. Shaxsiy kompyuterlar, umumiy vazifali serverlap, aqlli qurilmalar keng tarqalgan. Operatsion tizimlar, tarmoqlar, bulutli texnologiyalar rivojlangan.

Bu avlodlar EHM texnologiyalarining izchil taraqqiyotini aks ettiradi va bugungi kunda qo'llaniladigan tizimlarning asosini tashkil etadi.

S8. MS Windows arxitekturasini.

Windows arxitekturasini: x64 va x86

Microsoft Windows operatsion tizimi asosan ikki asosiy protsessor arxitekturasini qo'llab-quvvatlaydi:

1. **x86 (32-bit) arxitekturasini:**

- Intel 8086 protsessorlaridan boshlab rivojlangan 32-bitli arxitektura.
- 32-bit registrlar, 32-bit adreslar, 4 GB xotira hajmi bilan cheklangan.
- Birinchi Windows versiyalari 16-bit bo'lgan, keyinchalik 32-bitli x86 versiyalari paydo bo'lgan.
- 32-bit protsessorlar uchun mo'ljallangan Windows versiyalari: Windows 95/98/ME, Windows XP, Windows Vista, Windows 7 (32-bit).

2. **x64 (64-bit) arxitekturasini:**

- 64-bit protsessorlar uchun mo'ljallangan.
- 64-bit registrlar, 64-bit adreslar, 16 EB xotira hajmi.
- x86 arxitekturasidan asosiy farqi - 64-bit instruktsiyalar va kengaytirilgan funktsionallik.
- 64-bit Windows versiyalari: Windows XP x64 Edition, Windows Vista x64, Windows 7 x64, Windows 8/8.1/10/11 (64-bit).
- x86 ilovalarini ham qo'llab-quvvatlaydi (x86 emulatsiyasi orqali).

Windows operatsion tizimining 32-bit va 64-bit versiyalari asosiy farqi - protsessor arxitekturasini va xotira hajmidagi farqdir. 64-bit Windows kengaytirilgan funktsionallikka, quvvatli protsessor va katta xotiraga ega bo'lgan zamonaviy kompyuterlar uchun mo'ljallangan.

Ushbu ikki arxitektura o'rtasidagi asosiy farqlar:

- **Operand o'lchami:** x86 - 32-bit, x64 - 64-bit
- **Xotira joylashuv:** x86 - 4 GB, x64 - 16 EB
- **Registrlar soni:** x86 - 8 ta, x64 - 16 ta
- **Adres tashish qobiliyati:** x86 - 32 bit, x64 - 64 bit

Natijada x64 arxitekturasini x86 ga qaraganda ancha quvvatli, tezkor va katta xotira hajmiga ega.

S9. MSDOS operatsion tizimi strukturasi

MS-DOS (talaffuzi: em-es-dos; MicroSoft Disk Operating System "Microsoft Disk Ishlatuv Tizimi") x86 arxitekturali shaxsiy kompyuterlar uchun ishlab chiqarilgan ishlatuv tizimidir. 1981-yili IBM buyurtmasiga binoan Microsoft tomonidan Q-DOS asosida tayyorlangan. MS-DOSning asosiy vazifalari:

Windows ishlab chiqilishidan oldin ko'pchilik kompyuterlar MS-DOS (Microsoft Disk Operating System) nomi bilan mashhur bo'lgan Microsoft-dan boshqa operatsion tizimni boshqargan.

MS-DOS - grafik bo'lmagan operatsion tizim. Bu shuni anglatadiki, kompyuterga berilgan barcha buyruqlar klaviaturadan kiritilishi yoki fayldan kiritilishi kerak. Amaliy asboblardan ma'lumotlarni olish yoki boshqarish uchun ishlatiladigan ko'plab kompyuter dasturlari hanuzgacha MS-DOS ostida ishlaydi (ko'plab kompyuter o'yinlari kabi). Shuning uchun, Amaliy muhitida kompyuterlardan foydalanishni rejalashtirgan har bir kishi uchun (yoki har qanday jiddiy kompyuter o'yinlari o'yinchisi uchun) MS-DOS bo'yicha asosiy bilimlar talab qilinadi.

MS DOS OS si quyidagi qismlardan tashkil topagan:

BIOS (Basic Input-Output System), (Kiritish-chiqarishning asosiy tizimsi). Bu kompyuterning doimiy xotirasida joylashgan. Uning vazifasi axborotlarni kiritish va chiqarish bilan bog'liq bo'lgan yeng sodda va universal xizmatlarni bajarishdan iborat. Operatsion tizimning yo'qlovchisi. Bu tizim juda kiska dastur bo'lib, u MS DOSli disketaning birinchi sektorida joylashadi va uning vazifasi MS DOSning kolgan ikki modulini ukishdan iborat

io.sys va msdos.sys tizimli fayllar. Ularni operatsion tizim yo'qlovchisi xotiraga kuchiradi va kompyuter xotirasida doimo saqlaydi. io.sys (INPUT-OUTPUT) BIOSning xotiradagi davomi hisoblanadi. msdos.sys DOSning yuqori darajadagi vazifalarini bajaradi

Disketalarda ma'lumotlar fayllar ko'rinishida saqlanadi. Fayl -bu disketaning ma'lum nom bilan ataluvchi sohasidir. Faylda dastur va hujjat matnlari, bajarilishi mumkin bo'lgan dasturlar va xokazolar saqlanadi. Fayllar 2 xil bo'ladi: matnli va ikkili tizimdagi. Matnli fayllar foydalanuvchi ukishi uchun mo'ljallangan Fayllar va kataloglar.

Har bir fayl ikki qismdan iborat bo'lgan nomga yega. Fayl nomining birinchi qismi asosiy qismi hisoblanadi va 1 tadan 8 tagacha belgidan iborat bo'ladi. Fayl nomining ikkinchi qismi kengaytgich deyiladi va 1 tadan 3 tagacha belgidan iborat bo'ladi. Faylga nom berishda fakat lotin harflaridan foydalanish Maqsadga muvofikdir. Fayl nomida kengaytgich bo'lishi shart yemas. Lekin kengaytgich koidaga ko'ra, faylning qaysi turda yekanligini aks yettiradi va shu bilan foydalanuvchiga qulaylik yaratadi. Masalan:

.com va .exe ishlatishga tayyor buyrukli va bajariladigan fayllar;
.bat - foydalanuvchi yaratgan buyruk fayllar;

.bak - faylning uzgarishdan avvalgi nusxasi;

.txt - matnli fayl;

.doc – matnli, hujjatli fayl;

.bas - Beysik tilida tuzilgan dastur fayli;

.jpeg va bmp, gif va xakozo- grafik va tasvirli fayllar;

MS DOS ning komandalarni qabul qilishga tayyorligini ko'rsatadi. Ba'zan MS DOS ning taqlifi joriy disk va katalog haqidagi ma'lumotni uz ichiga oladi.

S10. Real vaqt rejimidagi operatsion tizimlar.

Real vaqt rejimidagi operatsion tizimlar (Real-Time Operating Systems - RTOS) - ma'lum bir muddatda ma'lumotlarni qayta ishlash va javob qaytarish uchun optimallashtirilgan operatsion tizimlardir. Ular quyidagi asosiy xususiyatlarga ega:

1. **Vaqt chegarasi (Time constraints):**
 - Jarayonlarni ma'lum qisqa muddat ichida bajarilishi shart.
 - Kechikishlar xatoliklariga olib kelishi mumkin.
2. **Determinstlik (Determinism):**
 - Ish jarayonining oldinden aytib berilishi va kutilishi mumkin bo'lishi kerak.
 - Uzilishlar, kutilmagan xodisalar yuz bermasligi kerak.
3. **Masshtablanuvchanligi (Scalability):**
 - Kichik yoki yirik tizimlar uchun moslashuvchan bo'lishi kerak.
 - Yangi qurilmalarni qo'shish oson bo'lishi kerak.
4. **Resurslarni samarali boshqarish:**
 - Xotira, protsessor vaqti, energiya sarfi kabi resurslarni oqilona boshqaradi.
 - Real vaqtda natijani qaytara olish uchun samarali bo'lishi kerak.

RTOS-lar odatda quyidagi sohalar uchun qo'llaniladi:

- **Avtomobil sanoati:** Yoqilg'i yoqish, tormoz tizimi, elektr xavfsizligi va boshqalar.
- **Tibbiyot:** Jarrohlik robotlari, xastalarni kuzatish va boshqarish tizimi.
- **Aviatsiya va kosmonavtika:** Samolyot va raketalar boshqaruvi, uchish nazorati.
- **Muhandislik:** Texnologik jarayonlarni boshqarish, robotlar.
- **Telekommunikatsiya:** Aloqa tarmog'i, telefon almashuvi.

RTOS-lar orasida eng mashhurlar: VxWorks, QNX Neutrino, FreeRTOS, Linux PREEMPT_RT.

S11. Operatsion tizimlarda resurs va jarayon tushunchasi

Operatsion tizimlarda (OT) resurs va jarayon tushunchalari OTning asosiy elementlaridan biri bo'lib, ular tizimning samarali ishlashini ta'minlaydi.

Jarayon — bu bajarilayotgan dasturdir. Jarayonlar OTda resurslardan foydalanish va vazifalarni bajarish uchun asosiy birlik hisoblanadi. Har bir jarayon o'zining xotirasi, registrlari va bajarilish holatiga ega.

Jarayonning asosiy atributlari:

- Jarayon identifikatori (PID) : Har bir jarayon o'ziga xos identifikatorga ega.
- Xotira holati : Jarayonning xotirada egallagan joyi.
- Rejalashtirish ma'lumotlari : OT jarayonlarni qanday rejalashtirishi va bajarishini ko'rsatadi.
- Jarayon holati : Jarayon bajarilayotgan, kutayotgan yoki to'xtatilgan holatda bo'lishi mumkin.
- Kontekst : Jarayonning registrlari, hisoblagichlari va boshqa resurslari holati.

Jarayonlar rejalashtiruvchisi (scheduler) jarayonlarni boshqaradi va ularni protsessorga taqsimlaydi.

Resurslar — bu OT tomonidan boshqariladigan va jarayonlar tomonidan foydalaniladigan fizik yoki virtual komponentlar. Resurslar turlari quyidagilardan iborat bo'lishi mumkin:

- Protsessor vaqti : Protsessor quvvati jarayonlarga taqsimlanadi.
- Xotira : Operativ xotira (RAM) jarayonlarga ajratiladi.
- Kirish-chiqish qurilmalari : Qattiq disklar, klaviaturalar, printerlar va boshqalar.
- Fayllar va fayl tizimlari : Ma'lumotlarni saqlash va ularga kirish uchun foydalaniladigan fayllar.

Jarayon va resurslar o'rtasidagi munosabat

Jarayonlar resurslarga kirish huquqiga ega bo'lishlari uchun OT ularni boshqaradi. Bu jarayonlar va resurslar o'rtasidagi muvozanatni ta'minlash, resurslarning samarali taqsimlanishini va tizimning umumiy ishlashini oshiradi.

Resurslarni boshqarish bir nechta asosiy konseptlarni o'z ichiga oladi:

- Rejalashtirish (Scheduling) : Jarayonlar va resurslar o'rtasida protsessor vaqtini taqsimlash.
- Blokirovka va o'zaro bloklanish (Deadlock) : Jarayonlar resurslarga kirishda bir-birini bloklash holatlari va ularni bartaraf etish usullari.
- Sinxronizatsiya (Synchronization) : Jarayonlar o'rtasida resurslarga kirishni tartibga solish.

Operatsion tizimlarda resurs va jarayon tushunchalari o'zaro chambarchas bog'liq bo'lib, OTni samarali boshqarish va tizimning barqaror ishlashini ta'minlash uchun muhim ahamiyatga ega.

S12.Jarayon

“Dastur” va “topshiriq” tushunchalari statik, faol bo’lmagan (неактивный) obyektlarni tavsiflash uchun ishlatiladi. Dastur bajarilishi uchun, operatsion tizim ma’lum hajmli xotira ajratishi, unga kiritish/chiqarish qurilmalarini va fayllarni bog’lashi, ya’ni butun hisoblash tizimi resurslari hisobidan ma’lum qismini ajratish qo’yishi lozim. Ularning soni va konfiguratsiyasi vaqt o’tishi bilan o’zgarishi mumkin. Bynday faol obyektlarni tavsiflash uchun “dastur” va “topshiriq” terminlari o’rniga yangi “JARAYON” terminini ishlatamiz. Demak qisqa qilib aytganda, jarayon– bu biron bir dastur bajarilishidagi faol topshiriqlar (buyruqlar) ketma-ketlig

Jarayonlar operatsion tizim boshqaruvi ostida hisoblanadi. — Bir protsessorli kompyuter tizimida vaqtning har bir momentida faqat bitta jarayon bajarilishi mumkin. — Multidasturli hisoblash tizimlarida bir nechta jarayonni psevdoparallel qayta ishlash, protsessorni bir jarayondan ikkinchi jarayonga o’tkazish yordamida amalga oshiriladi. Bir jarayon bajarilguncha qolganlari o’z navbatini kutadi.

Jarayon xolati - ko’pincha adabiyotlarda soddalashtirish uchun, jarayonni bajarilish vaqtidagi dasturni xarakterlaydigan abstraksiya sifatida olish tavsiya etiladi. Jarayon OT boshqaruvi ostida hisoblanadi. Bunday qabul qilishda hisoblash tizimlarida bajariladigan xamma narsa(faqat foydalanuvchi dasturlarigina emas, balki OTning ham ma’lum ismlari ham) jarayonlar to’plami sifatida tashkil qilingandir. Bir protsessorli kompyuter tizimida vaqtning har bir momentida faqat bitta jarayon bajarilishi mumkin. Multidasturli hisoblash tizimlarida bir nechta jarayonni psevdoparallel qayta ishlash protsessorni bir jarayondan ikkinchisiga o’tkazish yordamida amalga oshiriladi. Bir jarayon bajarilguncha, qolganlari o’z navbatini kutadi.

S13.Uzilishlarni qayta ishlovchi dasturlar

Uzilishlarni qayta ishlovchi dasturlar operatsion tizimlarda muhim rol o'ynaydi. Uzilish (interrupt) — bu tizimda sodir bo'ladigan va diqqatni talab qiladigan kutilmagan hodisa. Uzilishlar turli sabablarga ko'ra sodir bo'lishi mumkin, masalan, apparat qurilmasi signal yuborganida yoki dasturiy xato yuz berganida.

Uzilish turlari

Uzilishlar ikki asosiy turga bo'linadi:

1. **Apparat uzilishlari (Hardware Interrupts):** Bu uzilishlar apparat qurilmalaridan kelib chiqadi, masalan, klaviaturadan tugmani bosish yoki diskdan ma'lumot o'qish tugallanganda.
2. **Dasturiy uzilishlar (Software Interrupts):** Bu uzilishlar dasturiy ta'minot tomonidan hosil qilinadi. Misol uchun, tizim chaqiruvlari (system calls) yoki dasturiy xatolar (exceptions).

Uzilishlarni qayta ishlash jarayoni

1. **Uzilish aniqlanishi:** Protsessor uzilishni aniqlaydi va joriy jarayonni to'xtatadi.
2. **Uzilishni saqlash:** Protsessor joriy jarayonning kontekstini (registrlar va dastur hisoblagichi) saqlaydi.
3. **Uzilishni qayta ishlash:** Protsessor uzilish dasturini (interrupt handler yoki interrupt service routine - ISR) ishga tushiradi.
4. **Kontekstni tiklash:** Uzilish dasturi tugagandan so'ng, protsessor avvalgi jarayonning kontekstini tiklaydi.
5. **Jarayonni davom ettirish:** Protsessor to'xtatilgan jarayonni davom ettiradi.

Uzilishlarni qayta ishlovchi dasturlar

Uzilish dasturlari — bu maxsus dasturlar bo'lib, uzilish sodir bo'lganda avtomatik ravishda ishga tushadi va uzilishni qayta ishlaydi. Ular quyidagi vazifalarni bajaradi:

1. **Uzilish sababini aniqlash:** Qaysi qurilma yoki dastur uzilish yuborganini aniqlaydi.
2. **Zarur choralarni ko'rish:** Masalan, apparat uzilishida kerakli ma'lumotlarni o'qish yoki yozish, dasturiy uzilishda esa xatoni tuzatish.
3. **Kontekstni boshqarish:** Joriy jarayonning kontekstini saqlash va tiklash.

Misollar

1. **Klaviatura uzilishi:** Foydalanuvchi tugmani bosganda, klaviatura qurilmasi uzilish yuboradi. Uzilish dasturi bu uzilishni qayta ishlaydi va bosilgan tugma kodini o'qib, tegishli dasturga yuboradi.
2. **Taymer uzilishi:** Operatsion tizimda vaqtni boshqarish uchun taymer uzilishlaridan foydalaniladi. Masalan, jarayonlarni rejalashtirishda taymer uzilishi jarayonning bajarilish vaqtini belgilaydi va zarur bo'lganda jarayonlarni almashtiradi.
3. **Kirish-chiqish uzilishlari:** Ma'lumotlarni qattiq diskdan o'qish yoki yozish tugallanganda uzilishlar yuzaga keladi. Uzilish dasturi bu ma'lumotlarni qayta ishlashni tashkil qiladi.

Uzilishlarni qayta ishlash jarayonlari OTning samaradorligini oshirish va apparat bilan dasturiy ta'minot o'rtasidagi o'zaro aloqani ta'minlashda muhim ahamiyatga ega.

S14.Semaforalar

Semaforlar operatsion tizimlarda va parallel dasturlashda muhim sinxronizatsiya vositasi bo'lib, ular bir nechta jarayon yoki iplar (threads) o'rtasida resurslarga kirishni tartibga solish va o'zaro bloklanish (deadlock) holatlarining oldini olishda qo'llaniladi.

Semafor Turlari

1. **Ikkilik semafor (Binary Semaphore):** Shuningdek, mutex (mutual exclusion) deb ham ataladi. U faqat ikki holatda bo'lishi mumkin: 0 yoki 1. Bu turdagi semafor bir vaqtning o'zida faqat bitta jarayonga resursga kirish imkonini beradi.
2. **Umumiy semafor (Counting Semaphore):** Bu turdagi semaforlar ixtiyoriy butun son qiymatini qabul qilishi mumkin. Ular bir nechta jarayonning bir vaqtda bir nechta resursga kirishini boshqarish uchun ishlatiladi.

Semafor Operatsiyalari

Semafor ikki asosiy operatsiyaga ega:

- 1 **Poperatsiyasi (proberen yoki wait):** Bu operatsiya semaforning qiymatini 1 ga kamaytiradi. Agar semaforning qiymati noldan katta bo'lsa, jarayon resursga kiradi. Agar semaforning qiymati nol yoki noldan kichik bo'lsa, jarayon kutish holatiga o'tadi.
- 2 **V operatsiyasi (verhogen yoki signal):** Bu operatsiya semaforning qiymatini 1 ga oshiradi. Agar kutayotgan jarayonlar bo'lsa, ulardan birini uyg'otadi va resursga kirishga imkon beradi.

Semaforlarni Qo'llash

Operatsion tizimda semaforalar - resurslar va jarayonlar o'rtasidagi muloqotni tartibga soluvchi maxsus mexanizmlardir. Ular quyidagi asosiy vazifalarni bajaradi:

1. **Resurslardan foydalanish uchun ruxsat berish va bloklab qo'yish** - Bir vaqtning o'zida bir nechta jarayon resursdan foydalanmoqchi bo'ladi. Semaforlar jarayonlarga resursdan foydalanish uchun ruxsat beradi yoki bloklab qo'yadi.
2. **Chegaralangan resurslarni boshqarish** - Resurslar cheklangan bo'lsa, ulardan foydalanishni semaforlar orqali tartibga solish mumkin. Semaforida o'rnatilgan qiymat resurslar soni bilan mos bo'ladi.
3. **Jarayonlar orasidagi sinxronizatsiya** - Bir jarayon boshqasini kutishi yoki ma'lumotni olishini semaforlar bilan amalga oshirish mumkin. Semaforlar orqali jarayonlar o'rtasidagi ma'lumot almashinuvini tartibga solish mumkin.

Misol: Prodyuser-Konsyumer Muammosi

Prodyuser-konsyumer muammosini hal qilish uchun ikki turdagi semaforlardan foydalaniladi: bo'sh joylarni hisoblovchi semafor (bo'sh joylar sonini ko'rsatadi) va to'ldirilgan joylarni hisoblovchi semafor (to'ldirilgan joylar sonini ko'rsatadi).

S15.Operatsion tizimlarda jarayonlarni boshqarish

Jarayon — bu bajarilayotgan dasturdir. Har bir jarayon operatsion tizim tomonidan alohida boshqariladi va u quyidagi elementlarni o‘z ichiga oladi:

- **Jarayon identifikatori (PID):** Har bir jarayon o‘ziga xos identifikatorga ega.
- **Xotira:** Jarayonning xotirada egallagan joyi, ya'ni kod, ma'lumotlar va stek segmentlari.
- **Rejalashtirish ma'lumotlari:** Tizim jarayonlarni qanday rejalashtirishi va bajarishini ko'rsatadi.
- **Jarayon holati:** Jarayon bajarilayotgan, kutayotgan yoki to‘xtatilgan holatda bo‘lishi mumkin.
- **Kontekst:** Jarayonning registrlari, hisoblagichlari va boshqa resurslari holati.

Jarayon holatlari

Jarayonlar quyidagi holatlardan birida bo‘lishi mumkin:

- **Yaratilgan (New):** Jarayon yangi yaratilgan, lekin hali boshlanmagan.
- **Bajarilayotgan (Running):** Jarayon protsessorda bajarilmoqda.
- **Kutayotgan (Waiting):** Jarayon biror voqeani kutmoqda (masalan, kirish-chiqish operatsiyasi tugashini).
- **Tayyor (Ready):** Jarayon bajarilishga tayyor, lekin protsessorda bajarilmayapti.
- **Tugallangan (Terminated):** Jarayon bajarilishi tugagan.

Operatsion tizimlar jarayonlarni boshqarishda quyidagi asosiy mexanizmlardan foydalanadi:

- **Jarayon holati (Process State) -** Jarayon turli holatlarda bo'lishi mumkin: yangi, tayyor, ishlamoqda, uyqu, to'xtatilgan. Operatsion tizim jarayonlarni holatlari orqali boshqaradi.
- **Jarayon yaratish (Process Creation) -** Yangi jarayon yaratish uchun operatsion tizimdan maxsus chaqiruvlar (system calls) ishlatiladi. Yangi jarayon avvalgi jarayon (ota) asosida yaratiladi.
- **Jarayon tugash (Process Termination) -** Jarayon ma'lum bir sababga ko'ra tugashi mumkin: normal yakunlash, xato, o'chirib tashlash. Tugagan jarayon olib tashlangan, uning resurslaridan boshqa jarayonlar foydalanishi mumkin.
- **Jarayon muloqoti (Process Interaction) -** Jarayonlar o'zaro ma'lumot almashishi, sinxronizatsiyani ta'minlashi kerak. Bunda semaforlar, uzilishlar, xabar navbatlari, almashuvchan xotira kabi mexanizmlar ishlatiladi.

- **Jarayon rejalashtirish (Process Scheduling)** - Operatsion tizim jarayonlar orasida protsessor resurslarini taqsimlaydi. Turli rejalashtirish algoritmlaridan (FIFO, Round Robin, Prioritetlar) foydalaniladi.

Jarayonlarni samarali boshqarish operatsion tizimning asosiy vazifasi bo'lib, uni amalga oshirish uchun turli mexanizmlardan foydalaniladi.

Jarayonlarni rejalashtirish (Scheduling)

Jarayonlarni rejalashtirish operatsion tizimning asosiy vazifalaridan biri bo'lib, u protsessor vaqtini jarayonlar o'rtasida taqsimlashni boshqaradi. Rejalashtiruvchi (scheduler) jarayonlarni tanlash va ularga protsessor vaqtini ajratish uchun turli algoritmlar qo'llaydi. Rejalashtirish algoritmlari quyidagilardir:

1. **Birinci kelgan — birinci xizmat (First-Come, First-Served - FCFS):** Jarayonlar navbatga qo'yilib, birinci kelgan birinci bajariladi.
2. **Eng qisqa jarayon birinci (Shortest Job Next - SJN):** Eng qisqa bajarilish vaqti talab qiladigan jarayon birinchi bajariladi.
3. **Navbatlash (Round Robin):** Har bir jarayon vaqt kvantiga ega bo'lib, navbatma-navbat protsessorida bajariladi.
4. **Eng qisqa qolgan vaqt birinci (Shortest Remaining Time):** Eng qisqa qolgan bajarilish vaqti bo'lgan jarayon birinchi bajariladi.
5. **Ikkilamchi navbatlash (Multilevel Queue Scheduling):** Jarayonlar bir nechta navbatlarga bo'linib, har bir navbat o'z rejalashtirish algoritmiga ega bo'ladi.

Jarayonni yaratish va tugatish

- **Yaratish:** Yangi jarayon yaratish uchun operatsion tizim maxsus tizim chaqiruvlari (masalan, Unixda fork()) orqali yangi jarayon yaratadi. Yaratilgan jarayon ota-jarayon (parent process) va bola-jarayon (child process) o'rtasida resurslarni taqsimlaydi.
- **Tugatish:** Jarayon bajarilishi tugallanganda yoki dasturiy xato yuz berganda jarayon tugatiladi. Tizim chaqiruvlari (masalan, Unixda exit()) orqali jarayon o'z ishini tugatadi.

Jarayonlararo muloqot (Inter-process Communication - IPC)

Jarayonlar bir-biri bilan ma'lumot almashishi zarur bo'lgan holatlarda IPC mexanizmlari qo'llaniladi. IPC vositalari quyidagilardan iborat:

- **Signallar (Signals):** Jarayonlar o'rtasida signallar yuborish orqali muloqot qilish.

- **Pochta qutilari (Message Queues):** Jarayonlar orasida xabarlar navbati tashkil qilish.
- **Xotirani bo'lishish (Shared Memory):** Bir nechta jarayonlar bir xotira segmentini bo'lishib, ma'lumotlarga birgalikda kirish.
- **Quvurlar (Pipes):** Bir jarayondan boshqa jarayonga ma'lumot uzatish uchun quvurlar ishlatish.

Jarayonlarni boshqarish tizimi komponentlari

- **Jarayon boshqarish bloki (Process Control Block - PCB):** Har bir jarayonning holati, registrlari, xotira manzillari va boshqa ma'lumotlarni saqlaydi.
- **Rejalashtiruvchi (Scheduler):** Jarayonlarni protsessorda bajarilish tartibini belgilaydi.
- **Kontekst almashish (Context Switching):** Bir jarayondan boshqasiga o'tishda jarayonlarning kontekstini saqlash va tiklash jarayoni.

Jarayonlarni boshqarish operatsion tizimning samarali va barqaror ishlashini ta'minlaydi. Bu jarayonlarning to'g'ri tartibda bajarilishi, resurslarning to'g'ri taqsimlanishi va jarayonlararo muloqotni boshqarish kabi muhim vazifalarni amalga oshiradi.

S16.Vazifalarni rejalashtirish algoritmi

Vazifalarni (jarayonlarni) rejalashtirish algoritmlari operatsion tizimda muhim rol o'ynaydi. Ularning asosiy maqsadi cheklangan resurslardan (protssessor, xotira, disk, tarmoq va h.k.) samarali foydalanishni ta'minlashdir.

16 vazifalarni rejalashtirish algoritmi quyidagi asosiy xususiyatlarga ega:

1. **Maqsad** - Vazifalarni rejalashtirishda eng kam o'rtacha kutish vaqtini ta'minlash.
2. **Ishchilar soni** - Algoritm 16 ta vazifani boshqaradi. Bu deyarli har doimgi amaliy miqdor hisoblanadi.
3. **Rejalashtirishni amalga oshirish** - Vazifalar navbatga joylashtiriladi. Ularni rejalashtirishda quyidagi qoidalardan foydalaniladi: Ilk 4 ta vazifa yuqori prioritetga ega. Qolgan 12 ta vazifa past prioritetga ega. Yuqori prioritetli vazifalar past prioritetli vazifalardan ustun. Vazifalar orasidagi o'tish keskin bo'ladi (joylashtirilgan joyida tugaydi).

Natijalar - Yuqori prioritetli vazifalar past prioritetlilardan tez bajariladi. O'rtacha kutish vaqti past prioritetli vazifalar uchun katta bo'ladi. Umumiy rejalashtirishda samaradorlik yuqori.

Vazifali rejalashtirish algoritmi operatsion tizimlarning real vaqt rejimida ishlashi uchun yaxshi mos keladi. U maqsadga muvofiq va amaliy jihatdan qulay algoritm hisoblanadi. Operatsion tizimlarda vazifalarni rejalashtirish algoritmlari jarayonlarni va iplarni (threads) boshqarishda muhim rol o'ynaydi. Ular protssessor vaqtini samarali taqsimlash va tizim samaradorligini oshirish uchun ishlatiladi. Quyida eng mashhur 16 rejalashtirish algoritmi haqida qisqacha ma'lumot berilgan: Bu rejalashtirish algoritmlari har xil turdagi operatsion tizimlar va ish yuklariga qarab qo'llaniladi. Har bir algoritmning afzalliklari va kamchiliklari mavjud bo'lib, ularni tanlashda tizimning talab va maqsadlariga mos kelishi kerak

S17. Operatsion tizimlarda oqim(threads) tushunchasi

Operatsion tizimlarda oqim (threads) - bu jarayon ichidagi alohida bajarilish oqimlaridan biridir. **Oqim** — bu jarayon ichida mustaqil bajariluvchi yo'l bo'lib, u jarayonning kodini, ma'lumotlarini va boshqa resurslarini bo'lishadi. Har bir jarayon kamida bitta asosiy oqimga ega bo'ladi, lekin bir nechta qo'shimcha oqimlar ham yaratishi mumkin.

Oqimlarning afzalliklari

1. **Tezkorlik va samaradorlik:** Oqimlar bir xil xotira maydonini bo'lishgani uchun jarayonlar orasida kontekst almashinuvi (context switching) kamroq vaqt oladi.
2. **Parallelizm:** Oqimlar ko'p protsessorli tizimlarda parallel bajarilishi mumkin, bu umumiy ish samaradorligini oshiradi.
3. **Resurslarni bo'lishish:** Oqimlar bir xil xotira maydonidan foydalanadi, bu ma'lumotlar almashinuvini tezroq va samaraliroq qiladi.

Oqimlarning turlari

1.Foydalanuvchi darajasidagi oqimlar (User-level Threads) - Foydalanuvchi darajasida kutubxonalar yordamida boshqariladi. Oqimlarning rejalashtirilishi va boshqarilishi operatsion tizimdan mustaqil. Kamchilik: Bir oqim to'xtasa, butun jarayon to'xtashi mumkin.

Yadro darajasidagi oqimlar (Kernel-level Threads) = Operatsion tizim yadrosi tomonidan boshqariladi. Oqimlar yadro tomonidan rejalashtiriladi va boshqariladi. Afzallik: Bir oqim to'xtasa, boshqa oqimlar ishlashda davom etishi mumkin.

Oqimlar quyidagi asosiy xususiyatlarga ega:

1. Mustaqil bajarilish - Oqimlar mustaqil ravishda bajariladi va bir-birini bloklash yoki kutish holatiga olishi mumkin.
2. Umumiy resurslardan foydalanish - Oqimlar jarayon ichida umumiy resurslardan (xotira, fayllar va h.k.) foydalanadi.
3. Yengil yaratish va almashish - Oqimlarni yaratish va almashtirish jarayonlarga qaraganda tezroq amalga oshadi.
4. Sinxronizatsiya - Oqimlar o'rtasida sinxronizatsiya muhim, chunki ular umumiy resurslardan foydalanadi.

Oqimlar operatsion tizimda keng qo'llaniladi, jumladan grafik interfeyslarni, web-serverlarni, ma'lumotlar bazalarini ishlab chiqishda.

Operatsion tizimlarda oqimlar (threads) tushunchasi jarayonlarni yanada samarali boshqarish va ishlatish uchun kiritilgan. Oqimlar bir jarayon ichida bir nechta

bajariluvchi yo'llarni ifodalaydi va ular o'zaro resurslarni bo'lishadi. Bu tushuncha ko'p ipli (multithreading) dasturlashda asosiy element hisoblanadi.

Oqimlarni yaratish va boshqarish

Oqimlar yaratish va boshqarish uchun turli xil APIlar mavjud, masalan, POSIX Threads (Pthreads) Unix tizimlari uchun keng qo'llaniladi. Quyida Pthreads yordamida oqim yaratish misoli keltirilgan:

S18.Operatsion tizimlarda deadlock tushunchasi

Operatsion tizimlarda deadlock - bu bir necha jarayonlar yoki oqimlar bir-birlarini bloklab qolishlari natijasida yuzaga keladigan holat.

Deadlocklar quyidagi sharoitlarda yuzaga keladi:

1. **Resurslardan monopol foydalanish:** Jarayon yoki oqim bir resursni yagona egallab olishi.
2. **Resurslarni kutish:** Jarayon yoki oqim egallab turgan resursni boshqasiga berishni kutishi.
3. **Resurslarning kechikishi:** Biron bir sabab tufayli resurslarning kechikishi.
4. **Sirkulyar kutish:** Jarayonlar yoki oqimlar o'zaro bir-birini kutishi.

Deadlocklar operatsion tizimlarda keng tarqalgan muammo bo'lib, ularni bartaraf etish uchun quyidagi yondashuvlar mavjud:

1. **Deadlockni oldini olish:** Tizimdagi resurslardan foydalanishni nazorat qilish va ularning monopolizatsiyasiga yo'l qo'ymaslik.
2. **Deadlockni aniqlash:** Tizimda deadlockni mavjudligini aniqlab, uning sabablarini aniqlash.
3. **Deadlockdan chiqish:** Deadlockka tushib qolgan jarayonlarni to'xtatish yoki resurslardan mahrum qilish orqali tizimni deadlockdan chiqarish.

Operatsion tizimlar deadlockka qarshi turli yondashuvlarni amalga oshiradi, lekin u hamon amaliy muammolardan biri hisoblanadi. Operatsion tizimlarda "deadlock" (o'lik holat) tushunchasi tizimdagi resurslar va jarayonlar o'rtasidagi o'zaro bog'liqlik natijasida yuzaga keladigan holatni ifodalaydi. Bu holatda bir nechta jarayonlar bir-birining resurslarini kutib, cheksiz kutilish holatiga tushib qoladi. Deadlock yuz berganida, jarayonlar davom etolmaydi va tizim samaradorligi sezilarli darajada pasayadi.

Deadlockning 4 ta sharti

Deadlock yuzaga kelishi uchun quyidagi to'rtta shart bir vaqtda bajarilishi kerak:

1. **Mutual Exclusion (O'zaro istisno):** Resurslar bitta jarayon tomonidan eksklyuziv ravishda foydalaniladi, ya'ni bir vaqtning o'zida faqat bitta jarayon resursni egallashi mumkin.
2. **Hold and Wait (Ushlab turish va kutish):** Jarayon kamida bitta resursni ushlab turadi va boshqa resursni olishni kutmoqda.
3. **No Preemption (Majburlab olib qo'ymaslik):** Resurslar jarayondan majburan olib qo'yilmaydi, jarayon resurslarni o'z ixtiyori bilan bo'shatishi kerak.
4. **Circular Wait (Aylanma kutish):** Jarayonlar o'zaro resurslarni kutib, aylanma kutish holatini hosil qiladi, ya'ni jarayon A resurs B ni, B resurs C ni va C resurs A ni kutmoqda.

Deadlockni oldini olish va hal qilish usullari

1. Deadlockni oldini olish (Deadlock Prevention) Deadlockning to'rt shartidan kamida bittasini buzish orqali deadlockning oldini olish mumkin:

- **Mutual Exclusionni buzish:** Ba'zi resurslarni o'zaro istisno bo'lmagan resurslar sifatida ishlatish.
- **Hold and Waitni buzish:** Jarayonlar resurslarni oldindan talab qilishlari yoki resurslar bo'shaganda barcha kerakli resurslarni olishlari va undan keyin ishni boshlashlari.
- **No Preemptionni buzish:** Jarayonlar egallagan resurslarini vaqtinchalik bo'shatishga majbur qilish.
- **Circular Waitni buzish:** Resurslarni buyurtma bo'yicha berish, ya'ni jarayonlar resurslarni oldindan belgilangan tartibda talab qilishlari kerak.

2. Deadlockni oldini olish (Deadlock Avoidance)

Deadlockni oldini olishda jarayonlarning resurslarga talablarini oldindan bilish va tahlil qilish orqali deadlock yuz berishini oldini olish mumkin. Bunga misol:

- **Banker's Algorithm (Bankir algoritmi):** Jarayonlar resurs talablarini bildirganda, tizim bu talabni qanoatlantirishdan oldin tizimni xavfsiz holatda saqlay olishini tahlil qiladi. Agar xavfsiz holat bo'lsa, talab qanoatlantiriladi, aks holda talab rad etiladi.

3. Deadlockni aniqlash va davolash (Deadlock Detection and Recovery)

Agar deadlockni oldini olish yoki oldini olish imkoni bo'lmasa, tizimda deadlockni aniqlash va hal qilish usullari qo'llaniladi:

- **Deadlock Detection (Deadlockni aniqlash):** Tizim muntazam ravishda deadlockni aniqlash algoritmlari yordamida deadlock yuz berishini tekshiradi.
- **Deadlock Recovery (Deadlockni davolash):** Deadlockni aniqlagandan so'ng, tizim bir yoki bir nechta jarayonlarni tugatish yoki resurslarni majburiy ravishda bo'shatish orqali deadlockni hal qiladi.

Misol: Bankir algoritmi Deadlock operatsion tizimlarda jiddiy muammo hisoblanadi, chunki u tizimning to'xtab qolishiga olib kelishi mumkin. Shu sababli, deadlockni oldini olish, aniqlash va hal qilish uchun turli usullar va algoritmlar qo'llaniladi. Har bir usulning afzallik va kamchiliklari mavjud bo'lib, ularni tizim talablariga qarab tanlash kerak

S19. Operatsion tizimlarda xotirani boshqarish

Operatsion tizimlar xotirani samarali boshqarish uchun quyidagi asosiy mexanizmlarni qo'llaydi:

Xotira bo'lish Xotira turli hajmdagi bo'laklarga (sahifalar, bloklar) bo'linadi. Har bir jarayon o'z virtual xotirasiga ega. Virtual xotira fizik xotiraga kartalar orqali bog'lanadi.

Operatsion tizimlarda xotirani boshqarish tizim resurslarini samarali taqsimlash va jarayonlarni tezkor xotirada (RAM) ishlatish imkoniyatini beradi. Xotirani boshqarishning asosiy maqsadi jarayonlarga kerakli xotira maydonini taqdim etish, ularni himoyalash, samarali foydalanish va tizimning umumiy samaradorligini oshirishdan iborat. Quyida xotirani boshqarishning asosiy tamoyillari va usullari keltirilgan:

Xotirani boshqarishning asosiy tamoyillari

1. **Xotira taqsimoti (Memory Allocation):** Jarayonlarga xotira maydonlarini taqsimlash va ularni qayta bo'shatish.
2. **Xotira himoyasi (Memory Protection):** Bir jarayonning xotirasiga boshqa jarayonlar kira olmasligini ta'minlash.
3. **Virtual xotira (Virtual Memory):** Fizik xotira cheklovlaridan foydalanib, katta hajmdagi xotira maydonlarini yaratish.
4. **Fragmentatsiyani boshqarish (Fragmentation Management):** Xotirani samarali taqsimlash va foydalanish uchun fragmentatsiyani minimallashtirish.

Xotirani boshqarish usullari

1. Statik xotira taqsimoti (Static Memory Allocation) - Jarayonlar yaratish vaqtida xotira maydonlarini oldindan belgilangan hajmda ajratiladi. Afzalliklari: Oddiy va oson amalga oshiriladi. Kamchiliklari: Xotira samaradorligi past, jarayonlar o'rtasida xotira almashinuvi qiyin.
2. Dinamik xotira taqsimoti (Dynamic Memory Allocation) - Jarayonlar ishlayotgan paytda xotira maydonlarini ajratish va bo'shatish. Afzalliklari: Xotira samaradorligini oshiradi. Kamchiliklari: Fragmentatsiya muammosi, boshqarish murakkabligi.
3. Boshqariladigan xotira taqsimoti (Contiguous Memory Allocation) - Jarayonlarga uzluksiz xotira maydonlari ajratiladi. Afzalliklari: Oddiy boshqarish, past kechikish. Kamchiliklari: Tashqi fragmentatsiya, katta jarayonlar uchun xotira topish qiyin.

4. Bo'limlash (Paging)- Xotira bo'limlarga (sahifalarga) bo'linadi va har bir jarayonga sahifalar ajratiladi. Afzalliklari: Fragmentatsiyani kamaytiradi, xotira samaradorligini oshiradi. Kamchiliklari: Sahifa jadvalini boshqarish, sahifa almashinuvi kechikishi.
5. Segmentatsiya (Segmentation) - Xotira mantiqiy segmentlarga bo'linadi va har bir jarayon o'z segmentlariga ega bo'ladi. Afzalliklari: Mantiqiy struktura, fragmentatsiyani kamaytiradi. Kamchiliklari: Segment jadvalini boshqarish murakkabligi.
6. Virtual xotira (Virtual Memory) - Fizik xotiradan tashqarida, diskda saqlanadigan qo'shimcha xotira maydonini yaratadi. Afzalliklari: Jarayonlar katta xotira maydonlaridan foydalanishi mumkin, xotira samaradorligini oshiradi. Kamchiliklari: Sahifa almashinuvi kechikishi, tizimning umumiy tezligi pasayishi mumkin.

Virtual xotira boshqarish usullari

1. Paging (Sahifalash) - Jarayonlar va fizik xotira sahifalarga bo'linadi. Har bir jarayon sahifa jadvali orqali fizik xotiraga xarita qilinadi. Tizim sahifa almashinuvi yordamida xotira samaradorligini oshiradi.
2. Segmentation (Segmentatsiya) - Jarayonlar segmentlarga bo'linadi va har bir segment alohida boshqariladi. Segment jadvali yordamida jarayon segmentlari fizik xotiraga xarita qilinadi. Segmentatsiya sahifalash bilan birlashtirilishi mumkin (Segmented Paging).

Xotirani boshqarish operatsion tizimlarda resurslarni samarali taqsimlash va jarayonlarning samarali ishlashini ta'minlash uchun muhimdir. Har xil xotirani boshqarish usullari va strategiyalari mavjud bo'lib, ularning tanlovi tizim talablari va dasturiy ta'minot tuzilishiga bog'liq. Sahifalash va segmentatsiya kabi usullar yordamida xotira samaradorligini oshirish va fragmentatsiyani kamaytirish mumkin.

S20.Xotirani segmentli tashkillash

Segment — bu jarayonning mantiqiy qismi bo‘lib, u kod, ma'lumotlar yoki stack kabi turli komponentlardan iborat bo‘lishi mumkin. Har bir segment o‘zining bazaviy manzili va uzunligiga ega.

Xotirani segmentli tashkillash - bu xotirani alohida bloklarga (segmentlarga) bo'lish asosida xotirani boshqarish usullaridan biridir. Segmentli tashkillash asosiy xususiyatlari:

1. **Segmentlar** - Xotira bloklari bo'lib, har bir blokning o'ziga xos manzili va chegaralari bor.
2. **Segment registri** - Har bir jarayonga oid segmentning manzili, kattaligi va boshqa parametrlarini saqlovchi maxsus registr.
3. **Segment xaritasi** - Tizimda mavjud segmentlarning umumiy manzil va chegaralarini aks ettiruvchi ma'lumotlar tuzilmasi.
4. **Segmentlar orasidagi o'tish** - Jarayonlar ushbu segmentlar orasida qo'shimcha talablar bo'yicha o'tadi.

Segmentli tashkillashtirish afzalliklari:

- **Xotira muhofazasi** - Har bir jarayon faqat o'z segmentlari bilan ishlashi va boshqalarnikiga kirolmasligi.
- **Xotiradan foydalanish samaradorligi** - Kichik segmentlar orqali resurslardan tejimli foydalanish.
- **Xotiraning qo'shimcha xususiyatlari** - Segmentlarga turli xususiyatlar (ruxsat, xavfsizlik va boshqalar) berish imkoni.
- **Modulli loyihalashtirish** - Har bir jarayon o'z segmentlari bilan alohida ishlashi tufayli modulli dasturlash imkoni.

Segmentli tashkillash asosan 16 va 32 bitli operatsion tizimlarda keng qo'llaniladi. Bugungi kunda xotiraning boshqarilishi asosan sahifali (paging) tizim orqali amalga oshiriladi, lekin segmentli yondashuvdan ham foydalaniladi.

Xotirani segmentli tashkillash — bu xotirani boshqarishning yana bir usuli bo‘lib, u jarayonlar xotirasini mantiqiy segmentlarga bo‘lish orqali amalga oshiriladi. Har bir segment o‘zining xotira blokiga ega bo‘ladi va ular o‘zaro bog‘liq bo‘lishi shart emas. Bu usul jarayonlarning mantiqiy tuzilishini aks ettiradi va xotira bo‘linishining aniq va samarali boshqarilishini ta'minlaydi.

Segmentatsiyaning afzalliklari

1. **Mantiqiy bo‘linish**: Segmentatsiya jarayonlarni mantiqiy qismlarga bo‘lishni osonlashtiradi.
2. **Himoya va foydalanish huquqlari**: Har bir segment uchun alohida himoya va foydalanish huquqlari belgilanishi mumkin.

3. **Bo'sh xotiradan foydalanish:** Segmentatsiya tashqi fragmentatsiyani minimallashtirishga yordam beradi.

Segmentli xotira boshqarish usuli

1. Segment jadvali = Har bir jarayon o'zining segment jadvaliga ega bo'ladi. Segment jadvali segmentlar haqidagi ma'lumotlarni saqlaydi, jumladan:

- Segmentning bazaviy manzili (Base address)
- Segmentning uzunligi (Limit)

2. Segmentlarga murojaat qilish = Jarayon xotira manziliga murojaat qilganda, bu murojaat segment raqami va ofset (segment ichidagi joylashuv) ko'rinishida bo'ladi. Segment jadvali yordamida bazaviy manzil topiladi va ofset bazaviy manzilga qo'shiladi. Agar ofset segment uzunligidan oshmasa, to'g'ri manzil olinadi.

Segmentatsiyaning ishlash tartibi

1. Segmentni aniqlash: Jarayon xotiraga murojaat qilganda segment raqamini aniqlaydi.
2. Segment jadvali tekshiruvi: Segment raqami segment jadvalida tekshiriladi va segment mavjudligi hamda ofsetning uzunlikka mos kelishi aniqlanadi.
3. Fizik manzilni hisoblash: Bazaviy manzilga ofset qo'shiladi va fizik manzil olinadi.

Segmentli xotira boshqarish usuli jarayonlarning mantiqiy tuzilishini aks ettiradi va xotirani samarali boshqarish imkonini beradi. Segmentatsiya mantiqiy bo'linishni ta'minlaydi va har bir segmentga alohida himoya va foydalanish huquqlarini belgilash imkonini beradi. Segmentli sahifalash usuli esa segmentatsiya va sahifalashning afzalliklarini birlashtiradi, fragmentatsiyani kamaytiradi va xotira samaradorligini oshiradi.

S21.Virtual xotira

Virtual xotira - bu jarayonlar uchun yaratilgan xayoliy (virtual) xotira makonini ifodalovchi operatsion tizim tushunchasi.

Virtual xotira tizimining asosiy xususiyatlari quyidagilar:

1. **Virtual manzillar:** Har bir jarayon o'z virtual manzillariga ega bo'ladi. Bu manzillar jarayonlar uchun bir xil ko'rinadi, lekin ular fizik xotira manzillariga kartalar yordamida bog'lanadi.
2. **Sahifalarga bo'lish:** Fizik xotira sahifalarga bo'linadi va ular virtual manzillarga kartalar orqali biriktiriladi.
3. **Sahifalarning almashtirilishi:** Kam foydalaniladigan sahifalar diskka yoziladi, tez foydalaniladigan sahifalar esa xotiraga yuklana boshlaydi.
4. **Xotira boshqarish siyosati:** Operatsion tizim sahifalarni xotiraga yuklash va diskka yozish siyosatini amalga oshiradi.

Virtual xotira tizimi afzalliklari:

- **Xotiradan samarali foydalanish:** Jarayonlar cheklangan fizik xotiradan foydalanishi mumkin.
- **Xotiraning mustaqilligi:** Jarayonlar o'z virtual xotiralarida mustaqil ishlaydi.
- **Xotiraning modulli tuzilishi:** Operatsion tizim xotira boshqarishini jarayonlardan yashiradi.

Virtual xotira tizimi kamchiliklari:

- **Xotira sahifalarining almashtirilishi:** Sahifalarning xotiraga yuklanishi va diskka yozilishi vaqt sarfi bilan bog'liq.
- **Sahifalararo o'tish:** Har bir sahifalararo o'tish kechikish vaqtini talab qiladi.

Zamonaviy operatsion tizimlar virtual xotira tizimlaridan keng foydalanadi. Bu tizimlar xotirani samarali boshqarish va jarayonlarning mustaqil ishlashini ta'minlaydi.

Virtual xotira faollashtirilishi

1. **Jarayonning xotirasi:** Jarayon ishga tushishi bilan, uning virtual xotirasi tizim tomonidan aniqlanadi. Virtual xotira jadvali yordamida, jarayonning talab etgan xotira maydoni beriladi.
2. **Diskdan xotira o'qish/yozish:** Agar ma'lumotlar diskda bo'lsa, virtual xotiradan u diskga yoziladi yoki o'qiladi.
3. **Faraziy xotira ishlatilishi:** Agar ma'lumot diskda bo'lmasa, jarayon virtual xotiraga murojaat qiladi va ko'rsatilgan ma'lumotlar saqlanadi.
4. **Fizik xotira aloqasi:** Virtual xotira jadvali yordamida virtual xotira bloklari fizik xotira bloklari bilan bog'lanadi.

Virtual xotira operatsion tizimlarda ma'lumotlar o'zgarishi va saqlanishi uchun qo'shimcha xotira maydonlarini taqdim etadi. Bu, jarayonlarga katta miqdorda xotira berish.

S22.Operatsion tizimlarda fayl tizimlari

Operatsion tizimlarda fayl tizimlari fayllar va direktoriyalarni tashkil etish, saqlash va boshqarish uchun ishlatiladi. Fayl tizimlarining asosiy xususiyatlari quyidagilar:**Fayl tarkibi va tuzilishi:** Fayllar - ma'lumotlar va kod saqlash uchun asosiy birliklar. Direktoriyalar - fayllar va boshqa direktoriyalarni tashkil etish uchun ishlatiladi. Fayl tarkibi - ma'lumotlar, meta ma'lumotlar (nomi, kattaligi, yaratilgan vaqti va h.k.).

1. **Fayl nomlari va yo'llar:** Fayllar va direktoriyalar nom orqali identifikatsiya qilinadi. Yo'llar - fayllar yoki direktoriyalargacha bo'lgan to'liq manzil (masalan, /home/user/file.txt).
2. **Fayl operatsiyalari** - Yaratish, o'chirish, kochirish, ko'chirish kabi amallar. Ruxsatlar (faqat o'qish, yozish, bajarish). Fayl atributlari (yaratilgan vaqt, o'zgartirish vaqti, egasi va h.k.).
3. **Katalog (direktoriya) tuzilishi** - Fayllar va direktoriyalar daraxtsiman tuzilmada joylashtiriladi. Ildiz (root) direktoriya - eng yuqori darajadagi direktoriya.
4. **Fayl tizimi turi** FAT, NTFS, ext4, XFS, ZFS - turli operatsion tizimlar uchun ishlab chiqilgan fayl tizimi turlari. Har bir turi o'ziga xos xususiyatlarga ega.

Fayl tizimi operatsion tizimning muhim tarkibiy qismi bo'lib, fayllar va direktoriyalarni saqlash, boshqarish va nazorat qilish uchun xizmat qiladi. Zamonaviy operatsion tizimlarda fayl tizimining tuzilishi, operatsiyalari va xususiyatlari o'ta muhim ahamiyatga ega.

Operatsion tizimlarda fayl tizimlari, ma'lumotlarni saqlash, o'qish va yozish, fayllarni boshqarish va murojaat qilishga imkon beruvchi tizimlardir. Bu tizimlar, fayllarni tashqi qurilmalarda (masalan, diskda) saqlab turishga, ular bilan amalga oshiriladigan amallarni boshqarishga imkon beradi. Fayl tizimlari, tizim ichidagi barcha amallar uchun muhimdir, shu jumladan, fayllarni yaratish, ochish, yopish, o'qish, yozish, ko'chirish, o'chirish va fayllar orasidagi o'zgarishlarni kuzatish va foydalanuvchiga murojaat qilish.

Fayl tizimining asosiy funksiyalari

1. **Fayllarni yaratish va ochish:** Fayllarni yaratish, ochish va ularga murojaat qilish uchun funksiyalar.
2. **Fayllarga murojaat:** Fayllarga murojaat qilish, o'qish va yozish uchun funksiyalar.
3. **Fayllarni boshqarish:** Fayllarni ko'chirish, o'chirish, nomini o'zgartirish va yo'nalishlarini o'zgartirish uchun funksiyalar.
4. **Fayl tizimining tizim faoliyati:** Fayl tizimi faoliyati, yani fayllarni yaratish, o'qish, yozish va fayllar orasidagi boshqa operatsiyalar uchun funksiyalar.

5. **Fayllar va direktoriyalarning boshqarilishi:** Fayllar va direktoriyalarning strukturasini saqlash va ular bilan ishlash uchun funksiyalar.
6. **Fayl tizimi himoyasi:** Fayllarni va fayl tizimini himoya qilish funksiyalari, masalan, fayllarning ruxsatlarini boshqarish.
7. **Fayl tizimi uchun muloqotlar:** Fayl tizimining qurilmalari va yuklanadigan disklar bilan muloqotlar uchun funksiyalar.

Fayl tizimining turli turlari

1. **Hierarchical File System (Ierarkhik Fayl Tizimi):** Fayllarni ierarkhik tarzda (masalan, papka va ichki papka) saqlaydigan tizim.
2. **Distributed File System (Tarqalib qo'yilgan Fayl Tizimi):** Ma'lumotlarni qo'shuvchan qurilmalar orqali saqlaydigan tizim, masalan, tarmoq yoki bulut asosida.
3. **Network File System (Tarmoq Fayl Tizimi):** Tarmoq orqali murojaat qilish uchun fayl tizimi.
4. **File Allocation Table (FAT):** Odatda kichik hajmdagi xotira uchun ishlatiladi va uning muharririda joylashtirilgan fayllar haqida ma'lumotlar o'rnatilgan, bu fayl tizimi bilan birga yana bog'liq disk xotira usullaridan biri bo'lib, maxsus FAT qurilmalari orqali qo'llaniladi.
5. **NTFS (New Technology File System):** Bu Windows operatsion tizimlari uchun o'rnatilgan fayl tizimi shaklidir. U yaxshi himoyalangan, kengaytirilgan funksiyalarga ega va katta hajmdagi fayllarni boshqarishda yaxshi ishlaydi.

Fayl tizimlari, operatsion tizimlarda ma'lumotlarni saqlash va boshqarishning muhim qismlaridan birini ta'minlaydi. Fayl tizimlarining funksiyalari, fayllarni yaratish, ochish, yopish, o'qish, yozish, fayllarni boshqarish va fayllar orasidagi o'zgarishlarni kuzatishga imkon beradi. Fayl tizimlari turli turdagi xususiyatlarga ega bo'lgan turli turlari mavjud bo'lib, ular foydalanuvchi talablari va maqsadlari bo'yicha foydalaniladi.

S23.FAT, VFAT, FAT32 va NTFS

Bu operatsion tizimlar uchun ishlab chiqilgan fayl tizimi turlaridir:

FAT (File Allocation Table) - FAT, Microsoft tomonidan asoslangan va birinchi marta 1981 yilda IBM PC-DOS 1.0 bilan ishlatilgan fayl tizimidir. U Windows operatsion tizimlarida keng qo'llanilgan bo'lib, disk operatsiyalarini bajarishda qo'llaniladi. Fayllarni joylashtirish va fayl bo'limlarini yozish uchun yordam beradi. FAT fayl tizimi, hujjatlarni qaysi blokka joylashtirilganligini va qaysi bloklar bo'shligi bo'lib qolganini saqlash uchun fayl qurilmalari o'rnida joylashgan ma'lumotlar jadvali bilan belgilanadi. **FAT** Eng eski va oddiy fayl tizimi turi. 8.3 formatidagi qisqa fayl nomlarga ega. Maksimal fayl hajmi - 4 GB. Maksimal disk hajmi - 2 GB (FAT16).

VFAT (Virtual FAT) - VFAT, FAT32-ga qo'shimcha funktsiyalar qo'shgan o'zgaruvchan fayl tizimidir. U katta hajmli diskdagi fayllar va papkalarni qo'llashga imkon beradi va Windows 95 va keyingi versiyalarda foydalanilgan. VFAT, uzun fayl nomlari (boshqa so'zlar bilan 255 harf yoki undan kam emas) va fayl nomlarida kichik xotira harflari bilan ishlatiladigan diskda o'zgaruvchanliklar kiritilgan. FAT fayl tizimining kengaytirilgan versiyasi. Uzun fayl nomlarga (256 ta belgigacha) ruxsat beradi. Maksimal fayl hajmi - 4 GB. Maksimal disk hajmi - 32 GB. VFAT fayl tizimining yanada kengaytirilgan versiyasi. Maksimal fayl hajmi - 4 GB. Maksimal disk hajmi - 2 TB. Windows XP va keyingi versiyalarda qo'llab-quvvatlanadi.

FAT32 - FAT32, FAT ning kengaytirilgan versiyasi bo'lib, katta hajmli diskdagi fayllarni qo'llash uchun qo'llaniladi. U FAT-ga nisbatan keng fayl nomlarini (uzun fayl nomlarini) va katta hajmli disk maydonlarini qo'llaydi. FAT32 fayl tizimi Windows 95 va undan keyingi versiyalarda va boshqa operatsion tizimlarda (masalan, Linux va Mac OS) foydalaniladi. Microsoft tomonidan Windows NT/2000 uchun ishlab chiqilgan. Keng qamrovli, kuchli va xavfsiz fayl tizimi. Uzun fayl nomlarga (256 ta belgigacha) ruxsat beradi. Maksimal fayl hajmi - 16 EB (exabyte). Maksimal disk hajmi - 256 TB. Kengaytirilgan xususiyatlar (ruxsatlar, shovqinlar, shifrlash, qutular va h.k.).

NTFS (New Technology File System) - NTFS, Microsoft tomonidan ishlab chiqilgan, ko'p funksiyalarga ega, yuqori darajada xavfsizlikni ta'minlaydigan va boshqarishni yaxshilovchi fayl tizimidir. U katta hajmli diskdagi fayllarni qo'llash va boshqarishga imkon beradi. NTFS, sizga ruxsat beradi - Fayl shifri, Sirtqa saqlash, Katta hajmli fayllarni qo'llash, Fayl texnologiyasini tarkibiy qilish, Fayl ko'chirishni bekor qilishning ta'minlash.

NTFS, ko'p foydalanuvchilarning talablari va xavfsizlik maqsadlari uchun keng qo'llaniladi va ommaviy tizimlarda Windows NT, 2000, XP, Vista, 7, 8, va 10, hamda Windows Server va boshqa operatsion tizimlarda qo'llaniladi. FAT va VFAT keng

tarqalgan bo'lsada, kichik disk hajmi va fayl hajmi cheklovlari bilan ajralib turadi. NTFS esa kuchli va keng qamrovli fayl tizimi bo'lib, zamonaviy operatsion tizimlar uchun moslashtirilgan. Bu turlar FAT (File Allocation Table), VFAT (Virtual FAT), FAT32 va NTFS (New Technology File System) Windows operatsion tizimlarida foydalaniladigan fayl tizimi shakllaridir.

S24.Virtual(VFS) va tarmoq(NFS) fayl tizimlari

Virtual File System (VFS) - Virtual File System (Virtual Fayl Tizimi) - bu operatsion tizimning fayl tizimlarini birlashtiruvchi va standartlashgan interfeysdir. VFS, fayllar tizimini boshqarish uchun yuklab olinadigan keng xil xususiyatlar bilan qo'llaniladi. Bu, tizimning fizikaviy disk bo'lmalariga murojaat qilish uchun yordam beradi va turli tarmoqlar, fayl tizimlari va qurilmalar orasida o'zaro ishlashni ta'minlaydi.

VFS funksiyalari:

1. **Fayl ochish, yopish, o'qish va yozish:** Tarmoq tizimlari, disklar, ishlab chiqarilgan qurilmalar va boshqalar bilan ishlash uchun standart fayl amallarini boshqaradi.
2. **Fayl tizimi jadvali:** Fayllar, direktoriyalar va fayl tizimining boshqa komponentlarini identifikatsiya qilish uchun foydalaniladigan ma'lumotlar jadvali.
3. **Fayl himoyasi:** Fayllarni himoyalash, ruxsat berish va qoldirish funktsiyalari.
4. **Fayl tarmoqlari:** Tarmoqlar orqali fayllarga murojaat qilish va ularni boshqarishni ta'minlaydi.
5. **Disk I/O:** Disk operatsiyalarini (o'qish, yozish, ko'chirish) boshqarish.
6. **Fayl qurilmalari o'rtasida bog'lanish:** O'rtacha fayl tizimining standart interfeysi sifatida foydalaniladi. Masalan, Unix, Linux, va boshqa operatsion tizimlarida VFS foydalaniladi.

Network File System (NFS)

Network File System (Tarmoq Fayl Tizimi) - bu ma'lumotlar almashish tarmog'idagi fayllarga murojaat qilishni ta'minlaydigan protokol. NFS, tarmoq orqali fayllarga murojaat qilish uchun ishlatiladi. Ushbu protokol ma'lumotlarni boshqa tizimlarda joylashtirilgan fayllarga yozish, o'qish va ochish uchun yordam beradi. Masalan, bir kompyuterdan boshqa kompyuterga yoki serverga murojaat qilish imkonini beradi.

NFS funksiyalari:

1. **Fayl ko'rish:** Uzluksiz tarmoq orqali joylashgan fayllarga murojaat qilish.
2. **Fayl yozish:** Tarmoq orqali fayllarga ma'lumot yozish.
3. **Fayl o'qish:** Tarmoq orqali fayllardan ma'lumotlarni o'qish.
4. **Fayl qurilmalari orasida fayl almashish:** Kompyuterlar va serverlar orasida fayl almashishni ta'minlash.
5. **Xavfsizlik:** Ma'lumotlarni xavfsizlik va himoya qilish.
6. **Protokol:** Tarmoq Fayl Tizimi uchun standart protokolni ta'minlash.

VFS va NFS farq

- **VFS**, bir operatsion tizimning fayl tizimlarini birlashtiruvchi va standartlashgan interfeysdir.
- **NFS**, tarmoq orqali fayllarga murojaat qilish uchun ishlatiladigan protokoldir.
- **VFS**, lokal fayl tizimlarini birlashtiradi, boshqa so'z bilan, disk, USB kiritish, fayl sistemasi bilan ishlaydi.
- **NFS**, tarmoq orqali tuzilgan fayl tizimlari bilan ishlaydi va bir kompyuterdan boshqa kompyuterga yoki serverga murojaat qilish imkonini beradi.

Virtual File System (VFS) operatsion tizimning fayl tizimlarini birlashtiruvchi va standartlashgan interfeysdir, u esa lokal fayl tizimlarini birlashtiradi. Network File System (NFS) esa tarmoq orqali fayllarga murojaat qilishni ta'minlaydigan protokol hisoblanadi, ma'lumotlarni boshqa tizimlarda joylashtirilgan fayllarga yozish, o'qish va ochish uchun yordam beradi.

S25. Kirish/chiqish tizimlarini tashkillashtirish

Kirish/chiqish tizimlari (Login/Logout systems) kompyuter tizimlarida foydalanuvchilar yoki foydalanuvchi guruhlarining tizimga kirish va chiqish jarayonlarini boshqarish uchun mo'ljallangan. Ular foydalanuvchilar uchun tizimga kirishni boshqarish va tizimdan chiqishni amalga oshirish uchun kerakli avtomatlashtirilgan vositalarni taqdim etishadi.

Kirish/chiqish tizimlarini tashkillashtirish quyidagi vazifalarni o'z ichiga oladi:

1. **Foydalanuvchilar ro'yxati:** Kirish tizimi foydalanuvchilar ro'yxatini saqlaydi. Bu ro'yxatda foydalanuvchilarning foydalanuvchi nomi, paroli, huquqlari, va boshqalar ma'lumotlari bo'lishi mumkin.
2. **Kimdir kirish huquqini aniqlash:** Foydalanuvchining kiritilgan ma'lumotlarini tekshirib, uning identifikatorini (masalan, foydalanuvchi nomi va paroli) tekshiradi.
3. **Xavfsizlik:** Parolni tasdiqlash va xavfsizlik so'rovini bajarish yoki boshqa tasdiqlash mehmonchiliklarini bajarish orqali tarmoq xavfsizligini ta'minlash.
4. **Foydalanuvchi identifikatorini aniqlash:** Foydalanuvchining kim ekanligini va uning huquqlarini aniqlash uchun uning identifikatorini aniqlash.
5. **Foydalanuvchi sessiyasini boshlash:** Tizimga muvaffaqiyatli kiringan foydalanuvchiga yordam berish, foydalanuvchining ishlash sessiyasini boshlash.
6. **Foydalanuvchi tarixini saqlash:** Foydalanuvchi faoliyati tarixini saqlash va ko'rib chiqish jarayonlarini kuzatish.
7. **Foydalanuvchi sessiyasini yakunlash:** Foydalanuvchining tizimdan chiqilishi, sessiyasini yakunlash va tizimdan to'liq chiqilishni bajarish.
8. **Xavfsizlik:** Tizimni hakerlarga qarshi himoya qilish va xavfsizlikni ta'minlash.

Kirish/chiqish tizimini tashkillashtirish jarayoni quyidagi bosqichlardan iborat bo'ladi:

1. Foydalanuvchi ma'lumotlarini qabul qilish - Foydalanuvchi kiritish uchun ma'lumotlarni qabul qilish, masalan, foydalanuvchi nomi va paroli.
2. Ma'lumotlarni tekshirish - Kiritilgan foydalanuvchi ma'lumotlarini tekshirish, masalan, foydalanuvchi nomi va parolini tasdiqlash, foydalanuvchining ma'lumotlari bazasida mavjudligini va to'g'riligini tekshirish.
3. Kirish huquqlarini berish - Ma'lumotlar to'g'riligini tasdiqlashdan so'ng foydalanuvchiga kerakli kirish huquqlarini berish, masalan, foydalanuvchi ro'yxatidan o'tkazish, sessiya boshlash, ma'lumotlarga kirish huquqlarini olish va boshqalar.

4. Foydalanuvchi sessiyasini boshlash - Foydalanuvchining muvaffaqiyatli kiritilgani va to'g'riligini tasdiqlashdan so'ng, uning ishlash sessiyasini boshlash.

5. Foydalanuvchi faoliyatini kuzatish - Foydalanuvchi sessiyasi davomida foydalanuvchi faoliyatini kuzatish, masalan, qaysi fayllarga murojaat qilganligini, qanday operatsiyalarni bajarganligini, va boshqalar.

6. Foydalanuvchi sessiyasini yakunlash - Foydalanuvchi tizimdan chiqishni so'rab yoki sessiya muddati tugagach sessiyani avtomatik ravishda yakunlash.

7. Xavfsizlikni ta'minlash - Foydalanuvchi ma'lumotlarini himoya qilish, parolni tasdiqlash va avtomatik tizim tomonidan to'g'rilash uchun kerakli xavfsizlik protseduralarini o'rnatish.

8. Xatoliklarni tekshirish va bartaraf etish - Kirish/chiqish tizimi orqali kiritilgan ma'lumotlarni tekshirish, xato mavjud bo'lsa foydalanuvchiga ma'lumot berish.

9. Foydalanuvchi ma'lumotlarini saqlash - Foydalanuvchi ma'lumotlarini (nom, parol, huquqlar va boshqalar) ma'lumotlar bazasida saqlash.

10. Tizimdan chiqishni boshqarish - Foydalanuvchi sessiyasi tugaganida, uni tugatish va tizimdan chiqishni amalga oshirish.

Kirish/chiqish tizimlarini tashkillashtirish, operatsion tizimda foydalanuvchilar uchun turli xil ma'lumotlarni boshqarishni ta'minlaydi. Ular foydalanuvchilar uchun ishlash va tizimdan chiqishni boshqarish jarayonlarini avtomatlashtiradi va xavfsizlikni ta'minlaydi.

S26. Operatsion tizimlarda tarmoq xavfsizligi

Operatsion tizimlarda tarmoq xavfsizligi, tarmoq orqali aylanuvchilar foydalanuvchilar, serverlar, hujjatlar va boshqa tizimlar) orasida axborot almashishni ta'minlash, shuningdek tarmoqni hakerlarga, zararli dasturlarga, ma'lumot qasriroq dasturlarga qarshi himoya qilishni ta'minlaydi. Tarmoq xavfsizligi, tarmoq tuzilmalari, protokollar va muammo yechish protseduralari yordamida ta'minlanadi. Quyidagi jarayonlar tarmoq xavfsizligini ta'minlashda o'z vazifalarini bajaradi:

1. Xavfsizlik protokollari - Tarmoq xavfsizligini ta'minlash uchun xavfsizlik protokollari, masalan, Transport Layer Security (TLS) va Secure Socket Layer (SSL) kabi protokollar keng tarqalgan ko'lamda qo'llaniladi. Bu protokollar ma'lumotlarni xavfsizlik protokollariga muvofiq shifrlash, yuborish va qabul qilishni ta'minlaydi.
2. Foydalanuvchi haqiqiylikni tasdiqlash (Authentication) - Tarmoqdagi foydalanuvchilarning haqiqiylikni tasdiqlash uchun kerakli identifikatsiya va tasdiqlash protokollarlari, masalan, Kerberos, LDAP (Lightweight Directory Access Protocol), va OAuth kabi protokollar qo'llaniladi.
3. Ma'lumotlarni shifrlash (Encryption) - Xavfsizlikni ta'minlash uchun tarmoq trafiklarini shifrlash katta ahamiyatga ega. Bunda, ma'lumotlarni yuborish, qabul qilish va saqlash jarayonlari shifrlanadi. Bu shifrlash protokollariga TLS va SSL kabi protokollar kiradi.
4. Xavfsizlikning ta'minlash - Foydalanuvchi ma'lumotlari va tarmoq trafiklarini himoya qilish uchun yozuvlar, dasturlar va zararli dasturlarga qarshi himoya vositalaridan foydalanish.
5. Xavfsizlikni boshqarish - Tarmoq xavfsizligini boshqarish uchun, hujjatlarni, protokollar va asosiy ta'sirchilar tasdiriladi. Masalan, xavfsizlik siyosati, foydalanuvchilar huquqlari, xavfsizlikni amalga oshirish va tarmoqning xavfsizligini ta'minlash uchun to'g'ri jarayonlar va protseduralar tuziladi.
6. Xavfsizlikning monitoringi va ko'rib chiqish - Tarmoq xavfsizligini monitoring qilish, zararli faoliyatlar (hakerlik, atakalar va boshqa xavfsizlik o'chirish muammolari)ni aniqlash, xavfsizlik ko'rib chiqish va to'g'rijarayonlarni takomillashtirish uchun muhim vazifalardan biridir.
7. Hujjatlarni himoya qilish - Tarmoq xavfsizligi, ma'lumotlarni yuborish va qabul qilishda foydalaniladigan hujjatlarni himoya qilish, masalan, firewall, antivirus dasturlar va boshqa xavfsizlik himoyalari orqali amalga oshiriladi.

8. Foydalanuvchilar haqida ma'lumotlar - Foydalanuvchilar haqida ma'lumotlarni himoya qilish va uni yopish, xavfsizlikni ta'minlash uchun muhimdir.

Tarmoq xavfsizligi, kompyuter tarmoqlarida ma'lumotlarni himoya qilishning juda muhim qismidir. U foydalanuvchilar ma'lumotlarini himoya qilish, foydalanuvchilar identifikatsiyasini va tasdiqlash, tarmoq trafiklarini shifrlash va himoya qilish uchun kerakli protokollar va dasturlar orqali amalga oshiriladi. Tarmoq xavfsizligi, xavfsizlikni monitoring qilish, foydalanuvchilar haqida ma'lumotlarni saqlash, xavfsizlikni boshqarish va hujjatlarni himoya qilish jarayonlari orqali ta'minlanadi.

S27. Bulutli hisoblash tushunchasi.

Bulutli hisoblash (cloud computing) - bu ma'lumotlar va resurslarni internet orqali saqlash, boshqarish va ishlovchi xizmatlarni taqdim etishning elektron tizimlarini (bulut xizmatlarini) ishlatish. Bu, foydalanuvchilar uchun ma'lumotlar va xizmatlarga onlayn ravishda murojaat qilish imkonini beradi. Bulut xizmatlarida, ma'lumotlar va resurslar, biror mahalliy kompyuterda emas, balki bulut tarmoqlari orqali, internet orqali iste'mol qilinadi.

Bulutli hisoblashning muhim xususiyatlari quyidagilardir:

1. Oson foydalanish - Bulut xizmatlari internet orqali onlayn ravishda foydalanuvchilar uchun tez va oson murojaatga imkon beradi. Foydalanuvchilar uchun qurilishni o'rnatish va sozlash kerak emas, ular faqat internetga ulanish uchun kerak.
2. Qulaylik - Bulut xizmatlari, foydalanuvchilar uchun har qanday joyda va har qanday vaqtda murojaat qilishga imkon beradi. Bu, muhitda bo'lib, kompyuterlarni o'rnatish va sozlashni talab qilmaydi.
3. Oson murojaat - Bulut xizmatlari, foydalanuvchilar uchun murojaatga ishonchli bo'lish imkonini taqdim etadi. Internetga ulanish orqali, ma'lumotlar va resurslar iste'molchilar uchun osonlik bilan murojaatga imkon beradi.
4. Mablag'lar bilan tejamkorlik - Bulut xizmatlari, foydalanuvchilar uchun kichikdan katta vaqtning oxiriga qadar kerakli resurslarni sotib olish uchun talab qilmaydi. Ular faqat foydalanilgan resurslarni to'ldirish uchun mablag'lar to'plashadi.
5. Skalalarlik - Bulut xizmatlari, foydalanuvchilar uchun istalgan vaqt istalgan hajmdagi ma'lumotlar va resurslarni taqdim etishga imkon beradi. Bunda, foydalanuvchilar talablarga ko'ra ishlab chiqilgan hajmni oshirish yoki kamaytirishlari mumkin.
6. Avtomatlashtirish - Bulut xizmatlari, avtomatlashtirishni yaxshi qo'llaydi. Bular, murojaat qilish, konfiguratsiyalash, ma'lumotlar o'chirish va boshqa avtomatlashtirilgan operatsiyalarni osonlashtirish uchun avtomatlashtirilgan xizmatlarni taqdim etish orqali foydalanuvchilarga va kompaniyalarga qulaylik beradi.
7. Xavfsizlik - Bulut xizmatlari odatda ma'lumotlar ko'chirishni va saqlashni yoki boshqarishni ta'minlashda yuqori darajada xavfsizlikni ta'minlash uchun keng qamrovli xavfsizlik tizimlarini ishlatadi.

8. Masshtablash - Bulut xizmatlari, boshqa kompyuter tizimlariga nisbatan qulaylik va osonlik bilan masshtablanishga imkon beradi. Bunda, foydalanuvchilar o'z xizmatlari uchun zarur resurslarni osonlik bilan qo'shish yoki kamaytirishlari mumkin.

9. Ishonchli xizmat - Ko'p bulut xizmatlari, foydalanuvchilarga ishonchli va qonun bilan amalga oshirilgan xizmatlarni taqdim etish uchun mablag'lar to'plashga ko'z yumadi.

Bulutli hisoblash, internet orqali murojaat qilish va ma'lumotlarni saqlash, boshqarish va ishlovchi xizmatlarni taqdim etishda keng qo'llaniladi. Ular foydalanuvchilar uchun qulay va qulaydir, chunki ular osonlik va ishonchli xizmatlar taqdim etishga yordam beradi. Bulut xizmatlari xavfsizlik, qulaylik, mablag'lar bilan tejamkorlik va masshtablash imkonini taqdim etadi.

S28. Bulutli xisoblashda xizmatlar, ma'lumotlarni qayta ishlash markazlari

Bulutli xisoblashda ma'lumotlarni qayta ishlash markazlari (data centers) – bu shuningdek "bulut" xizmatlarining asosiy qismlaridan biridir. Ular ma'lumotlarni saqlash, ishlovchi kuchlarni (computing power), internet orqali xizmatlarni taqdim etish va boshqarish uchun ishlatiladi. Bulut xizmatlarining keng doirasini ta'minlash uchun, katta miqdordagi ma'lumotlar va kuchlarni birlashtirish, tarmoqaro xizmatlarni ta'minlash va xavfsizlikni ta'minlash uchun kerakli tizimlarni yaratish va boshqarish uchun bulutli xisoblash markazlari foydalaniladi. Quyidagi bulut xisoblashda xizmatlar, ma'lumotlarni qayta ishlash markazlarini o'z ichiga oladi:

1. Saqlash xizmatlari - Saqlash xizmatlari, foydalanuvchilar uchun ma'lumotlar bazalarini, fayl tizimlarini va boshqa ma'lumotlarni saqlash uchun bulutli xisoblash markazlarida muhit taqdim etadi. Bu xizmatlar orqali, foydalanuvchilar ma'lumotlarini internet orqali bulutda saqlab qolishadi va ularga internet orqali murojaat qilishlari mumkin.
2. Ishlovchi kuchlar (Computing Power) - Bulut xisoblashda ishlovchi kuchlar, foydalanuvchilar uchun internet orqali katta miqdordagi ishlovchi kuchlarni taqdim etish uchun ishlatiladi. Bu xizmatlar orqali, foydalanuvchilar IT muammosini yechish uchun kuchli kompyuterlarni o'zlarida o'rnatish va sozlash kerak emas.
3. Platforma xizmatlari (Platform as a Service - PaaS) - Platforma xizmatlari, foydalanuvchilar uchun tarmoqaro ilovalarni yaratish va boshqarishni osonlashtirish uchun ishlatiladi. Bu xizmatlar orqali, foydalanuvchilar tarmoqaro ilovalarni o'zlarida yaratish uchun loyihalar, dasturlash tillari va asboblari, ma'lumotlar bazalari, aloqa va boshqa xizmatlar bilan ta'minlangan bo'lishi mumkin.
4. Dasturlash xizmatlari (Software as a Service - SaaS) - Dasturlash xizmatlari, tarmoqaro dasturlar va ilovalarni internet orqali foydalanuvchilarga taqdim etish uchun ishlatiladi. Bu xizmatlar orqali, foydalanuvchilar dasturlarni o'rnatish, sozlash va foydalanishi mumkin.
5. Ma'lumotlarni qayta ishlash xizmatlari (Data Processing Services) - Bu xizmatlar, foydalanuvchilar uchun ma'lumotlarni qayta ishlash, to'plamlarini tahlil qilish, statistik ma'lumotlar tayyorlash, ma'lumotlarni shifrlash va boshqa operatsiyalarni bajarish uchun ishlatiladi.
6. Ma'lumotlarni qayta ishlash markazlari (Data Processing Centers) - Bu markazlar, katta miqdordagi ma'lumotlarni saqlash, ishlovchi kuchlarni taqdim etish va boshqa bulut xizmatlarini ta'minlash uchun yaratilgan fizikaviy tizimlardir. Ular ma'lumotlarni ishlovchi, muhafaza qilish va ma'lumotlarni ko'chirish uchun xavfsiz,

ishonchli va kuchli bo'lgan tarmoqaro tizimlar yaratish uchun kerakli texnikaviy infrastrukturani ta'minlaydi.

Bulutli xisoblashda, ma'lumotlarni qayta ishlash markazlari, katta miqdordagi ma'lumotlar va ishlovchi kuchlarni taqdim etish, tarmoqaro xizmatlarni ta'minlash va xavfsizlikni ta'minlash uchun yaratilgan tarmoqaro tizimlardir. Ular bulut xizmatlarining to'g'ri ishlashi uchun muhim qismni tashkil etadi va bulut xizmatlarini foydalanuvchilarga internet orqali oson va qulay taqdim etishda yordam beradi.

S29.Bulutli hisoblash uchun operatsion tizimlar va dasturiy vositalar

Bulutli hisoblash ekotizimi uchun quyidagi asosiy operatsion tizimlar va dasturiy vositalar ishlatiladi:

Operatsion tizimlar:

Linux- Keng tarqalgan ochiq kodli operatsion tizim. Bulut xizmat provayderlari tomonidan keng qo'llaniladi, xususan Amazon EC2, Google Cloud, Microsoft Azure. Yengil, moslashuvchan, xavfsiz va yuqori unumdorlikka ega.

Windows Server- Microsoft tomonidan rivojlantiriladigan operatsion tizim. Bulut xizmatlarida Windows-asosidagi xizmatlarni taqdim etish uchun ishlatiladi. Active Directory, SQL Server kabi Microsoft texnologiyalari bilan yaxshi integratsiyalashgan.

FreeBSD - Ochiq kodli Unix-like operatsion tizim. Yuqori barqarorlik, xavfsizlik va unumdorlikka ega. Bulut serverlarida, ma'lumotlar markazlarida keng qo'llaniladi.

Dasturiy Vositalar:

Vmware - VMware, sanal masinlarni ishga tushirish va boshqarish uchun yaxshi bilan tanilgan bir dasturiy vositadir. Bu, ma'lumotlar omborlari va ishlovchi kuchlarni taqdim etishda o'zgaruvchan ishlatiladi. VMware, bulutli hisoblash muhitida ham qo'llaniladi.

OpenStack - OpenStack, ochiq manbali bulut xizmatlarini ishga tushirish va boshqarish uchun yaratilgan platformadir. Bu, ishlovchi kuchlarni taqdim etish, sanal yordamchi ishlab chiqish, ma'lumotlar omborlari va boshqa xizmatlarni boshqarishda ishlatiladi.

Amazon Web Services (AWS) - AWS, Amazon tomonidan taqdim etilgan qo'llanmalardan biridir va eng mashhur bulut xizmatlardan biridir. AWS, hajmdan katta miqdorda ishlovchi kuchlar, ma'lumotlar omborlari, ma'lumotlar analiz qilish, tezkor tarmoq yordamchi ishlab chiqish va boshqalar uchun xizmatlarni taqdim etadi.

Microsoft Azure - Microsoft Azure, Microsoft tomonidan taqdim etilgan bir qo'llanmadir. U, ishlovchi kuchlar, ma'lumotlar omborlari, tashqi xizmatlarni ishlab chiqish, ma'lumotlar analiz qilish, tezkor tarmoq yordamchi ishlab chiqish va boshqalar uchun xizmatlarni taqdim etadi.

Google Cloud Platform (GCP) - Google Cloud Platform (GCP), Google tomonidan taqdim etilgan qo'llanmadir. U, ishlovchi kuchlar, ma'lumotlar omborlari, ma'lumotlar

analiz qilish, tezkor tarmoq yordamchi ishlab chiqish va boshqalar uchun xizmatlarni taqdim etadi.

IBM Cloud - IBM Cloud, IBM tomonidan taqdim etilgan bulut xizmatlari qo'llanmasidir. U, ishlovchi kuchlar, ma'lumotlar omborlari, ma'lumotlar analiz qilish, tezkor tarmoq yordamchi ishlab chiqish va boshqalar uchun xizmatlarni taqdim etadi.

Bulutli hisoblash uchun operatsion tizimlar va dasturiy vositalar, foydalanuvchilar uchun kuchli va ishonchli bulut xizmatlarni taqdim etishda keng qo'llaniladi. Bu, ma'lumotlarni saqlash, ishlovchi kuchlarni taqdim etish, bulut xizmatlarini boshqarish, ma'lumotlar analiz qilish va boshqa maqsadlar uchun muhim qo'llanmalardir.

S30 Mobil operatsion tizimlar

Mobil operatsion tizimlar, telefonlar, tablet kompyuterlar, smartwatchlar va boshqa ko'chmas elektron qurilmalar uchun mo'ljallangan operatsion tizimlardir. Bu tizimlar, kichik hajmli va kam quvvatli qurilmalar uchun mahsulotiviyat, ishonch va ishlatish osonligini ta'minlash uchun mo'ljallangan. Quyidagi keng tarqalgan mobil operatsion tizimlar mavjud:

Android, Google tomonidan ishlab chiqilgan, dunyoda eng mashhur va ko'p ishlatilgan mobil operatsion tizimlardan biridir. Bu tizim, ko'pchilik smartfon va planshetlarda ishlaydi. Android ochiq manbali tizimdir va ko'p miqdorda ilovalarni, o'zgartirilgan mahsulotlarni va xizmatlarni taqdim etadi. Android, keng tarqalgan foydalanuvchi interfeysi va mahsulotiviyati bilan mashhur.

iOS, Apple tomonidan ishlab chiqilgan va Apple kompaniyasining iPhone, iPad va iPod Touch qurilmalari uchun ishlatiladigan mobil operatsion tizimdir. iOS ochiq manba, amalda, faqat Apple tomonidan ishlatilgan qurilmalarda ishlaydi. iOS mos kelishuvli foydalanuvchi interfeysi, qulaylik va ishonchga ega. App Store, iOS ilovlar uchun eng mashhur dastur do'konidir.

HarmonyOS, Huawei tomonidan ishlab chiqilgan va 2019 yilida taqdim etilgan yangi mobil operatsion tizimdir. Ushbu tizim, telefonlar, planshetlar, smartwatchlar va boshqa qurilmalar uchun ishlatiladi. HarmonyOS, murakkab o'zgaruvchan elektron qurilmalarni birlashtirish va birlashtirish uchun yaratilgan.

KaiOS, kichik mahsulotlar uchun mo'ljallangan mobil operatsion tizimdir. U, kichik hajmli smartfonlarda, masalan, Doro, JioPhone, Nokia 8110 va boshqalarida ishlaydi. KaiOS, Android va iOS tizimlariga nisbatan oson va sodda bo'lgan mobil operatsion tizimdir.

Tizen, Samsung tomonidan ishlab chiqilgan va ko'p miqdorda Samsung smart TV, smartwatch va boshqa smart qurilmalar uchun ishlatiladigan mobil operatsion tizimdir. U, boshqa Samsung qurilmalari bilan integratsiyalanadi va murakkab smart qurilmalarni birlashtirish uchun yaratilgan.

Wear OS, Google va boshqa qurilmalarning jamoasi tomonidan ishlab chiqilgan smartwatchlarda ishlatiladigan mobil operatsion tizimdir. U, smartwatchlarga mahsulotiviyat, foydalanuvchi interfeysi va ko'plab ilovalarni taqdim etadi.

Mobil operatsion tizimlar, telefonlar, planshetlar, smartwatchlar va boshqa ko'chmas elektron qurilmalar uchun mo'ljallangan tizimlardir. Eng mashhur mobil operatsion tizimlar Android va iOS hisoblanadi, lekin boshqa tizimlar ham xususan kichik hajmli qurilmalarda qo'llaniladi. Bu tizimlar, foydalanuvchi interfeysi,

mahsulotiviyati va qulayligi bo'yicha farq qiladi va murakkablik darajasi ham farq qiladi.

S31. IBM, HP, Oracle/Sun va boshqa firmalarning operatsion tizimlari

1. IBM operatsion tizimlar:

- **z/OS** - IBM mainframe qurilmalari uchun mo'ljallangan asosiy operatsion tizim.
- **AIX** - IBM UNIX operatsion tizimi, Linux va Windows bilan raqobatlashadi.
- **i OS** (avvalgi OS/400) - IBM Power tizimlari uchun mo'ljallangan operatsion tizim.

2. HP operatsion tizimlar:

- **HP-UX** - Hewlett Packard tomonidan rivojlantirilgan UNIX operatsion tizimi.
- **NonStop OS** - HP kompaniyasining yuqori ishonchlilikka ega operatsion tizimi.

3. Oracle/Sun operatsion tizimlar:

- **Solaris** - Sun Microsystems tomonidan rivojlantirilgan UNIX-asosidagi operatsion tizim.
- **OpenSolaris** - Solaris operatsion tizimning ochiq kodli versiyasi.

4. Boshqa kompaniyalarning operatsion tizimlar:

- **FreeBSD** - Berkeley Software Distribution (BSD) UNIX oilasiga qarashli operatsion tizim.
- **macOS** - Apple tomonidan rivojlantirilgan operatsion tizim, asosan Macintosh qurilmalari uchun mo'ljallangan.
- **OpenVMS** - Digital Equipment Corporation (DEC) kompaniyasi tomonidan rivojlantirilgan operatsion tizim.

Ushbu operatsion tizimlar asosan korporativ sohada, ma'lumotlar markazlarida, kompyuter tizimlarida qo'llaniladi. Ular yuqori xavfsizlik, barqarorlik va unumdorlik darajasiga ega.

S32. Operatsion tizimlarda taqsimlangan tizimlar

Operatsion tizimlarda taqsimlangan tizimlar (Distributed Systems) quyidagi asosiy xususiyatlarga ega:

1. **Tarmoqlash:**
 - Tizim komponentlari turli tarmoqlar va geograik joylashganlar.
 - Tarmoq orqali aloqa va ma'lumot almashinuvi amalga oshiriladi.
2. **Moslashuvchanlik:**
 - Yangi komponentlarni qo'shish, olib tashlash orqali tizimni moslashtirib borish imkoniyati.
 - Tarmoq o'zgarishlari va nosozliklar ta'sirini kamaytirish.
3. **Markaziy boshqaruv:**
 - Tizimni markaziy boshqarish va monitoringini amalga oshirish.
 - Resurslar va yuklamani muvozanatlash.
4. **Xavfsizlik:**
 - Tarmoqdagi ma'lumotlar almashinuvini himoya qilish.
 - Tarqatilgan tizim komponentlarini himoya qilish.
5. **Yuqori ishonchoqlik:**
 - Alohida komponentlar nosozliklarga uchraganda ham tizimning faoliyati davom etishi.
 - Tizim resurslarining redundantligi.

Taqsimlangan tizimlar asosan korporativ muhitlarda, bulut xizmatlarida, katta ma'lumotlar tizimlarida qo'llaniladi. Linux, Windows, macOS, Unix kabi operatsion tizimlar taqsimlangan tizimlarni qo'llab-quvvatlaydi.

Operatsion tizimlarda, ishlovchi kuchlarni boshqarishni osonlashtirish, xizmatlar va resurslarni birlashtirish uchun taqsimlangan tizimlar mavjud. Operatsion tizimlarda taqsimlangan tizimlar, ishlovchi kuchlarni boshqarish, xizmatlarni birlashtirish va modular tuzishda yordam beradi. Mikroservislarni boshqarish tizimlari, tarqalishga asoslangan tizimlar va pardazh tizimlari keng tarqalgan va kuchli operatsion tizimlar hisoblanadi, ular kompaniyalar va sohalar o'rtasida ko'p qatlamli va integratsiya loyihalarni osonlashtirishda yordam beradi.

S33. Operatsion tizimlarda parallel hisoblash tizimlari

Parallel hisoblash tizimlar (Parallel Computing Systems) operatsion tizimlar ichida quyidagi asosiy xususiyatlarga ega:

1. **Ko'p protsessorlilik:**

- Bir nechta protsessorlardan iborat bo'lgan tizimlar.
- Protsessorlar bajariladigan vazifani parallel tarzda bo'lib oladi.
- Bajarish tezligini sezilarli oshirishi mumkin.

2. **Klasterlar:**

- Bir-biriga umumiy tizim sifatida birlashtiriladigan kompyuterlar to'plami.
- Katta hajmli vazifalarni bajarish uchun qo'llaniladi.
- Tarmoq orqali aloqa o'rnatib, ishni parallel taqsimlaydi.

3. **Parallel algoritm va dasturlar:**

- Algoritmalar va dasturlar parallel bajarilishi uchun mo'ljallangan.
- Vazifani bir nechta protsessor yoki protsess orasida taqsimlashni ta'minlaydi.
- Parallel bajarilish tezligini oshiradi.

4. **Parallel operatsion tizimlar:**

- Linux, Windows, macOS kabi operatsion tizimlar parallel hisoblash imkoniyatlarini qo'llab-quvvatlaydi.
- Parallel protsessorlar, CUDA, OpenMP, MPI kabi texnologiyalarni qo'llaydi.

Parallel hisoblash tizimlari katta hajmli ma'lumotlar bilan ishlash, ilmiy hisoblar, ma'lumotlar tahlili va boshqa sohalar uchun juda foydali.

Operatsion tizimlarda taqsimlangan tizimlar, ishlovchi kuchlarni boshqarish, xizmatlarni birlashtirish va modular tuzishda yordam beradi. Mikrosvislarni boshqarish tizimlari, tarqalishga asoslangan tizimlar va pardazh tizimlari keng tarqalgan va kuchli operatsion tizimlar hisoblanadi, ular kompaniyalar va sohalar o'rtasida ko'p qatlamli va integratsiya loyihalarni osonlashtirishda yordam beradi.

S34.Operatsion tizimlarda masofadan resurslarni boshqarish

Operatsion tizimlarda masofadan resurslarni boshqarish quyidagi asosiy xususiyatlarga ega:

- **Masofadan kirish** - Foydalanuvchilar ishlash uchun zarur resurslarni (fayllar, ilovalar, hisoblash quvvati) masofadan kirib boshqarish imkoniyati. SSH, RDP, VNC kabi protokollar orqali masofadan ulanish.
- **Markaziy boshqaruv** - Barcha resurslar va foydalanuvchilar markaziy boshqaruv ostida. Resurslarni o'rnatish, yangilash, o'chirish, mahkamlash imkoniyati. Foydalanuvchilar va huquqlarni boshqarish.
- **Resurs taqsimoti** -Resurslarga bo'lgan talabni tahlil qilish va muvozanatlash. Yuklamani muvozanatlash va resurslardan samarali foydalanish.
- **Xavfsizlik va monitoring** - Foydalanuvchilar, resurslar va operatsiyalarni xavfsizlik nuqtayi nazaridan nazorat qilish. Resurslar va foydalanuvchilar faoliyatini kuzatish va tahlil qilish.
- **Masofadan boshqarish vositalari** - Windows Server, Linux, macOS kabi operatsion tizimlarda masofadan boshqarish vositalari mavjud. PowerShell, SSH, RDP, VNC, Web console kabi vositalar orqali boshqarish mumkin.

Masofadan resurslarni boshqarish operatsion tizimlarning asosiy imkoniyatlaridan biri bo'lib, korporativ muhitlarda, bulut xizmatlarida keng qo'llaniladi.

Operatsion tizimlarda masofadan resurslarni boshqarish, tizim administratorlari uchun juda muhim bir vazifadir. Bu, har qanday tarmoqaro muhitda qurilgan tizimlarda, masofadan joylashtirilgan kompyuterlarni, serverlarni, va boshqa tizim resurslarini boshqarishni ta'minlash uchun mo'ljallangan usullardan biridir. Quyidagi turli usullar masofadan resurslarni boshqarish uchun foydalaniladi:

1. Masofadan Qo'llanuvchilar Boshqarish (Remote User Management) - Bu usul, masofadan qo'llanuvchilar yoki tizim administratorlari uchun qurilgan masofadan boshqarish vositalaridan foydalanishni o'z ichiga oladi. Ushbu vositalar, tarmoqaro tizimlarga yo'naltirilgan qo'llanuvchilar, serverlar va tizim resurslarini masofadan boshqarishga imkon beradi. Masofadan qo'llanish vositalari, kompyuterlar, serverlar va boshqa tizim resurslarini masofadan boshqarishga imkon berish uchun SSH (Secure Shell), RDP (Remote Desktop Protocol), VNC (Virtual Network Computing), TeamViewer va boshqalar kabi protokollardan foydalanadi.

2. Masofadan Tizim Boshqarish (Remote System Administration) - Bu usul, masofadan tizim administratorlari uchun qurilgan boshqarish vositalari orqali tizimlar va tarmoqaro resurslarni boshqarishni ta'minlaydi. Ushbu vositalar, masofadan qo'llanuvchilar boshqarish vositalari bilan bir xil bo'lib, administratorlar tizim resurslarini boshqarish uchun xizmat ko'rsatadi. Masofadan tizim boshqarish

vositalari, tizimni boshqarish, yozish, sozlash va monitoring qilish imkoniyatlarini ta'minlaydi. Masofadan tizim boshqarish vositalari, SSH, PowerShell, WMI (Windows Management Instrumentation), Ansible, Puppet, Chef, SaltStack va boshqa protokollar va dasturlash tillari orqali foydalaniladi.

3. Otomatlashtirilgan Qo'llanish (Automated Administration) - Bu usul, tizim administratorlari uchun muhim vazifalarni otomatlashtirishga asoslanadi. Otomatlashtirilgan boshqarish vositalari yordamida, ma'lumotlar ko'chirish, sozlash, monitoring, o'rnatish va boshqa qiyinchiliklarni avtomatlashtirish mumkin. Otomatlashtirilgan qo'llanish vositalari, tizimni avtomatik ravishda boshqarish, vazifalarni otomatik ravishda ishga tushirish, tizimni otomatik ravishda sozlash va boshqa vazifalarni amalga oshirishda foydalaniladi.

4. Masofadan Kodni Boshqarish (Remote Scripting) - Bu usul, masofadan kodni yuborish orqali masofadan tizim resurslarini boshqarishni ta'minlaydi. Bu, avtomatlashtirilgan boshqarishni osonlashtirish uchun foydalaniladi. Masofadan kodni boshqarish vositalari, tarmoqaro tizimlarga yo'naltirilgan ma'lumotlar va avtomatlashtirilgan vazifalarni bajarish uchun foydalaniladi. Bu vositalar, kompyuterlar, serverlar va boshqa tizim resurslarini boshqarish uchun PowerShell, Bash, Python, Ruby va boshqa dasturlash tillaridan foydalanish imkonini ta'minlaydi.

S35. Operatsion tizimlarda xavfsizlik

Operatsion tizimlar Linux, Windows, macOS kabi zamonaviy platformalar turli xil xavfsizlik mexanizmlarini taqdim etadi.

Operatsion tizimlarda xavfsizlik, tarmoqaro tizimlarda ma'lumotlar va resurslarni himoya qilishni ta'minlaydi. Bu, tizimlarni noziklik va xavf o'rtasida yaxshi to'g'ri chiqarish, soxta kirishlarni oldini olish, ma'lumotlar to'g'risida himoya ta'minlash va tarmoqaro hamkorlar bilan ma'lumot almashishni xavfli bo'lmagan holda amalga oshirishni o'z ichiga oladi. Quyidagi tahlil qilinadigan muhim xavfsizlik tushunchalari va usullar mavjud:

1. Kirish/Chiqish Tizimlarini Himoya Qilish:

- **Parol Politikasi:** Tizimga kirish vaqti foydalanuvchilardan parollar so'raladi. Yaxshi parol politikasi, qulay va xavfsiz parollar yaratishni va ularni doimiy ravishda yangilashni ta'minlaydi.
- **Ikkinchi Darajali Tasdiqlash (Two-Factor Authentication, 2FA):** 2FA, foydalanuvchilarga birinchi darajali autentifikatsiya jarayonidan keyin ikkinchi darajali tasdiqlashni so'ragan bo'lishini ta'minlaydi. Masalan, parol kiritishdan so'ng, SMS orqali yoki ilova yordamida tasdiqlash kodi kiritilishi.
- **Biometrik Tasdiqlash:** Foydalanuvchi identifikatsiyasini uning biometrik ma'lumotlari orqali amalga oshirish. Misol uchun, retinalar, oyoqlar yoki qo'l izi.
- **Mavjud parol yoki kirish xavfsizlik jarayonlarini tekshirish:** Kirish va parol qo'llanuvchilarning ma'lumotlarini tekshirish va tez-tez parolni yangilashni talab qilish.

2. Ma'lumotlarni Xavfsiz Saqlash:

- **Ma'lumotlarni Shifrlash:** Hassas ma'lumotlarni shifrlash, ularni e'tiborsiz saqlash uchun yaxshi usul bo'lib hisoblanadi.
- **Regular Backup:** Ma'lumotlarni qo'llab-quvvatlash uchun mustahkam ro'yxatlarni, boshqa tizimlarni yoki bulut xizmatlarini qo'llab-quvvatlash.
- **Ma'lumotlarni Unutkicha O'chirish:** Ma'lumotlarni kerak emas bo'lgan joylardan unutkicha o'chirish, unutib qolgan ma'lumotlarni noldan saqlashni oldini olish.
- **Ma'lumotlarni To'g'risida Monitoring:** Ma'lumotlarni boshqarish va shifrlashni amalga oshirish, ma'lumotlarni to'g'risida monitoring qilish, shunday qilib, yoki aniq o'g'irishni aniqlash.

3. Xavfsizlikni Ta'minlash:

- **Anti-Virus va Anti-Malware Dasturlar:** Kompyuterlarga zararli dasturlar va viruslarga qarshi himoya ta'minlaydigan antivirus va anti-malware dasturlari o'rnatish.
- **Yangilashlar:** Operatsion tizim va dasturlarni so'nggi yangilashlari orqali xavfsizlikni ta'minlash. Bu yangilashlar, xavfsizlik yopiqliklarini va zararli dasturlarning yangi variantlariga qarshi himoya ta'minlaydi.
- **Firewall:** Tarmoqaro xavfsizlikni oshirish uchun tarmoqaro portlarni va ma'lumotlarni boshqarish uchun firewoll o'rnatish.
- **Xavfsizlik Auditlari:** Tizim xavfsizlik sozlamalarini va o'zgarishlarni tekshirish va monitoring qilish.
- **Tizim Monitoring:** Tizimni monitoring qilish va to'g'ri ishlashini ta'minlash. Bu, tizimda yuz beradigan xavfsizlik hodisalari va hujumlar to'g'risida ogohlantirishni ta'minlaydi.

4. Hamkorlik Xavfsizligi:

- **Virtuallik Hisoblash:** Resurslarni boshqarishda virtual tizimlar yordamida xavfsizlikni ta'minlash. Bu, virtual mashinlar yordamida resurslar va ma'lumotlarni boshqarishga imkon beradi, shuningdek, tizim xavfsizligini osh

S36. Operatsion tizimlarda umumiy ruhsat va foydalanuvchi huquqlarini sozlash

Operatsion tizimlarda foydalanuvchi huquqlari va ruhsatlarini sozlash quyidagi asosiy yondashuv va vositalarni o'z ichiga oladi:

1. **Foydalanuvchi turlari** - Administratorlar, standart foydalanuvchilar, cheklangan foydalanuvchilar. Har bir foydalanuvchi turi uchun o'ziga xos huquqlar va cheklovlar.
2. **Ruhsatlar** - Fayllar, direktoriyalar, qurilmalar, ilovalarga kirish uchun ruhsatlar. Qo'shish, o'zgartirish, o'chirish, bajarish huquqlari. Chmod, chown, icacs kabi vositalar orqali ruhsatlarni belgilash.
3. **Guruhlar** - Foydalanuvchilarni bir necha guruhlariga ajratish. Guruhlar uchun ruhsatlarni sozlash. Foydalanuvchilarni bir necha guruhga kiritish.
4. **Avtorizatsiya va autentifikatsiya** - Kirish uchun foydalanuvchi nomi va parol yoki boshqa usullar. Foydalanuvchilar uchun odam va roli asosida huquqlarni ajratish.
5. **Siyosat va qoidalar** - Xavfsizlik, ruhsatlar, foydalanuvchilar uchun siyosatlar. Tizim administratorlari tomonidan belgilanadi. Tizim etiborsizlik va qoidabuzarliklarga qarshi himoyani oshiradi.

Operatsion tizimlardagi ruhsatlar va foydalanuvchi huquqlari tizim xavfsizligini ta'minlash uchun muhim ahamiyatga ega.

Operatsion tizimlarda umumiy ruhsat va foydalanuvchi huquqlarini sozlash, foydalanuvchilarga ma'lumotlarga kirish, tizim resurslarini foydalanish, va boshqa operatsiyalarni bajarish imkonini ta'minlash uchun mo'ljallangan. Quyidagi usullar bilan bu huquqlar birlashtiriladi:

1. Group va Foydalanuvchi Profillarini Yaratish:

Grouplar: Tizimdagi foydalanuvchilarni kelgusi huquq va ruxsatlarni birlashtirish uchun guruhlar (Grouplar) yaratish. Masalan, "administratorlar", "foydalanuvchilar", "menejerlar" kabi guruhlar yaratiladi.

Profillar: Har bir guruh yoki foydalanuvchi uchun mahsus profillar yaratish. Bu profil o'z ichiga xususiyatlarni (masalan, parol politikasi, ma'lumotlarga kirish huquqlari, foydalanuvchi interfeysi sozlamalari) o'z ichiga oladi.

2. Ruhsat Birlashtirish:

Katalog va Fayllarga Kirish Ruhsati: Foydalanuvchilarga, belgilangan katalog va fayllarga kirish ruhsatini birlashtirish.

Foydalanuvchi Profillari bo'yicha Foydalanuvchi Kirish Huquqlari: Har bir foydalanuvchi profiliga mos keladigan kirish huquqlarini sozlash. Masalan, "administrator" profili yoki "foydalanuvchi" profili.

Foydalanuvchi Huquqlari: Har bir foydalanuvchiga kerakli foydalanuvchi huquqlarini berish. Misol uchun, fayl ochish, yozish, ochish, ochish, va boshqa tizim resurslariga kirish.

Qurilish Sozlamalari: Foydalanuvchilarga kerakli qurilish sozlamalari (masalan, qo'shimcha vositalar, tizim parametrlari va boshqalar) birlashtirish.

3. Xavfsizlik:

Kirish va Parol Politikasi: Kerakli kirish va parol politikasini sozlash. Masalan, minimal uzunlik, katta va kichik harflar, sonlar, va maxfiy belgilar talab qilish.

Kerakli Amal: Kerakli amallarni bajarish uchun foydalanuvchilar huquqlarini ta'minlash. Masalan, faqat kerakli fayllarni o'chirish huquqlari, tizimni boshqarish, va boshqa amallar.

Monitoring va Logging: Foydalanuvchilar faoliyatini monitoring qilish va foydalanuvchi faoliyati tarixini saqlash. Bu, tizimni hujumlar va qo'shimcha yo'l o'rnatishlariga qarshi himoya qilishga yordam beradi.

4. To'g'riqqa Olish:

Huquqni Boshqarish: Foydalanuvchilar to'g'ri boshqarish, yani, ularning kerakli huquqlar va ruxsatlari bo'yicha monitoring qilish.

Foydalanuvchi Foydalanish Tarixini Tahlil Qilish: Foydalanuvchi faoliyatini tahlil qilish va qo'shimcha jarayonlarga qarshi ko'rsatishlar chiqarish.

Foydalanuvchilarning Foydalanish Huquqlarini Yopish: Foydalanuvchi hisobidan foydalanuvchilar to'g'risidagi ruxsatlarini yopish. Masalan, ishdan bo'shatilgan bo'lishi yoki yo'qolish sababli.

5. Avtomatlashtirilgan Huquqlar:

Avtomatik Amal: Foydalanuvchilarga avtomatik ravishda huquqlar berish. Masalan, yangi foydalanuvchilarga birinchi darajali ruxsat berish va keyin qo'shimcha huquqlarni o'rnatish.

Operatsion tizimlarda umumiy ruhsat va foydalanuvchi huquqlarini sozlash, foydalanuvchilarga ma'lumotlarga kirish, tizim resurslarini foydalanish, va boshqa operatsiyalarni bajarish imkonini ta'minlaydi. Bu huquqlar tizim administratorlari tomonidan sozlash, boshqarish va monitor qilish uchun muhimdir, bu hamda tizimning xavfsizlik va qo'llanishni amalga oshirishda yordam beradi.

S37. Operatsion tizimlarda samaradorlik monitoring

Operatsion tizimlarning samaradorligini monitoring qilish quyidagi asosiy yo'nalishlarda amalga oshiriladi:

1. Tizim resurslarini monitoring qilish:

Protsessor, xotira, disk, tarmoq resurslaridan foydalanish.
Resurs yuklamasi, tiqilishlar, limitsizlanish holatlarini aniqlash.
Ilovalar, xizmatlar, jarayonlar tomonidan resurslardan foydalanish.

2. Tizim faoliyatini monitoring qilish:

Tizim logiklarini kuzatish, xatolar va ogohlantirish xabarlarini.
Disk, tizim, tarmoq yadrosu, foydalanuvchilar faoliyatini kuzatish.
Nosozlik va uzilishlar haqida ogohlantirishlar berish.

3. Ishlov berish qobiliyatini baholash:

Tizim qo'zg'alishini, javob vaqtlarini, qabul qilingan talablarni o'rganish.
Talablar bajarilish tezligini, muddatini, unumdorlikni baholash.

4. Xavfsizlik va moslashuvchanlikni monitoring qilish:

Xavfsizlik qoidalarining buzilib qolishi, kirish urinishlarini aniqlash.
Tizimning yangi konfiguratsiyalarga moslashuvchanligini baholash.

5. Tahlil va hisobotlar:

Tizim faoliyati, resurslardan foydalanish, muammolar haqida hisobotlar.
Olingan ma'lumotlarni tahlil qilish, natijalar asosida chora-tadbirlar ishlab chiqish.

Operatsion tizimlarning samaradorligini monitoring qilish uchun PowerShell, Linux utililiklari, monitoring dasturlari, xizmatlar kabi vositalardan foydalanish mumkin.

S38. Operatsion tizimlarda xizmatchi ilova dasturlari

Ba'zi mashhur xizmatchi dasturlar: Windows PowerShell, Linux systemd, macOS Launchd, Nagios, Zabbix, Prometheus.

Xizmatchi dasturlar tizim faoliyatini nazorat qilish, xavfsizlikni ta'minlash, foydalanuvchilarga yordam berish, texnik xizmat ko'rsatish vazifalarini bajaradi.

Operatsion tizimlarda xizmatchi ilova dasturlari, tizimni boshqarish, monitor qilish va samaradorlikni oshirish maqsadida foydalanuvchilarga yordam beradi. Bu ilovalar tizim administratorlarining tizimni boshqarish, ma'lumotlarni nazorat qilish, xavfsizlikni ta'minlash, xato yoki nosozlikni aniqlash, va boshqa kuchli amallarni bajarishga yordam beradi. Quyidagi xizmatchi ilova dasturlar samaradorlik monitoringida foydalaniladi:

Nagios, tarmoqaro tizimlarni monitoring qilish uchun kuchli va mashhur bir ilovadir. U, serverlarni, tarmoqlarni, protsesslarni va boshqa tizim resurslarini monitoring qilish, xabarlar yuborish va muammoni aniqlash uchun oson, ishonchli va samaradorlikni oshirishiga yordam beradi. Nagios, xavfsizlik, ishlab chiqarish, va sozlashlarni monitoring qilish uchun plaginlar bilan keng imkoniyatlar ta'minlaydi.

Zabbix, IT infrastrukturining monitoringi uchun kuchli, keng xil funksiyalar bilan yaratilgan bepul va ochiq manbalarga ega bir ilovadir. U, ma'lumotlarni to'plab, nazorat qilish, va statistik ma'lumotlar yig'ish uchun kerakli vositalarni taqdim etadi. Zabbix, real vaqt, kompleks qurilishlarda ishlaydi va o'z ichiga olgan veb interfeysi orqali monitoring statistikalarini kuzatib borish imkoniyatiga ega.

SolarWinds, IT infrastrukturining monitoringi va boshqarishini ta'minlovchi bir qator ilovalar to'plami. U, tarmoqaro tizimlar, serverlar, ulanishlar, tarmoqaro qurilmalar, ma'lumotlar qog'ozlari va boshqa tizim resurslarini monitoring qilish uchun imkon beradi. SolarWinds, o'zaro uzviyatli ma'lumotlar (Correlation Engine) tahlil qilish, xato xabarnomalar yuborish, va IT infrastrukturining samaradorligini oshirish uchun qo'llaniladi.

Prometheus, oson va samaradorlik monitoringi va metrics yig'ilishi uchun ozbek ma'lumotlar tahlil tizimi hisoblanadi. U, serverlarni, tarmoqaro qurilmalar, va o'zaro aloqalar bilan monitoring qiladi. Prometheus, grafiklarga, xatoliklarni aniqlash va serverlarni avtomatik ravishda qo'llab-quvvatlash uchun qo'llaniladigan bog'lovchi monitoring dasturi (alerting) bilan integratsiya qilinadi.

Splunk, tizim ma'lumotlarini, tarmoqaro qurilmalar, ilova loglarini va boshqa ma'lumotlarni monitoring qilish, qayta ishlash va analiz qilish uchun kuchli va mashhur bir ilovadir. Splunk, qulay va real vaqt monitoring, muammoni aniqlash, yozuvchi analiz va yordamchi yozuvchi dasturlar orqali monitoringni oshiradi.

Grafana, metrikalarni vizualizatsiya qilish uchun moslashuvchan dastur hisoblanadi. U, muddatli, soatlik, haftalik va sozlashlarni nazorat qilish uchun grafiklarga aylanishi mumkin. Grafana, ulkan qo'llanuvchilar tomonidan qo'llaniladigan, masofaviy monitoring va ma'lumotlarni visualizatsiya qilishni osonlashtiradi.

Operatsion tizimlarda xizmatchi ilova dasturlari, tarmoqaro tizimlar, serverlar, qurilmalar va boshqa tizim resurslarini monitoring qilish, xato va nosozliklarni aniqlash, va samaradorlikni oshirishga yordam beradi. Bu ilovalar tizim administratorlari uchun samarali va qulay vositalardir, ularga tizimni kuzatish, nazorat qilish va boshqarishda yordam beradi.

S39. Linux operatsion tizimi terminali

Linux operatsion tizimida terminal muhim vositalardan biri hisoblanadi. Terminal quyidagi asosiy vazifalarni bajaradi:

1. **Buyruqlarni ijro etish** - Tizim buyruqlari, skriptlar, ilovalarni bajarish. Tizim sozlamalarini o'zgartirish, boshqarish. Foydalanuvchi, jarayon, xizmatlar, fayllar bilan ishlash.
2. **Fayllar va direktoriyalar bilan ishlash** - Fayl va direktoriyalarni ko'rish, yaratish, o'zgartirish, o'chirish. Fayl tizimida navigatsiya qilish. Fayllar va direktoriyalar ustida operatsiyalarni amalga oshirish.
3. **Tizim ma'lumotlarini olish** - Tizim resurslarini, jarayonlarni, xizmatlarni kuzatish. Tizim loglarini ko'rish va tahlil qilish. Tizim ma'lumotlari va holatini tahlil qilish.
4. **Tizimni boshqarish**- Tizim konfiguratsiyalarini o'zgartirish. Xizmatlar, jarayonlar, foydalanuvchilarni boshqarish. Tizimni yangilash, ta'mirlash, o'rnatish.
5. **Skriptlar yozish**- Tizimda bajarilishi uchun skriptlar yozish. Skriptlarni avtomatlashtirilgan holda ishlatish. Tizimni samarali boshqarish uchun skriptlardan foydalanish.

Terminal Linux tizimining asosiy vositalaridan biri bo'lib, u tizimni boshqarish, nazorat qilish, tuzatish va moslashtirishda qo'l keladi. Terminaldan asosiy buyruqlar, skriptlar, dasturiy ta'minot vositalari orqali foydalanish mumkin.

Linux operatsion tizimining terminali, grafik interfeys (GUI) orqali emas, balki matnli interfeys (CLI) yordamida ishlovchi dastur hisoblanadi. Linux terminali foydalanuvchilarga tizimni boshqarish, boshqa dasturlarni ishga tushirish, fayllarni boshqarish, ma'lumotlarni ko'rish, tarmoqaro ulanishlar va boshqalar kabi amallarni amalga oshirish imkoniyatini beradi.

S40. MS Windows operatsion tizimida ota-ona nazorati xizmati

Ota-ona nazorati xizmati orqali ota-onalar bolalarning kompyuter va internet foydalanishini nazorat qilish, ularning xavfsizligini ta'minlash, sog'lom va yetarli foydalanishini ta'minlash imkoniga ega bo'ladilar.

MS Windows operatsion tizimida ota-ona nazorati xizmati (Parental Control Service), o'quvchi yoki yosh foydalanuvchilarni tizimda nazorat qilish, ular uchun ma'lumotlar va amal limitlarini sozlash, uyga yaqin nazorat tizimlarini ta'minlash va qo'llanish sohalarini cheklash uchun mo'ljallangan imkoniyatlarni ta'minlaydi. Bu xizmat o'quv markazlari, oila markazlari, institutlar yoki hukumat tashkilotlari kabi muassasalar uchun juda muhim bo'lishi mumkin. Quyidagi asosiy funktsiyalar MS Windows operatsion tizimida ota-ona nazorati xizmati orqali amalga oshiriladi:

1. O'quvchi va Yosh Foydalanuvchilarni Ta'qiqlash:

O'qish va ishlab chiqarish vaqti cheklovlarini sozlash: O'quvchi yoki yosh foydalanuvchilarni tizimda necha soat ishlashi mumkinligini cheklash.

Tizimda ishlatiladigan dasturlarni cheklash: Belgilangan dasturlarni (masalan, o'yinlar, 18+ dasturlar, onlayn resurslar) ishlatishga ruxsat berish yoki cheklash.

Veb saytlarni bloklash: Foydalanuvchilarni ma'lum veb saytlardan cheklash uchun filtrlar qo'yish.

2. Tizimda Bo'lgan Amallarni Cheklash:

Foydalanuvchi aktivligini nazorat qilish: Qaysi dasturlar va veb-saytlar ishlatilganligini va qanday vaqtda ishlatilganligini kuzatish.

Foydalanuvchi bilan bog'lanish vaqtlarini cheklash: Kechirasizlik vaqtlarini cheklash va bog'lanishlarni nazorat qilish.

3. Ma'lumotlar va Amallarni Cheklash:

Ma'lumotlarga kirishni cheklash: Foydalanuvchilarni mahalliy ma'lumotlar va tizim sozlamalari qismlariga kirishni cheklash.

Fayllarga kirishni cheklash: Belgilangan fayllar yoki kataloglarga kirishni cheklash.

4. Internet Fayllarini Cheklash:

Internet fayllarini cheklash: Foydalanuvchilarning internetga kirib veb-saytlarni ko'rish imkoniyatini cheklash.

Internet fayllarini bloklash: Belgilangan internet saytlarini yoki veb-sayt turlarini bloklash.

5. Vaqti bilan Cheklash:

Cheklash vaqtlarini sozlash: Foydalanuvchilar uchun kompyuterda haroratni belgilangan vaqtdan so'ng yopish va qanday kun vaqtni ishlatishlarini cheklash.

Foydalanuvchi aktivligini soatlar bo'yicha cheklash: Foydalanuvchi faoliyatini, ishlatish vaqtlarini, foydalanuvchi aktivligini soatlar bo'yicha nazorat qilish.

6. Monitoring va Xabarnomalar:

Monitoring: Foydalanuvchilarning faoliyatini kuzatish va yozish.

Xabarnomalar: O'quvchi yoki yosh foydalanuvchilarning faoliyatiga oid xabarnomalar yuborish.

7. Foydalanuvchi Profili va Cheklash Sozlamalari:

Foydalanuvchi Profili: Har bir foydalanuvchi uchun o'z hisobini yaratish va maxsus sozlamalarini sozlash.

Cheklash sozlamalari: Ota-ona nazorati xizmati orqali foydalanuvchilarni cheklash uchun maxsus sozlamalarni sozlash.

Bu funktsiyalar ota-ona nazorati xizmatlarida chet eldagi tizimlar uchun ham umumiy bo'lib, o'quvchilarni va yosh foydalanuvchilarni tizimda ta'lim olish, ma'lumotlarni himoya qilish va ularni onlayn mahsulotlar va tizimlar bilan ta'minlash jarayonida foydalanuvchilarni nazorat qilishga yordam beradi.

S41. O'rnatilgan operatsion tizimlar

O'rnatilgan operatsion tizimlar - bu mahsulotlardagi tayyor o'rnatilgan operatsion tizimlar hisoblanadi. Ular quyidagi maqsadlarda ishlatiladi:

1. Mahsulotlar:

Kompyuter, noutbuk, planshet, smartfon kabi asboblari.

Televizor, oyoq kiyimlar, aqlli uylar, avtomobillar kabi mahsulotlar.

2. Sanoat va texnologiyalar:

Robotlar, CNC mashinalar, smartfabrikalardagi tizimlar.

Medikal asbob-uskunalar, aerokosmik tizimlar.

Elektr uzatish tizimlari, kommunal xizmatlar.

3. Ixtisoslashgan qurilmalar:

Terminal qurilmalari, bank terminallari.

ATM, Bank kassa apparatlari.

Pul-kredit terminallari, xisoblagichlar.

O'rnatilgan operatsion tizimlar quyidagi afzalliklarga ega:

1. Mahsulotlar uchun moslashtirilgan va optimallashtirilgan.
2. Foydalanish uchun sodda va oddiy, kamida funktsionalligi bor.
3. Taqdim etilgan barcha xizmatlar va funktsiyalar ishlaydi.
4. Xavfsizlik, barqarorlik, ta'mirlash masalalari yaxshi sozlangan.
5. Qurilma va tizimlar uchun maxsus moslashtirilgan.

Mashhur o'rnatilgan operatsion tizimlar: Windows Embedded, Android, iOS, Linux-based tizimlar, VxWorks, QNX, FreeRTOS va boshqalar.

Ular turli xil mahsulotlar, qurilmalar, texnologik tizimlarni boshqarish uchun ishlatiladi.

O'rnatilgan operatsion tizimlar, bilgisayarning diskiga qo'yilgan va uni ishga tushirilgan tizimdir. Bu tizimlar o'rnatilgan diskga ma'lumotlarni saqlaydi va bilgisayarning ishlashini boshqaradi. Quyidagi turdagi mashhur operatsion tizimlar mavjud:

Microsoft Windows, dunyoda eng keng tarqalgan operatsion tizimlaridan biridir. Windows, ajoyib grafik interfeysi va oson foydalanishga ega bo'lib, ayni vaqtda bir qancha variantlari mavjud: Windows 10, Windows 8, Windows 7, Windows Vista, Windows XP kabi.

macOS Apple tomonidan ishlab chiqilgan operatsion tizimdir. macOS, Apple kompyuterlari (Macintosh) uchun mo'ljallangan va iOS (iPhone va iPad) tizimi bilan mos ravishda ishlaydi.

Linux, ochiq manbali, bepul va kuchli operatsion tizimdir. Linux ommaviy ko'pgina versiyalari va turli variantlari mavjud, masalan, Ubuntu, Debian, Fedora, CentOS, openSUSE kabi.

Ubuntu, Linux distributivi, jamiyat tomonidan eng ko'p qo'llaniladigan va o'rnatilgan variantlaridan biridir. U bepul, ochiq manbali va ishlab chiqarish jarayonida qizg'in qo'llanuvchilarning ehtiyojlarini qondiradi.

Fedora, Red Hat tomonidan ishlab chiqilgan va o'rganuvchilar, ishlab chiqaruvchilar va boshqa kuchlarga qaratilgan ochiq manbali Linux distributividir. Fedora, eng so'nggi xususiyatlar bilan yuklanadi va yangilanadi.

CentOS, Red Hat Enterprise Linux (RHEL) distributivining boshqa ismi hisoblanadi. Uning asosiy maqsadi, korporativ xususiyatlarga ega Linux operatsion tizimini bepul taqdim etishdir.

Android, Google tomonidan ishlab chiqilgan va keng qo'llanilgan mobil operatsion tizimdir. Uning qo'shimcha tizimlaridan biri ham **Android xizmatlar** hisoblanadi.

iOS, Apple tomonidan ishlab chiqilgan va ishlatilgan mobil operatsion tizimdir. U faqatgina iPhone, iPad va iPod Touch qurilmalari uchun mo'ljallangan.

Chrome OS, Google tomonidan ishlab chiqilgan va xususan Chromebook uchun mo'ljallangan operatsion tizimdir. U ommaviy ko'p ishlatiladigan veb-saytlar, veb-alloqalar va veb-servislar orqali ishlaydi.

FreeBSD, Unix bilan birga ishlovchi, ommaviy tomonlama ochiq manbali operatsion tizimdir. U eng keng qo'llaniladigan serverlar, hibrid, va manzaralar tizimlaridan biridir.

Bu, faqat qiziqarli operatsion tizimlar deyilgan. Boshqa ko'plab variantlar va distributivlar mavjud bo'lsa-da, ular o'rnatilgan tizimlar hisoblanadi. Har biri, o'z xususiyatlari va maqsadlari bo'yicha foydalanuvchilar uchun mos bo'lgan tuzilmalarga ega.

S42. Operatsion tizimlarda ma'lumotlarni kiritish-chiqarishni boshqarish

Operatsion tizimlar ma'lumotlarni kiritish va chiqarishni boshqarish uchun quyidagi asosiy funksiyalarni bajaradi:

1. Kiritish-chiqarish qurilmalari bilan ishlash:

Klaviatura, sichqoncha, monitor, printer kabi qurilmalar bilan muloqot qilish.

Qurilmalarni aniqlash, ularni haydovchilar yordamida boshqarish.

Qurilmalar orqali ma'lumotlar uzatilishini ta'minlash.

2. Fayllar va ma'lumotlar bilan ishlash:

Ma'lumotlarni fayllarga yozish, fayllardan o'qish.

Fayllar sistemasini boshqarish, fayl va direktoriyalarga ruxsatlarni boshqarish.

Fayllar va ma'lumotlar saqlanishini ta'minlash.

3. Tarmoq interfeyslari bilan ishlash:

Tarmoq qurilmalari bilan aloqa o'rnatish va ularga ruxsatlarni boshqarish.

Tarmoq protokollari yordamida ma'lumotlarni uzatish.

Tarmoq xavfsizligini ta'minlash.

4. Ma'lumotlarni boshqarish va optimizatsiya qilish:

Ma'lumotlarni ularning kerakli ko'rinishga o'zgartirish.

Ma'lumotlar oqimini optimallashtirish va samaradorlikni oshirish.

Ma'lumotlar saqlanishini boshqarish, nusxalash va qayta tiklash.

5. Foydalanuvchi interfeysi bilan ishlash:

Foydalanuvchi interfeysini yaratish, sozlash va boshqarish.

Oynalar, menyu, tugmalar, dialoglar yaratish.

Foydalanuvchining kiritishlarini qabul qilish va qayta ishlash.

Operatsion tizimlar bu funksiyalarni bajarish orqali ma'lumotlarni kiritish, qayta ishlash, uzatish, saqlash va chiqarish jarayonlarini boshqaradi. Bu esa qurilmalar, foydalanuvchilar va tizim o'rtasidagi o'zaro muloqotni ta'minlaydi.

Operatsion tizimlarda ma'lumotlarni kiritish-chiqarishni boshqarish uchun quyidagi muhim asosiy vositalar va usullar mavjud:

1. Fayllar va Kataloglar:

Fayllarni yaratish, o'qish, yozish va o'chirish: Fayllarni yaratish, fayllarga ma'lumotlarni yozish, fayllarni o'qish va o'chirish uchun standart komandalarni

(masalan, mkdir, touch, cat, rm kabi) yoki fayl-boshqaruv dasturlarini (masalan, File Explorer Windows da yoki Finder macOS da) ishlatish mumkin.

Kataloglarda bo'lgan fayllarni ko'rish: Fayllarni kataloglarda, kataloglar hirarkiyasida ko'rish uchun standart komandalardan foydalanish mumkin (masalan, ls Linux va macOS da, dir Windows da).

2. Fayl Turlari:

Matn fayllari: Matnli ma'lumotlar, kodlar, yozuvlar, maqolalar va boshqa matnli ma'lumotlar uchun ishlatiladi. Masalan, .txt, .doc, .docx, .pdf kabi.

Rasm fayllari: Rasm, diagramma, logotip, tasvirlar va boshqa grafik elementlarni saqlash uchun ishlatiladi. Masalan, .jpg, .png, .gif, .bmp kabi.

Audio fayllari: Audio ma'lumotlarni saqlash uchun ishlatiladi. Masalan, .mp3, .wav, .flac kabi.

Video fayllari: Video ma'lumotlarni saqlash uchun ishlatiladi. Masalan, .mp4, .avi, .mkv, .mov kabi.

3. Dasturlar va Aplikatsiyalar:

Mundarija qilish dasturlari: Ma'lumotlar tahrirlash, formatlash va saqlash uchun dasturlar. Masalan, Microsoft Word, LibreOffice Writer, Google Docs kabi.

Jadvallar va Hisobotlar: Ma'lumotlarni tuzatish, hisobot tuzish va ko'rib chiqish uchun dasturlar. Masalan, Microsoft Excel, Google Sheets kabi.

Tahrirlovchi dasturlar: Rasm va video muharrirlari, grafik dasturlari va boshqa tahrirlovchi vositalar. Masalan, Adobe Photoshop, GIMP, Adobe Premiere Pro kabi.

4. Tarmoq Ulashish:

Internet brauzerlari: Internetda ma'lumotlar izlash, maqolalar o'qish, elektron pochta yuborish va boshqa internetdagi faoliyatlar uchun ishlatiladi. Masalan, Google Chrome, Mozilla Firefox, Microsoft Edge kabi.

Elektron pochta dasturlari: Elektron pochta qabul qilish, jo'natish va ma'lumot almashish uchun ishlatiladi. Masalan, Microsoft Outlook, Gmail, Apple Mail kabi.

Fayl almashish va ulanish: Fayllarni boshqa kompyuterlarga yoki serverlarga yuborish va ulanish uchun fayl almashish protokollari va vositalar. Masalan, FTP, SSH, SMB/CIFS kabi.

5. Tizim O'zgaruvchilari:

Operatsion tizim parametrlari: Tizim sozlamalari, o'zgaruvchilari va konfiguratsiyalari o'zgartirish uchun ishlatiladi. Masalan, tizim bilan bog'liq ma'lumotlar (registry Windows da, plist fayllari macOS da, konfiguratsiya fayllari Linux da).

6. Ma'lumotlar Bazalari va Serverlar:

Ma'lumotlar bazalari (MB): Ma'lumotlar bazasini yaratish, ma'lumotlarni saqlash, ulanish va tahrirlash uchun ishlatiladi. Masalan, MySQL, PostgreSQL, Microsoft SQL Server kabi.

Serverlar: Ma'lumotlar, fayllar va ilovalarga qo'llab-quvvat berish uchun serverlar. Masalan, Web server (Apache, Nginx), File server, Mail server kabi.

7. San'atida qo'llaniladigan tarmoq protokollari:

- **HTTP/HTTPS:** Veb saytlariga bog'lanish va ma'lumot almashish.
- **FTP/SFTP:** Fayllarni o'qish va yozish uchun fayl almashish protokollar.
- **SMTP/POP/IMAP:** Elektron pochta almashish uchun protokollar.
- **TCP/IP:** Internetda ma'lumotlar almashish va ulanish uchun protokollar.

Bu, operatsion tizimlarda ma'lumotlarni kiritish-chiqarishni boshqarish uchun eng ko'p ishlatiladigan asosiy vositalar va usullarning faqat bir qismi. Har bir tizim ma'lumotlarni boshqarish uchun o'ziga xos dasturlar va o'zgaruvchilar ketma-ketligini taqdim etadi.

S43.Multidasturlash va multiprotsessorlash

Multidasturlash:

Multidasturlash - bu operatsion tizimning bir vaqtda bir nechta dasturlarni bajarish qobiliyatidir. Bu quyidagilarni anglatadi:

1. Resurslarga ega bo'lish:

- Bir vaqtda bir necha dastur protsessor, xotira, kiritish/chiqarish qurilmalari kabi tizim resurslaridan foydalanadi.
- Resurslarga ruxsatlar va prioritetlar asosida ega bo'lish.

2. Ochib tura olish:

- Bir dastur to'xtatilganda, boshqa dastur protsessorni egallaydi.
- Operatsion tizim dasturlar o'rtasida muvofiqlashtirilgan almashish amalga oshiradi.

3. Foydalanuvchi ishlari:

- Foydalanuvchi bir nechta dasturlarni bir vaqtda ishga tushirishi mumkin.
- Har bir dastur o'z oynasi va interfeysiga ega bo'ladi.

Multiprotsessorlash:

Multiprotsessorlash - bu bir nechta mustaqil protsessorlardan iborat tizimlar yordamida bir vaqtda bir necha vazifalarni bajarish qobiliyatidir. Bu quyidagilarni anglatadi:

1. Parallel ishlov berish:

- Bir vazifani bir necha protsessorlar bo'lib olib bajarishi.
- Har bir protsessor o'z ulushini bajaradi.

2. Katta hajmdagi ma'lumotlarga ishlov berish:

- Katta hajmdagi hisoblash jarayonlarini parallellashtirish.
- Protsessorlar soni ortishi bilan samaradorlik oshadi.

3. Yuqori ishonchlilik:

- Bir protsessor ishdan chiqsa, qolgan protsessorlar davom ettirib ishlayveradi.
- Tizimning mavjudligi va barqarorligi oshadi.

Multidasturlash va multiprotsessorlash operatsion tizimlar tomonidan qo'llab-quvvatlanadi. Bu tizimning samaradorligini va resurslardan foydalanish imkoniyatlarini oshiradi.

Multidasturlash va multiprotsessorlash, bir vaqtning o'zida bir nechta dasturlar yoki protsesslarni boshqarishni anglatadi.

S44.Operatsion tizimlarning jarayonlari boshqaruvi

Operatsion tizimlar jarayonlarni boshqarish uchun quyidagi asosiy funksiyalarni bajaradi:

1. Jarayonlarni yaratish va boshqarish:

- Yangi jarayonlarni yaratish va ularga resurslar ajratish.
- Jarayonlarni bloklash, to'xtatish, davom ettirish va tugatish.
- Jarayonlar o'rtasida o'zaro aloqani ta'minlash.

2. Protsessor vaqtini tarqatish:

- Jarayonlarni protsessor vaqti bo'yicha qanday taqsimlash.
- Har bir jarayonga ajratiladigan protsessor vaqtini boshqarish.
- Protsessor vaqtdan samarali foydalanish.

3. Xotira boshqaruvi:

- Har bir jarayonga xotira maydonlarini ajratish.
- Xotira ajratishda prioritetlar va cheklashlarni qo'llash.
- Xotira resurslaridan maqbul foydalanishni ta'minlash.

4. Kirish/chiqish jarayonlarini boshqarish:

- Jarayonlarning kiritish/chiqarish talablarini bajarish.
- Prioritetlarni belgilash va ularga muvofiq amal qilish.
- Kirish/chiqish jarayonlarining samaradorligini oshirish.

5. Oraliq va yakuniy holatlarni boshqarish:

- Jarayonlarning boshlash, to'xtatish, davom ettirish holatlari.
- Jarayonlar o'rtasidagi o'zaro munosabatlar.
- Jarayonlar tugatilishida yakuniy holatlarni qayd etish.

Operatsion tizim bu funksiyalar orqali jarayonlarni dinamik boshqaradi, resurslardan oqilona foydalanadi va jarayonlarning bajarilishini ta'minlaydi. Bu esa tizimning umumiy samaradorligini oshiradi.

Operatsion tizimlarning jarayonlari boshqaruvi, kompyuter tizimining barcha jarayonlarini nazorat qilish va boshqarishni ta'minlashga oid boshqaruv tizimi va vositasi haqidir. Bu, tizim ishini muvaffaqiyatli boshqarish, resurslarni boshqarish, foydalanuvchilar va dasturlarni boshqarish, xavfsizlikni ta'minlash va amalni monitoring qilishda juda muhimdir.

Operatsion Tizimlarning Jarayonlari Boshqaruvi Asosiy Vazifalari:

- 1. Resurslarni boshqarish:** Tizim resurslarini (protsessor, xotira, disk, tarmoq bog'liqlari kabi) boshqarish, ularni boshqarish va ulanishda moslashtirish. Bu, resurslarning qanday vaqtning qanday dastur tomonidan ishlatilishi kerakligini belgilash, taqsimlash va monitoringini o'z ichiga oladi.
- 2. Dasturlarni boshqarish:** Ishlayotgan dasturlarni boshqarish va ulanishda ustunlik tartibini aniqlash, dastur faoliyatini nazorat qilish, va qanday vaqtning qanday dastur tomonidan qanday ishlatilishi kerakligini belgilash.

3. **Protsesslarni boshqarish:** Tizimdagi har bir protsessni, ularning faoliyatlarini nazorat qilish, ularni qo'zg'atish, to'xtatish yoki o'chirish, ularga resurslarni ajratish va distributivlashtirish.
4. **Fayl tizimlarini boshqarish:** Fayllarni yaratish, o'qish, yozish, o'chirish va ko'chirish, fayl tizimlarini nazorat qilish, fayllarni qo'shish, fayllar ustida operatsiyalarni bajarish, fayllarni taqsimlash va ulash.
5. **Tarmoq xavfsizligi va boshqaruvi:** Tarmoq xavfsizligini ta'minlash, tarmoq resurslarini nazorat qilish, tarmoq xavfsizligi uchun sozlamalar va atrof-muhitlarni boshqarish.
6. **Xavfsizlik boshqaruvi:** Tizimni xavfsizlik holatida saqlash, ularga viruslar, kiber hamkorlar va maxfiy ma'lumotlarga hujumlar qarshi himoya qilish.
7. **Dastlabki yuklash jarayonini boshqarish:** Tizimning o'qilishini yuklash, tizimni faol holatga keltirish, yuklash qat'iyotini yoki muvaffaqiyatini tekshirish, tizimni boshqarish va to'xtatish.
8. **Monitoring va xabarlar:** Tizim faoliyatini nazorat qilish, xatoliklarni aniqlash, ish rejimini nazorat qilish va foydalanuvchilarga yordam berish uchun xabarnomalar yuborish.
9. **Amalni tartibga solish:** Har bir amalni, qanday vaqtning qanday dastur tomonidan qanday ishlatilishi kerakligini aniqlash va moslashtirish.

Operatsion Tizimlarning Jarayonlari Boshqaruvi Asosiy Vositalari:

1. **Jarayon boshqaruvi dasturlari:** Kompyuter tizimini boshqarish uchun xususiy jarayon boshqaruvi dasturlari mavjud. Masalan, Task Manager (Windows), Activity Monitor (macOS), top va htop (Linux) kabi.
2. **Qurilma biosi:** Kompyuter qurilmalarida, BIOS (Basic Input Output System) yoki UEFI (Unified Extensible Firmware Interface) kabi bioslar o'rniga, protsessor, xotira va boshqa qurilma komponentlarini sozlash imkoniyatini ta'minlaydi.
3. **Tarmoq boshqaruvi vositalari:** Tarmoq boshqaruvi tizimlari (masalan, Cisco Systems, Juniper Networks) tarmoq xavfsizligi, resurslarni boshqarish, bant aloqalarni monitor qilish va boshqarishga yordam beradi.
4. **Xavfsizlik vositalari:** Antivirus dasturlari, firewal, intrusion detection systems (IDS), intrusion prevention systems (IPS), security information and event management (SIEM) tizimlar, parol boshqaruv dasturlari va boshqalar kabi xavfsizlik vositalari kompyuter tizimining xavfsizligini ta'minlash uchun foydalaniladi.

Operatsion tizimlarning jarayonlari boshqaruvi, kompyuter tizimining ishlashini nazorat qilish, muvaffaqiyatli boshqarish va boshqarish imkonini ta'minlash uchun juda muhimdir. Bu boshqaruv tizimi, tizimni ishga tushirish, o'rnatish va b

S45.Operatsion tizimlarning jarayonlari boshqaruvini rejalashtirish parametrlari, rivojlantirish ko'rsatkichlari

Jarayonlar boshqaruvini rejalashtirish parametrlari:

1. Jarayonlarni yaratish va boshqarish:

- Yangi jarayonlarni yaratish uchun qoidalar
- Jarayonlarni bloklash, to'xtatish, davom ettirish va tugatish mezonlari
- Jarayonlar o'rtasidagi o'zaro aloqa qoidalari

2. Protsessor vaqtini tarqatish:

- Protsessor vaqtini taqsimlash algoritmlar
- Har bir jarayonga ajratiladigan protsessor vaqtining miqdori
- Protsessor vaqtidan samarali foydalanish strategiyalari

3. Xotira boshqaruvi:

- Xotira ajratish va erkinlashtirish qoidalari
- Xotira ajratishda prioritetlar va cheklovlar
- Xotira resurslaridan foydalanish samaradorligi

4. Kirish/chiqish jarayonlarini boshqarish:

- Kirish/chiqish talablarini bajarish qoidalari
- Kirish/chiqish jarayonlari prioritetlari
- Kirish/chiqish jarayonlari samaradorligini oshirish yechimlar

5. Oraliq va yakuniy holatlarni boshqarish:

- Jarayonlarning oraliq va yakuniy holatlari
- Jarayonlar o'rtasidagi o'zaro munosabatlar
- Jarayonlar tugatilganda yakuniy holatlarni qayd etish

Rivojlantirish ko'rsatkichlari:

1. Tizim samaradorligi:

- Protsessor va xotira resurslaridan foydalanish darajasi
- Jarayonlararo muloqot tezligi
- Kirish/chiqish operatsiyalari vaqti

2. Tizim barqarorligi:

- Jarayonlar o'rtasidagi o'zaro bog'liqlik darajasi
- Jarayonlarni boshqarishda xatolar soni
- Uzilishlar va to'xtab qolishlar soni

3. Tizim moslashuvchanlik:

- Yangi jarayonlarni yaratish va qo'shish imkoniyati
- Resurslarni dinamik qayta taqsimlash
- Tizimga yangi funktsionalliklar qo'shish imkoniyati

4. Tizim ishonchlilik:

- Jarayonlar boshqaruvi qoidalarining aniqlik va ishonchliligi
- Xato va noxush holatlarni boshqarish mexanizmlari
- Tizimning foydalanuvchi talablariga mos kelishi

Ushbu parametrlar va ko'rsatkichlar operatsion tizimlarning jarayonlar boshqaruvini samarali rejalashtirish va nazorat qilish uchun asos bo'ladi.

Operatsion tizimlarning jarayonlari boshqaruvini rejalashtirish va rivojlantirish parametrlari va ko'rsatkichlari, tizimning ishlashini o'rganish, optimallashtirish, xavfsizligini ta'minlash va foydalanuvchi tajribasini yaxshilash maqsadida tuziladi. Quyidagi asosiy parametrlar va ko'rsatkichlar operatsion tizimlarning jarayonlari boshqaruvi uchun kritik bo'lib hisoblanadi:

1. Protsessor Resurslari:

- **Protsessor yuklamasi:** Protsessor yuklamasini, protsesslar va dasturlar tomonidan qo'llab-quvvatlanayotgan va har bir protsessning protsessorda qancha vaqtni ishlatishini nazorat qilish.
- **Protsessor kerakli qismi soni:** Sistemning tizimi yo'qotish yo'li bilan bir protsessordan boshqa protsessorga o'tkaziladigan vaqtni kamaytirish uchun kerakli protsessorga egani sonini aniqlash.
- **Protsessorning har bir intizomning amal qilishining raqami:** Har bir intizomning ishlashi vaqti, xotiraning ne'matda qanday qismi uchun ishlatilishi mumkinligini aniqlash.

2. Xotira va Operativ Xotira:

- **Bo'sh xotira miqdori:** Bo'sh xotira miqdorini nazorat qilish, boshqa dasturlar uchun xotirani qo'llashga yordam beradi.
- **Xotira sohasining ajratilishi:** Fayllarni o'qish va yozish va boshqa dasturlar tomonidan ishlatiladigan xotira bo'lgan operatsion tizim ishlaydi.

3. Dastur va Protsesslar:

- **Dasturlarning amalda ishlayotgan muvaffaqiyati:** Dastur amaliyotlari, dasturlar amaliyotlari tomonidan qo'llab-quvvatlanadi, ularning bajarilishi, ishlayotgan vaqtida ishlatilishi, dastur tarzida ishlatilishi va joriy yuklamani aniqlash.
- **Protsesslarning bajarilishi va joriy ishchi joyi:** Qancha protsess bajarilishi mumkin, ularning ishga tushishi vaqtini va ishchi joyini nazorat qilish.

4. Tarmoq Boshqaruvi:

- **Tarmoq xavfsizligi:** Tarmoq faolliklarini va xavfsizlik holatini nazorat qilish, tarmoq xavfsizlik sozlamalarini sozlash va xavfsizlik holatini tekshirish.
- **Tarmoq yuklamasi:** Tarmoq xizmatlarining ishlab chiqarish jarayonlarini va ish rejimini nazorat qilish, tarmoq xavfsizligini o'rganish va optimallashtirish.

5. Foydalanuvchi Interfeysi:

- **Foydalanuvchi tomonidan amalga oshirilgan dasturlar va amal :** Foydalanuvchi interfeysini (UI) nazorat qilish, foydalanuvchi tajribasini yaxshilash va amalga oshirilgan dasturlarning bajarilishini va natijalarini monitor qilish.

6. Xavfsizlik va Maxfiylik:

- **Xavfsizlik faoliyatlarini nazorat qilish:** Xavfsizlik jarayonlarini va muddatlarini nazorat qilish, xavfsizlik nuqtalarini aniqlash va xavfsizlik sozlamalarini optimallashtirish.
- **Maxfiylik va himoya holatini nazorat qilish:** Maxfiylik ma'lumotlari va shaxsiy axborotlar himoyalanganligini nazorat qilish, maxfiylik sozlamalarini sozlash va himoya holatini nazorat qilish.

7. Monitoring va Diagnostika:

- **Monitoring va jarayonlarni diagnos etish:** Tizim faoliyatini nazorat qilish, amaliyotlar va xatoliklarni diagnos etish uchun dastlabki yuklash, xatoliklar va halolatlarni yozish va tekshirish.

S46.Zamonaviy operatsion tizimlar ilovalari

Zamonaviy operatsion tizimlar, bugungi texnologiyalarga asoslangan va ko'plab turli loyihalarni bajarish, xizmatlar va tizimlarni o'rganish, boshqarish va taqdim etish imkonini ta'minlaydigan tizimlardir. Quyidagi, zamonaviy operatsion tizimlarning mashhur va ko'p foydalaniladigan ilovalaridan ba'zi misollar:

1. Microsoft Windows:

- **Windows 10:** Microsoft tomonidan ishlab chiqilgan eng oxirgi versiya, moslashuvchan interfeys va qulay foydalanishni ta'minlaydi.
- **Windows Server 2019:** Server dasturlash uchun mo'ljallangan, moslashuvchan va xavfsizlik bilan yaxshi o'rganilgan Windows Server versiyasi.

2. macOS:

- **macOS Big Sur:** Apple tomonidan ishlab chiqilgan eng oxirgi versiya, kuchli grafik interfeysi va boshqaruv qulayliklari bilan tan olinadi.
- **macOS Server:** Apple'ning server dasturlash uchun mo'ljallangan operatsion tizimi, server loyihalarini boshqarish va yuklashga imkon beradi.

3. Linux:

- **Ubuntu:** Bepul va ochiq manba operatsion tizimi, o'rganish va ishlayotganloq uchun yaxshi tan olinadi.
- **Fedora:** Red Hat tomonidan ishlab chiqilgan, yangilanishlari va so'nggi xususiyatlari bilan tan olinadi.
- **CentOS:** Red Hat Enterprise Linux (RHEL) distributivining bir xil funksiyasiga ega bepul distributiv.

4. Chrome OS:

- **Chromebook:** Google tomonidan ishlab chiqilgan, Chrome OS operatsion tizimini qo'llagan, o'ziga xos funksiyalar va yaxshi qulayliklar bilan tan olinadi.

5. Android:

- **Android 12:** Google tomonidan ishlab chiqilgan, mobil qurilmalar va planshetlar uchun eng oxirgi versiya, qulayliklar va yangilanishlar bilan tan olinadi.

6. iOS:

- **iOS 15:** Apple tomonidan ishlab chiqilgan, iPhone va iPad qurilmalari uchun eng oxirgi versiya, keng qo'llaniladigan interfeysi va yaxshi qulayliklari bilan tan olinadi.

7. FreeBSD:

- **FreeBSD 13:** Unix bilan birga ishlovchi, tarmoq va server loyihalarini boshqarish uchun qo'llaniladi.

8. IBM AIX:

- **AIX 7.2:** IBM tomonidan ishlab chiqilgan, xususiy tizim va server loyihalarini boshqarish uchun mo'ljallangan.

Bu, zamonaviy operatsion tizimlar va ularning mashhur ilovalari haqidagi faqat bir qancha misollardir. Har bir operatsion tizimning o'zining xususiyatlari va imkoniyatlari mavjud bo'lib, ular qanday ishlab chiqilgan va ishlovchilar uchun qulayliklar ta'minlaydi.

S47. Operatsion tizimlarda hisoblash jarayoni

Operatsion tizimlarida hisoblash jarayoni asosiy quyidagi komponentlardan iborat:

1. **Protsessor (CPU):** Hisoblash jarayonlarini bajaruvchi asosiy birlik. Protsessor buyruqlarni o'qiydi, dekodlaydi va bajaradi.
2. **Xotira:** Dasturlar va ma'lumotlar saqlanadigan joy. Protsessor xotiradan ma'lumotlarni o'qib, qayta ishlash uchun foydalanadi.
3. **Kirish/Chiqish (I/O) tizimi:** Tizimga tashqi qurilmalardan ma'lumotlarni kiritish va qayta ishlangan ma'lumotlarni chiqarish uchun xizmat qiladi.

Hisoblash jarayonining bosqichlari:

1. **Buyruqlarni olish:** Protsessor xotiradan dastur buyruqlarini o'qib oladi.
2. **Buyruqlarni dekodlash:** O'qib olingan buyruqlar protsessor tomonidan tarjima qilinadi va bajarilish uchun tayyorgarlik ko'riladi.
3. **Bajarish:** Dekodlangan buyruqlar protsessor tomonidan bajariladi. Bu jarayon ma'lumotlar ustida arifmetik va mantiqiy amallarni bajarishni o'z ichiga oladi.
4. **Natijalarni yozish:** Bajarilgan amallarning natijalari xotiraga yoziladi yoki tashqi qurilmalarga yuboriladi.
5. **Keyingi buyruqni olish:** Bajarilgan buyruqdan keyin protsessor navbatdagi buyruqni olishga o'tadi.

Bu jarayon uzluksiz ravishda takrorlanib turadi, tizimning ishlashi uchun tabiiy ravishda zarur bo'lgan asosiy mexanizm hisoblanadi.

Operatsion tizim bu hisoblash jarayonini muvofiqlashtirish, samaradorlashtirish va boshqarish uchun javobgar bo'ladi. U protsessor, xotira va kirish/chiqish tizimlarini samarali boshqarish orqali hisoblash jarayonini yaxshilaydi.

Operatsion tizimlarda hisoblash jarayoni, tizimning asosiy funksiyalaridan biridir. Bu jarayon, kompyuter tizimida dasturlarni boshqarish va foydalanuvchilarning buyruqlarini bajarish uchun ishlatiladi. Hisoblash jarayoni asosan quyidagi bosqichlardan iborat:

1. Buyruqlarni olish:

Operatsion tizim, foydalanuvchi va dasturlarning buyruqlarini qabul qiladi.

Foydalanuvchilar ko'p turlarda buyruqlarni berish uchun terminal, GUI (Graphical User Interface) yoki boshqa interfeysdan foydalanishlari mumkin. Dastur buyruqlari, foydalanuvchi buyruqlari va tarmoq buyruqlari kabi turli turdagi buyruqlarni olish jarayonlarini ta'minlaydi.

2. Buyruqlarni tushuntirish:

Operatsion tizim buyruqlarni tushuntiradi va ularni qo'llab-quvvatlaydigan dasturlarni aniqlaydi. Bu jarayonda, tizim buyruqning maqsadini tushuntiradi va uni bajaruvchi

dasturni aniqlaydi. Dastur buyruqlari amaliy buyruqlar yoki tegishli dasturlarga o'rgatiladi.

3. Buyruqlarni bajarish:

Bu jarayonda, tizim buyruqlarni bajaradi va natijani olish uchun kerakli operatsiyalarni bajaradi. Masalan, fayllarni yaratish, o'qish, yozish, ko'chirish yoki o'chirish, protsesslarni boshqarish, tarmoq operatsiyalari kabi amallarni bajarish.

4. Natijani qaytarish:

Tizim bajarilgan buyruqlarning natijasini foydalanuvchiga qaytaradi. Bu natijalar odatda dasturni ijro etish natijasi, xato xabarlar yoki so'rovlarni bajarish natijalari bo'lishi mumkin.

5. Buyruqlarni bajarishni boshqarish:

Operatsion tizim buyruqlarni bajarishni boshqarish jarayonlarida, kerakli resurslarni, dasturlarni va protsesslarni boshqarish uchun kerakli komponentlarni aktivlashtiradi va boshqaradi. Bu jarayonda, tizim resurslarini, dastur buyruqlarini va protsesslarni bajarish uchun kerakli resurslarni ajratib berish, ularga murojaat qilish, ularning ishini nazorat qilish, va jarayonlarni samarali bajarishni ta'minlash kiritiladi.

6. Jarayonlarni boshqarish:

Operatsion tizim, bir nechta dasturlarni bir vaqtning o'zida bajarishni ta'minlash uchun multidasturlashni va multiprotsessorlashni qo'llab-quvvatlayadi. Bu jarayonda, tizim dasturlarni bajarishni o'zaro ko'rsatib, resurslarni ishlab chiqarish, ish rejimini nazorat qilish va bajarilayotgan dasturlarni monitor qilish uchun qo'llanadi.

7. Xatoliklar va Iste'molchilar uchun Xabarlar:

Operatsion tizim, buyruqlarni bajarish jarayonida chiqadigan xatoliklarni aniqlaydi va foydalanuvchilarga qayta urinish buyruqlari bermoqchi bo'lgan iste'molchilar uchun xabarlar chiqaradi.

8. Bajarilgan buyruqlarni uyg'otish:

Buyruqlar bajarilgandan so'ng, tizim ulardan olingan natijani foydalanuvchiga qaytaradi va qo'llab-quvvatlash uchun keyingi operatsiyalarni bajaradi.

Operatsion tizimlarning hisoblash jarayoni bu tizimning asosiy qismlaridan biridir va kompyuter tizimining ishlashida kritik ahamiyatga ega bo'lib, buyruqlarni qabul qilish, tushuntirish, bajarish va natijalarni qaytarishni ta'minlaydi.

S48. Multidasturlash. Ajratilgan vaqt tizimlarida ko'p foydalanuvchi rejimi

Multidasturlash - bu bir vaqtning o'zida bir nechta dasturlarni bajarish qobiliyati. Operatsion tizimlar multidasturlash imkonini beradi va uni quyidagi usullarda amalga oshiradi:

1. **Ko'p jarayonli:** Bir vaqtda bir nechta jarayonlar(protssellar) bajariladi, operatsion tizim ularni ajratib, navbat bilan CPU vaqtini ajratadi.
2. **Ko'p yadroli:** Bir paytda bir nechta protsessorlar mavjud bo'ladi, ular bir vaqtda turli jarayonlarni bajaradilar.
3. **Virtualizatsiya:** Bir vaqtda bir nechta virtual mashina mavjud bo'ladi, ular bir-biridan ajratilgan holda ishlaydilar.

Ajratilgan vaqt tizimlarida ko'p foydalanuvchi rejimi

Ajratilgan vaqt tizimlarida bir nechta foydalanuvchilar bir vaqtning o'zida ishlashi mumkin. Bunda operatsion tizim quyidagi vazifalarni bajaradi:

1. **Foydalanuvchilarni identifikatsiya qilish:** Har bir foydalanuvchini nomlari, parollari yoki boshqa usullar orqali aniqlab oladi.
2. **Resurslarni boshqarish:** Protsessor, xotira, fayllar, qurilmalar kabi resurslarni foydalanuvchilar o'rtasida muvozanatli taqsimlaydi.
3. **Foydalanuvchilar uchun himoya:** Har bir foydalanuvchining ma'lumotlari va ishlari boshqalardan himoya qilinadi.
4. **Muloqot imkonini berish:** Foydalanuvchilar o'rtasida ma'lumot almashish, hujjatlar bo'yicha hamkorlik qilish imkonini beradi.
5. **Nazorat va hisobotlar:** Foydalanuvchilar ish faoliyati va resurslardan foydalanish haqida hisobotlarni olish imkonini beradi.

Ko'p foydalanuvchili tizimlar yirik kompaniyalar, universitet, hukumat va boshqa tashkilotlarda keng qo'llaniladi. Ular bir vaqtda bir nechta foydalanuvchilarning turli ehtiyojlarini qondira oladi.

Multidasturlash, bir nechta dasturlarni bir vaqtning o'zida ishga tushirish imkonini ta'minlaydigan jarayondir. Ajratilgan vaqt tizimlarida ko'p foydalanuvchi rejimi, ko'plab foydalanuvchilar tomonidan boshqariladigan vaqtda bir nechta dasturlarni ishga tushirish imkonini ta'minlaydi. Bu, yaqin zamonda, serverlar, bulut kompyuterlar, baza tizimlari va ko'p foydalanuvchili tizimlarda keng qo'llaniladi. Quyidagi jarayonlar ko'plab foydalanuvchi rejimida multidasturlashni ta'minlashda ishlaydi:

1. Resurslarni taqsimlash:

- **Protsessor:** Ajratilgan vaqt tizimlarida, har bir foydalanuvchi uchun protsessorni qanday vaqt ishlatishni belgilab berish zarur. Bu, protsessor yuklamasini qanday ajratish va protsessor resurslarini maqsadli ravishda boshqarishga yordam beradi.

- **Xotira:** Har bir foydalanuvchining o'z xotira joyini e'lon qilish va har bir foydalanuvchi uchun xotira miqdorini chegaralash juda muhimdir. Ajratilgan vaqt tizimlarida, xotira resurslarini to'g'ri taqsimlash foydalanuvchilar orasida qo'llanishni ta'minlaydi.
- **Tarmoq bandwithi:** Ko'plab foydalanuvchi rejimida, tarmoq bandwithi ham ajratiladi. Har bir foydalanuvchi uchun kerakli tarmoq bandwithini aniqlash va taqsimlash, to'g'ri ishlovchi tarmoq xizmatini ta'minlaydi.

2. Dasturlarni ajratish:

- **Vaqtни ajratish:** Ajratilgan vaqt tizimlarida, har bir foydalanuvchi uchun ishga tushiriladigan vaqtни belgilash juda muhimdir. Bunda, kerakli protsesslarni va dasturlarni qanday vaqt ishga tushirish kerakligini aniqlash kiritiladi.
- **Prioritetlar:** Har bir foydalanuvchi uchun ishga tushiriladigan dasturlar va protsesslarning prioritetlarini aniqlash ham juda muhimdir. Misol uchun, urg'uchilik dasturlarining, operatsion tizim dasturlarining va qo'shimcha foydalanuvchi dasturlarining prioritetlarini ta'minlash muhimdir.

3. Foydalanuvchi tanlash va yo'nalish:

- **Foydalanuvchi tanlash:** Ajratilgan vaqt tizimlarida, foydalanuvchilar o'zlariga mos foydalanuvchi tanlash yo'nalishlarini tanlashlari mumkin. Bu, foydalanuvchilar uchun ajratilgan vaqt tizimlarida hamkorlikni ta'minlashda muhim ahamiyatga ega.
- **Foydalanuvchi tanlash rejimi:** Ajratilgan vaqt tizimlarida, bir nechta foydalanuvchi tanlash rejimlarini taqdim etish mumkin. Masalan, birinchi kelmachilar qo'llanuvchilari, hamkorlikda ishlaydigan qo'llanuvchilari yoki administratorlar kabi.

4. Monitor qilish va boshqarish:

- **Jarayonlarni monitor qilish:** Ajratilgan vaqt tizimlarida, har bir foydalanuvchi uchun ish rejimini va qo'llanuvchi dasturlarning faoliyatini monitor qilish juda muhimdir. Bu, jarayonlarni to'g'ri boshqarish, xatoliklarni aniqlash va takomillashtirish imkonini ta'minlaydi.
- **Boshqarish:** Ko'plab foydalanuvchi rejimida, boshqaruv tizimlari har bir foydalanuvchi uchun faol boshqaruv imkoniyatlarini ta'minlaydi. Bu, dastur bajarilishi va qo'llanuvchi dasturlarining sozlamalarini o'zgartirish va boshqa amallarni bajarish uchun ruxsat berishni o'z ichiga oladi.

5. Foydalanuvchi interfeysi:

- **Qo'llanishga mos moslamalar:** Ajratilgan vaqt tizimlarida, foydalanuvchilar o'zlariga mos moslamalar yordamida ishlay oladilar. Bu, mos moslamalar foydalanuvchilar uchun qulaylik yaratadi va ish rejimini aniqlashga yordam beradi.
- **Birgalikda ishlash:** Ko'plab foydalanuvchi rejimida, bir nechta foydalanuvchi o'zlarini bir xil vaqtda birgalikda ishlovchi foydalanuvchi interfeysini qo'llashadi. Bu, jamoatchilikni yaxshilash va hamkorlikni ko'rish uchun juda muhimdir.

6. Xavfsizlik:

- **Maxfiylik:** Ajratilgan vaqt tizimlarida, foydalanuvchilar orasidagi max

S49.Jarayon xolati diagrammalari va jarayon diskriptori

Jarayon xolati diagrammalari

Jarayon xolati diagrammalari operatsion tizimda jarayonlarning holatlarini aks ettiradi. Bu diagrammalar jarayonlarning turli xolatlariga o'tishini ko'rsatadi. Asosiy jarayon xolatlari quyidagilar:

1. **Yangi (New):** Yangi yaratilgan, lekin hali bajarilmayotgan jarayon.
2. **Tayyor (Ready):** Bajarilishi uchun tayyor jarayon, lekin hali protsessor tomonidan boshlanmagan.
3. **Bajarilmoqda (Running):** Protsessor tomonidan hozirda bajarilayotgan jarayon.
4. **Bloklangan (Blocked):** Biror resurs (masalan, kirish/chiqish operatsiyasi) uchun kutayotgan jarayon.
5. **Tugatildi (Terminated):** Bajarilishini yakunlagan jarayon.

Jarayonlar quyidagi xolat o'tishlarini amalga oshiradi:

- Yangi -> Tayyor
- Tayyor -> Bajarilmoqda
- Bajarilmoqda -> Bloklangan
- Bloklangan -> Tayyor
- Bajarilmoqda -> Tugatildi

Jarayon diskriptori

Jarayon diskriptori - operatsion tizim tomonidan har bir jarayon uchun yaratilgan ma'lumotlar tuzilmasi. Uning tarkibida quyidagi asosiy ma'lumotlar mavjud:

1. **Jarayon identifikatori (PID):** Har bir jarayonning unikal raqami.
2. **Holat:** Yuqorida ko'rsatilgan jarayon xolatlari.
3. **Prioritet:** Jarayonni bajarishda protsessorning ajratadigan vaqt miqdori.
4. **Protsessordan foydalanish vaqti:** Jarayon tomonidan protsessordan foydalangan vaqt.
5. **Xotira sarfi:** Jarayon tomonidan egallangan xotira miqdori.
6. **Keyingi/oldingi jarayonlar:** Agar jarayon boshqa jarayonlar bilan bog'liq bo'lsa, ularning ma'lumotlari.
7. **Kirish/chiqish ma'lumotlari:** Jarayon tomonidan foydalanilgan kirish/chiqish operatsiyalari.

Operatsion tizim jarayon diskriptorlarini saqlash, yangilash va tahlil qilish orqali jarayonlarni boshqaradi.

Jarayon xolati diagrammalari (Process State Diagrams) operatsion tizimlarda protsesslar yoki jarayonlar o'rtasidagi o'zgarishlarni vizual ravishda ifodalovchi modellar hisoblanadi. Bu diagrammalar jarayonlar yoki protsesslar orasidagi o'zgarishlarni ko'rsatadi, masalan, bir protsessning boshlanishi, to'xtash va faol holatlar orasidagi o'zgarishlar kabi. Jarayon diskriptorlari (Process Descriptors) esa operatsion tizimdagi har bir protsess uchun ma'lumotlarni saqlash uchun ishlatiladi.

Bu ma'lumotlar protsessning holati, xotira joyi, ehtiyot qatorlar, ishchi fayllar kabi protsessning amaliyotlarini boshqarishga yordam beradi.

Quyidagi misolda, bir protsessning odatda ko'rib chiqiladigan jarayon xolati diagrammasi va jarayon diskriptori ko'rsatilgan:

Jarayon Xolati Diagrammasi (Process State Diagram):

- **Boshlash:** Protsessning yaratilishi.
- **Faol:** Protsessning ishlayotgan holati.
- **To'xtash:** Protsessning ishga tushirilishi.
- **Xotira:** Protsessning xotira holati.

Jarayon Diskriptori (Process Descriptor):

Bu misolda, Web Server nomli bir protsess uchun jarayon xolati diagrammasi va jarayon diskriptori ko'rsatilgan. Jarayon xolati diagrammasida, protsessning boshlashidan boshlab faol holatga o'tishi va keyin to'xtash holatga o'tishi ko'rsatilgan. Jarayon diskriptorida esa, protsess haqidagi qiziqarli ma'lumotlar ko'rsatilgan, masalan, protsessning holati, oxirgi to'xtash vaqti, xotira manzili, ehtiyot qatorlar va ishchi fayllar. Bu ma'lumotlar protsessning boshqarilishi uchun muhimdir

S50. Jarayon va topshiriqlarni rejalashtirish va dispetcherlash

Jarayon va topshiriqlarni rejalashtirish va dispetcherlash, operatsion tizimda bajarilishi kerak bo'lgan kritik vazifalardan biridir. Bu, protsesslarni boshqarish va ularga resurslarni ajratib berish jarayonidir. Quyidagi bosqichlar rejalashtirish va dispetcherlash jarayonini tushuntiradi:

1. Topshiriqlarni Rejalashtirish (Task Scheduling):

- **Topshiriqlar olish:** Foydalanuvchi topshiriqlarini yoki dastur buyruqlarini qabul qilish, qatori bilan o'qish, yoki tarmoq orqali kelgan buyruqlarni qabul qilish.
- **Topshiriqlarni qayta ishlovchi loyihalarni aniqlash:** Qabul qilingan topshiriqlarni bajarish uchun kerakli resurslarni aniqlash va ularga topshiriqlarni qayta ishlovchi loyihalarni ajratib berish. Bu qadamda, topshiriqlarni qayta ishlovchi loyihalar (processes) yaratiladi.
- **Topshiriqlarni bajarish tartibi:** Ajratilgan topshiriqlarni bajarish tartibini aniqlash, ularni bajarish uchun protsessor va boshqa resurslarni ajratib berish.

2. Dispetcherlash (Dispatching):

- **Topshiriqlarni ishga tushirish:** Topshiriqlarni bajarish uchun ishga tushirish. Bu qadamda, dispetcher bajarishga tayyor bo'lgan topshiriqlarni protsessorlarga yoki boshqa resurslarga yuboradi.
- **Kontekst almashish:** Oldingi topshiriqni bajarishning holatini saqlab qolish va keyingi topshiriqlarni ishga tushirish orqali uning holatini qayta tiklash.
- **Vaqtini bajarish:** Protsessorlarga bajarish uchun topshiriqlarni taqsimlash va ularni bajarish uchun vaqtini aniqlash.

Misol:

Topshiriqlarni Rejalashtirish:

1. Foydalanuvchidan kelgan topshiriqlarni qabul qilish.
2. Qabul qilingan topshiriqlar uchun loyihalar yaratish.
3. Ajratilgan topshiriqlarni bajarish tartibini aniqlash.

Dispetcherlash:

1. Birinchi loyiha bajarilayotgan vaqt kelganda, protsessorni ajratish.
2. Ajratilgan protsessorni boshqa topshiriqlar uchun ishga tushirish.
3. Bajarilgan topshiriqlarni saqlash va keyingi topshiriqlarni bajarish uchun tayyorlash.
4. Kontekst almashish - bajarilgan topshiriqlarning holatini saqlash.

Rejalashtirish va dispetcherlash jarayonlari, operatsion tizimning yaxshi ishlashi uchun zarur bo'lgan protsesslardir. Bu jarayonlar tizimda ishlovchi topshiriqlarni bajarishni ta'minlaydi va resurslarni samarali ravishda ishlatish imkonini beradi.

Jarayon rejalashtirish

Operatsion tizim jarayonlarni bajarish uchun protsessorning cheklangan vaqtini samarali taqsimlash kerak. Buning uchun quyidagi rejalashtirish algoritmlari qo'llaniladi:

1. **FIFO (First-In-First-Out):** Jarayonlar kelish ketma-ketligida bajariladi. Oddiy va oson amalga oshiriladi, lekin adolatli emas.
2. **SJF (Shortest Job First):** Eng qisqa bajarish vaqtiga ega jarayon birinchi bajariladi. Samarali, lekin bajarish vaqtini oldindan bilish talab qilinadi.
3. **Prioritetli:** Har bir jarayonga ma'lum prioritet beriladi. Yuqori prioritetli jarayonlar birinchi bajariladi.
4. **Aylanma (Round Robin):** Jarayonlarga teng vaqt ajratiladi, bunda kichik vaqt dilimi (quantum) ishlatiladi.
5. **Multiprogrammalash darajasi:** Bir paytda bajarilishi kerak bo'lgan jarayonlar soni operatsion tizim tomonidan boshqariladi.

Topshiriq dispetcherlash

Topshiriq dispetcherlash - bu jarayonlar orasida protsessorlar vaqtini taqsimlash. Bunda quyidagi asosiy jarayonlar amalga oshiriladi:

1. **Jarayon yaratish:** Yangi jarayonni yaratish va uni jarayon diskriptoriga joylash.
2. **Kontekstni almashtirish:** Aktiv jarayondan yangisiga o'tish uchun registrlar, xotira, CPUning holati kabi ma'lumotlarni almashish.
3. **Bloklanish va bloklashdan chiqish:** Jarayon bloklanishi yoki bloklashdan chiqishi hisobga olinadi.
4. **Tugatish:** Jarayonning bajarilishi tugagandan keyin resurslarni bo'shatish.

Dispetcher topshiriq kelishiga qarab, rejalashtirish algoritmlarini qo'llab, jarayonlar orasida protsessorni almashtirib turadi.

Jarayon va topshiriq rejalashtirish operatsion tizimning asosiy funksiyalaridan biri bo'lib, tizimning samaradorligini oshirishda muhim rol o'ynaydi.

S51.O'rnatilgan tizimlar imkoniyatlari.

O'rnatilgan tizimlar - bu mikroprotessorlar, mikrokontrollerlar yoki maxsus qurilmalarga asoslangan o'ziga xos dasturiy ta'minot va apparat ta'minotiga ega bo'lgan kompyuter tizimlardir. Bu tizimlar quyidagi imkoniyatlarga ega:

1. **Xususiylashtirish:** O'rnatilgan tizimlar aniq vazifalar uchun maxsus loyihalangan bo'ladi, shu sababli ular juda moslashuvchan va maqsadli bo'ladi.
2. **Tez ishlab chiqish:** Maxsus dastlabki loyihalar va tayyor komponentlardan foydalanish o'rnatilgan tizimlarni tez ishlab chiqish imkonini beradi.
3. **Past narx:** O'rnatilgan tizimlar odatda umumiy maqsadli kompyuterlarga nisbatan arzonroq bo'ladi.
4. **Unumli ishlash:** O'rnatilgan tizimlar ko'pincha umumiy maqsadli kompyuterlarga nisbatan kam quvvatli protsessorlarga ega bo'ladi, lekin maxsus vazifalarni bajarish uchun yetarli bo'ladi.
5. **Samaradorlik:** O'rnatilgan tizimlar faqat o'zlari uchun zarur bo'lgan funksiyalarni bajaradi, shuning uchun resurslardan samarali foydalanadi.
6. **Kichik o'lcham:** O'rnatilgan tizimlar odatda juda kichik hajmga ega bo'lib, mos kelmaydigan joylarga o'rnatilishi mumkin.
7. **Ishonchlilik:** O'rnatilgan tizimlar faqat o'zlari uchun zarur funksiyalarni bajaradi, shu sababli ularda ishonchlilik darajasi yuqori bo'ladi.

O'rnatilgan tizimlar turli xil sohalarda keng qo'llaniladi: sanoat avtomatlashtirilishi, mashinasozlik, tibbiyot, aloqa tizimlarida, uylar va binolar avtomatlashtirilishi, transport, urush sanoati kabi.

52 Real vaqt operatsion tizimlari.

Real vaqt operatsion tizimlari (RVOT) - bu o'rnatilgan tizimlar uchun maxsus rivojlantirilgan operatsion tizimlar bo'lib, quyidagi asosiy xususiyatlarga ega:

1. **Vaqt cheklovi:** RVOT-lar ma'lum vaqt chegarasida ma'lum vazifalarni bajarishga qodir bo'lishi kerak. Masalan, sensor ma'lumotlarini o'z vaqtida qayta ishlash yoki boshqaruv tizimlarini boshqarish.
2. **Reaksiya tezligi:** RVOT-lar tashqi voqealarga juda tez javob qaytarishi kerak. Masalan, avtonom transport vositalarida to'qnashuvlarni oldini olish.
3. **Determinantlik:** RVOT-lar bir xil sharoitlarda bir xil natija qaytarishi kerak. Bu ularning xavfsizlikni ta'minlashida muhimdir.
4. **Ishonarlilik:** RVOT-lar uzluksiz va uzog'i bilan foydalanish uchun mo'ljallanganligi sababli juda ishonchli bo'lishi kerak.
5. **Rejalash va dispetcherlash:** RVOT-lar jarayonlar uchun qat'iy va oldindan belgilangan jadval bo'yicha ishlaydi.

RVOT-lar quyidagi xususiyatlarni ham taqdim etadi:

- **Real vaqtda jarayonlarni boshqarish**
- **Prediktiv va invariantli nazorat**
- **Reaksiya va bajarish vaqti ustidan nazorat**
- **Resurslardan samarali foydalanish**

- **Uzluksiz va ishonchli ish**

RVOT-lar avioprognoslar, tibbiy diagnostika, transport boshqaruvi, sanoat avtomatlashtirilishi, robotlar va boshqa o'rnatilgan tizimlar uchun keng qo'llaniladi.

Real vaqt operatsion tizimlari

Real vaqt operatsion tizimlari (RVOT) - bu o'rnatilgan tizimlar uchun maxsus rivojlantirilgan operatsion tizimlar bo'lib, quyidagi asosiy xususiyatlarga ega:

1. **Vaqt cheklovi:** RVOT-lar ma'lum vaqt chegarasida ma'lum vazifalarni bajarishga qodir bo'lishi kerak. Masalan, sensor ma'lumotlarini o'z vaqtida qayta ishlash yoki boshqaruv tizimlarini boshqarish.
2. **Reaksiya tezligi:** RVOT-lar tashqi voqealarga juda tez javob qaytarishi kerak. Masalan, avtonom transport vositalarida to'qnashuvlarni oldini olish.
3. **Determinantlik:** RVOT-lar bir xil sharoitlarda bir xil natija qaytarishi kerak. Bu ularning xavfsizlikni ta'minlashida muhimdir.
4. **Ishonarlilik:** RVOT-lar uzluksiz va uzog'i bilan foydalanish uchun mo'ljallanganligi sababli juda ishonchli bo'lishi kerak.
5. **Rejalash va dispetcherlash:** RVOT-lar jarayonlar uchun qat'iy va oldindan belgilangan jadval bo'yicha ishlaydi.

RVOT-lar quyidagi xususiyatlarni ham taqdim etadi:

- **Real vaqtda jarayonlarni boshqarish**
- **Prediktiv va invariantli nazorat**
- **Reaksiya va bajarish vaqti ustidan nazorat**
- **Resurslardan samarali foydalanish**
- **Uzluksiz va ishonchli ish**

RVOT-lar avioprognoslar, tibbiy diagnostika, transport boshqaruvi, sanoat avtomatlashtirilishi, robotlar va boshqa o'rnatilgan tizimlar uchun keng qo'llaniladi.

S53. O'rnatilgan tizimlar uchun operatsion tizimlar

O'rnatilgan tizimlar uchun operatsion tizimlar, odatda, real vaqt operatsion tizimlarini (RVOT) o'z ichiga oladi. Bu operatsion tizimlar o'rnatilgan tizimlarning xususiyatlari va talablariga moslashtirilgandir:

1. **Real vaqtda ishlash:** RVOT-lar reaksiya tezligi, determinantlik va vaqt chegaralariga rioya qilish orqali real vaqtda ishlashni ta'minlaydi.
2. **Kam resurs talab qilish:** O'rnatilgan tizimlar, odatda, cheklangan resurslar (protessor, xotira, quvvat) ga ega bo'ladi. Shu sababli, ularning operatsion tizimlari ham kam resurs talab qilishi kerak.
3. **Maxsuslashtirish:** O'rnatilgan tizimlar aniq vazifalar uchun mo'ljallanganligi uchun, operatsion tizimlar ham aniq vazifalar uchun maxsuslantiriladi.
4. **Kichik hajm:** O'rnatilgan tizimlar kompakt va kichik bo'lgani uchun, ularning operatsion tizimlarining ham hajmi kichik bo'lishi kerak.
5. **Ishonchlilik:** O'rnatilgan tizimlar, odatda, uzluksiz ishlatiladi, shuning uchun ularning operatsion tizimlarida yuqori ishonchlilik muhimdir.

O'rnatilgan tizimlar uchun eng ko'p qo'llaniladigan operatsion tizimlar quyidagilar:

- **VxWorks** - sanoat va harbiy sohalarda keng qo'llaniladi.
- **QNX Neutrino** - xavfsizlik talablari yuqori bo'lgan sohalarda (meditsina, transport) qo'llaniladi.
- **FreeRTOS** - yengil va arzon o'rnatilgan tizimlar uchun mo'ljallangan.
- **Linux** - ochiq kodli bo'lib, keng moslashuvchan va takomillashtirilgan.
- **Windows Embedded** - Microsoft kompaniyasi tomonidan ishlab chiqilgan o'rnatilgan tizimlar uchun.

Ushbu operatsion tizimlar o'rnatilgan tizimlarning turli xil talablariga mos keladigan xususiyatlarga ega. Bu, o'rnatilgan tizimlar uchun keng qo'llaniladigan operatsion tizimlardan faqat ba'zi misollar hisoblanadi. Har birining o'zining afzalliklari, imkoniyatlari va xususiyatlari mavjud. O'rnatilgan tizimni aniqlangan maqsadlarga mos ravishda tanlash juda muhim.

S54 O'rnatilgan tizimlarga qo'yiladigan asosiy talablar.

O'rnatilgan tizimlar o'ziga xos xususiyatlarga ega bo'lib, ular uchun quyidagi asosiy talablar qo'yiladi:

1. **Real vaqtda ishlash:**

- Tizim reaksiya tezligi va determinantligi bilan ajralib turishi kerak.
- Vaqt chegaralarini qat'iy ta'minlashi lozim.

2. **Kam resurs talab qilish:**

- Cheklangan protsessor quvvati, xotira sig'imi va quvvat sarfiga ega bo'lishi kerak.
- Resurslardan samarali foydalanish.

3. **Maxsuslashtirish:**

- Aniq vazifalar uchun mo'ljallangan bo'lishi lozim.
- Foydalanuvchi talablariga moslashtirilgan bo'lishi kerak.

4. **Kichik hajm:**

- Tizim hajmi kichik va kompakt bo'lishi kerak.
- Joylashtirish va integratsiya imkoniyatlari yaxshi bo'lishi lozim.

5. **Ishonchlilik:**

- Uzluksiz va uzun muddat ishlashi kerak.
- Xavfsizlik va ishonchlilik talablariga javob berishi lozim.

6. **Kuzatish va nazorat:**

- Tizim holatini monitoring qilish va boshqarish imkoniyatlari bo'lishi kerak.
- Diagnostika va xizmat ko'rsatish funksiyalarga ega bo'lishi lozim.

7. **Soddalik:**

- Foydalanish va boshqarish uchun sodda va tushunarli bo'lishi kerak.
- Murakkab funktsionallikka ega bo'lmasligi lozim.

Ushbu talablarni qoniqtirish uchun, o'rnatilgan tizimlar uchun maxsus operatsion tizimlar va dasturiy ta'minotlar ishlab chiqiladi.

O'rnatilgan tizimlarga qo'yiladigan asosiy talablar quyidagilar:

1. **Qulaylik:**

- **Moslashtirish va O'rnatish:** Tizimni o'rnatish va moslashtirishni oson qilish.
- **Foydalanish Qulayligi:** O'rnatilgan tizimni foydalanish qulay va aniq bo'lishi kerak.

2. **Xavfsizlik:**

- **Tarmoq Xavfsizligi:** Internetga ulanish o'rnatilgan tizim uchun kritik bo'lgan bir xususiyatdir. Tizim ma'lumotlarini himoya qilish va hujjatlarni ko'chirish qulayliklari shartlarida ta'minlash zarur.
- **Muhim axborotlar Xavfsizligi:** Muhim ma'lumotlar uchun yuqori darajada xavfsizlik ta'minlash.

3. Tezlik:

- **Yuklanish va ishlash Tezligi:** O'rnatilgan tizimni yuklash va boshlash tez va samarali bo'lishi lozim.
- **Amalga oshirish Tezligi:** Tizim ma'lumotlarini qidirish, ko'rish va ishlashni tez va aniq bajarish.

4. Muhit va Uygunlik:

- **Moslashtirilgan Tarmoq va Serverlar:** O'rnatilgan tizim, ma'lumotlarni o'zaro almashish va serverlar bilan integratsiyasini ta'minlash uchun moslashtirilgan bo'lishi kerak.
- **Ko'p foydalanuvchili muhitlar:** Agar tizim ko'p foydalanuvchili bo'lsa, uning muhitlariga moslashtirilishi kerak.

5. Yuqori Darajada Qo'llanish:

- **Bartaraf Ko'rsatish:** O'rnatilgan tizimda yo'l ko'rsatish, xatolik aniqlash va xato qo'llashning qulay ko'rsatishlar mavjud bo'lishi kerak.
- **Qo'llanuvchilar Uchun O'qituvchi Mavqelar:** Qo'llanuvchilarga moslashtirilgan va qulay bo'lgan o'qituvchi mavqelar taqdim etilishi kerak.

6. To'g'ri Hamda Sodda Ishlovchi:

- **Intuitiv Interfeys:** O'rnatilgan tizimga kirish va uni boshqarish uchun sodda va intuitiv interfeyslar zarur.
- **Yaxshi Xabarnomala:** Muammo yoki xatolik yuz berish paytida foydalanuvchilarga yaxshi xabarnomala ko'rsatish zarur.

7. Qo'llanish Kengaytirilish:

- **Ko'p Yo'nalishli Qo'llanish:** O'rnatilgan tizimning ko'p yo'nalishli bo'lishi va keng qo'llanish imkonini berish kerak.

8. Yaxshi Yuqori Darajadagi Xavfsizlik:

- **Yaxshi Xavfsizlik Ta'minlash:** Tizimning ma'lumotlarini himoya qilish uchun yuqori darajadagi xavfsizlik tizimlarini ta'minlash.

9. Modulli:

- **Modulli struktura:** Tizimni modullar bo'yicha tuzilishi, ta'minlash, yangilash va boshqarishni oson qiladi.

10. Tez Tezlashtirish va Yaxshi Yuqori Darajadagi O'zaro Ishtirok:

- **Tezlashtirish va Yangilanish:** Tizimni tezlashtirish va yangilanishga tayyor bo'lgan bo'lishi kerak.

11. Monitorlash:

- **Jarayonlarni Monitorlash:** Tizimning ishlashini, jarayonlarni va resurslarni monitor qilish uchun moslashtirilgan tizimlar ta'minlash.

Bu asosiy talablar, o'rnatilgan tizimlar uchun kritik bo'lgan va ma'lumotlar tizimi tahlil qilinib qo'yilgan asosiy talablardir. O'rnatilgan tizimlar, faqatgina ma'lumotlar tizimi va ishlab chiqilgan mahsulotlarni o'rnatishni talab etmaydi, balki hamda ularga qulay va ishlovchi bo'lishini talab etadi.

S55.Foydalanuvchi interfeysi va ob'ekt interfeysi

Foydalanuvchi Interfeysi (UI):

Foydalanuvchi interfeysi (UI) foydalanuvchilar bilan aloqani ta'minlaydi va ularga dastur yoki tizim funktsiyalariga kirish, ular bilan munosabatda bo'lish imkoniyatini beradi. UI, dastur yoki tizimni boshqarish va unga murojaat qilishning ko'rsatkichi bo'ladi. UI quyidagi turdagi asosiy komponentlardan iborat bo'ladi:

1. **Grafik interfeys (GUI):** Yuqori darajada grafik tizimlarni qo'llaydigan interfeys turi. Misol uchun, tugmachalar, oynalar, menyu elementlari, yozuv kiritish maydonchalari va boshqalar.
2. **Teskari interfeys (CLI):** Komanda qatorlaridan foydalanish orqali amallarni bajarish imkoniyatini ta'minlaydi. Bu turi mashhur Unix/Linux terminali va Windows PowerShell danib chiqarish mumkin.
3. **Tezlanish va ob'ekt interfeys (NUI/OUI):** Qo'llanuvchi tomonidan yaratilgan harakatlarga bo'lgan javobni o'rnatish uchun avtomatlashtirilgan interfeys turi. Misol uchun, sensorlar, kamera va boshqa qurilmalar orqali bajariladigan yo'nalishlar.
4. **Veb interfeys (Web UI):** Brauzer orqali foydalanuvchilar bilan aloqa o'rnatish imkoniyatini beradi. Bu interfeys turi HTML, CSS, JavaScript va boshqa veb texnologiyalari yordamida yaratiladi.

Foydalanuvchi interfeysining asosiy vazifalari quyidagilar:

- Foydalanuvchilar bilan munosabatda bo'lishni ta'minlash.
- Amalga oshirilishi kerak bo'lgan vazifalarni amalga oshirish uchun moslashtirilgan foydalanuvchi tomonidan oson aniq tizim yaratish.
- Foydalanuvchining dastur yoki tizimning funktsiyalaridan foydalanishini ta'minlash.

Ob'ekt Interfeysi (API):

Ob'ekt interfeysi (API), dasturlar va tizimlar o'rtasidagi aloqani o'tkazish uchun moslashtirilgan va qo'llanilgan o'zgaruvchilardan va funktsiyalardan iborat bo'lgan dasturlash interfeysidir. API o'z ichiga o'zgaruvchilar, funktsiyalar, protseduralar va boshqa dasturiy elementlarni o'z ichiga oladi.

API-ni ishlatish orqali, dastur yoki tizimning boshqa qismi bilan murojaat qilish va uning funktsiyalaridan foydalanish mumkin. API orqali, dastur qanday bilan amalga oshirilishi kerakligini aytish uchun tayyor funktsiyalar va jarayonlar mavjud. Misol uchun, veb-saytlar ko'plab API-lar orqali ma'lumotlarga murojaat qiladi.

Ob'ekt interfeysining asosiy vazifalari quyidagilar:

- Dastur yoki tizimning qanday amalga oshirilishi kerakligini ta'minlash.
- Boshqa dastur yoki tizimlar bilan o'rtasidagi aloqani o'tkazishni ta'minlash.
- Dastur yoki tizimning mahsulotlari va funksiyalaridan foydalanishni ta'minlash.

Foydalanuvchi interfeysi va ob'ekt interfeysi turli turdagi tizimlarda va dasturlarda qo'llaniladi. Bu ikkala interfeys turi dasturlash va tizimlar dasturlashda aloqani o'tkazishni osonlashtiradi va boshqarishni sahollarda yaxshi bir nutqni yaratadi.

S56. O'rnatilgan tizim qurilmalarini boshqarish ob'ekti bilan ulash vositalari.

O'rnatilgan tizimlar qurilmalarini boshqarish ob'ekti bilan ulash uchun quyidagi vositalar ishlatiladi:

1. Mikrokontrollerlar:

- O'rnatilgan tizimlarning asosiy boshqaruv elementlari.
- Protsessor, xotira va kirib-chiqish portlarini o'z ichiga oladi.
- Sensor va aktuatorlarni boshqarish uchun keng qo'llaniladi.

2. Sensor va aktuatorlar:

- Sensor - atrof-muhit holatini aniqlash uchun qurilmalar (harorat, bosim, tezlik, oqim va h.k.).
- Aktuator - tashqi dunyoga ta'sir o'tkazish uchun qurilmalar (dvigatel, klapan, yoritish va h.k.).
- Mikrokontrollerlar ular bilan o'zaro aloqada bo'ladi.

3. Interfeys shinalari:

- O'rnatilgan tizim komponentlari orasidagi aloqa uchun.
- Masalan, I²C, SPI, CAN, Ethernet, WiFi, Bluetooth va boshqalar.
- Tezlik, sinxronizatsiya, ko'p nuqtali aloqa kabi parametrlar.

4. Aloqa va uzatish protokollari:

- Qurilmalar orasidagi ma'lumotlarni uzatish.
- Masalan, ModBus, MQTT, OPC UA, BACnet va boshqalar.
- Ishonchlilik, xavfsizlik, real vaqt talablariga javob beradi.

5. Grafik interfeys va ekranlar:

- Foydalanuvchiga tizim holatini vizual ko'rsatish uchun.
- Mikrokontrollerlar yoki maxsus displey kontrollerlari bilan bog'lanadi.
- Sensorlardan olingan ma'lumotlar aks ettiriladi.

Ushbu vositalar yordamida o'rnatilgan tizim qurilmalari boshqaruv ob'ekti bilan ulanadi va o'zaro aloqada bo'ladi.

Bu vositalar va uskunalar, o'rnatilgan tizim qurilmalarini boshqarish va monitoring qilishni osonlashtiradi. Har bir tizim turiga moslashtirilgan maxsus vositalar mavjud bo'lishi mumkin.

57 WiFi yoki IEEE 802.11x standarti

WiFi (Wireless Fidelity) - bu IEEE 802.11x oilasiga kiruvchi, simsiz lokal tarmoq aloqasi standartidir. IEEE 802.11x standarti quyidagi asosiy versiyalardan iborat:

1. **IEEE 802.11 (1997):** Dastlabki 2 Mbit/s va 11 Mbit/s tezlik versiyasi.
2. **IEEE 802.11a (1999):** 5 GHz diapazonda maximum 54 Mbit/s tezlikni qo'llab-quvvatlaydi.
3. **IEEE 802.11b (1999):** 2.4 GHz diapazonda maximum 11 Mbit/s tezlikni qo'llab-quvvatlaydi.
4. **IEEE 802.11g (2003):** 2.4 GHz diapazonda maximum 54 Mbit/s tezlikni qo'llab-quvvatlaydi.
5. **IEEE 802.11n (2009):** 2.4 GHz yoki 5 GHz diapazonda maximum 600 Mbit/s tezlikni qo'llab-quvvatlaydi.
6. **IEEE 802.11ac (2013):** 5 GHz diapazonda maximum 1.3 Gbit/s tezlikni qo'llab-quvvatlaydi.
7. **IEEE 802.11ax (2019):** 2.4 GHz yoki 5 GHz diapazonda maximum 9.6 Gbit/s tezlikni qo'llab-quvvatlaydi (Wi-Fi 6 nomi bilan ham tanilgan).

WiFi standarti asosida ishlayotgan qurilmalar simsiz aloqa orqali ma'lumot uzata oladi. Ular yordamida Internet, lokal tarmoqlar va boshqa qurilmalar o'rtasida aloqa o'rnatish mumkin. Standarti yangiliklariga qarab tezlik, menzil, quvvat sarfi kabi parametrlari ham takomillashib borgan.

WiFi, kompyuter tarmoqlari va qurilmalari orasidagi radiosignallar orqali ma'lumotlarni o'rgatish va almashish uchun ishlatiladigan radiotexnologiyasi. Bu, ko'pgina mobil qurilmalar, smartfonlar, planshetlar, noutbuklar, stol kompyuterlar va boshqa qurilmalar bilan bog'liq tarmoq ulanish standarti.

WiFi (IEEE 802.11x) standarti asosiy xususiyatlari:

1. **Qurilmalar orasidagi Ishonchli Ma'lumot Almashtirish (WPA/WPA2):** WPA va WPA2 standartlari ma'lumotlar almashishni himoya qilish uchun ishlatiladi. Bu standartlar, istalgan WiFi tarmoqiga kirish uchun maxfiy kalitlarni kafolatlaydigan shifrlash protokollari sifatida ishlatiladi.
2. **Qisqa qarish va berish (QoS):** WiFi standarti, tarmoqdagi yo'nalishlar va ma'lumotlarni prioritetli yo'lda uzatish uchun QoS (Quality of Service) protokollari yordamida xizmat ko'rsatishni ta'minlaydi.
3. **Beytani bekatlilik (Interference Mitigation):** WiFi standarti, boshqa radiosignallar bilan shakllanadigan takomillashtirish va bog'liq qanday qiyinchiliklariga qarshi o'zaro tizimlarni ajratish uchun texnologiyalar qo'llaydi.

4. **Batareya yaxshiligi (Power Efficiency):** WiFi standarti, portativ qurilmalarni uzun vaqt ishlaydigan batareya yaxshiligiga ega bo'lish uchun xususiy ko'rsatkichlar taqdim etadi.
5. **Seksiya muxlati (Bandwidth Multiplexing):** WiFi standarti, ko'p sonli qurilmalar uchun bir xonadon qurilmalar orasidagi bandvichni muxlatishni ta'minlaydi.
6. **Yirik bog'lanish (Wide Coverage):** WiFi standarti, uzluksiz WiFi signalini uzluksiz ravishda uzatish uchun keng kapsulov qurilmalari yordamida yirik bog'lanishni ta'minlaydi.
7. **Avtomatik rejim (Auto Mode):** WiFi standarti, avtomatik rejimni ta'minlaydi, bu esa qurilmalarni uchdan besh dan ko'p sonli bog'lovchi rejimlar orasida avtomatik ravishda o'zgartirish imkonini beradi.
8. **Tarmoq asboblarning moslashishini ta'minlash:** WiFi standarti, kompyuterlar, printerlar, smartfonlar, planshetlar va boshqa qurilmalar orasida ko'p to'la aloqani ta'minlaydi.
9. **O'zaro bog'lovchi vositalarni qo'llash:** WiFi standarti, qurilmalarning boshqa qurilmalar bilan o'zaro bog'lanishini yaxshi ko'rib chiqishga imkon beradi. Masalan, printerlarga, bilim olish qurilmalariga, o'yin konsollariga, smart televizorlarga, musiqi o'ynatgichlarga va boshqalarga bog'liq tarmoqlarni yaratish uchun ishlatiladi.

WiFi standarti foydalanish maqsadlari:

- Ko'p sonli qurilmalar orasidagi ko'rsatkichlarni boshqarish.
- Internetga ulanishni osonlashtirish.
- Portativ qurilmalar uchun batareya yaxshiligini ta'minlash.
- Tarmoq asboblarning moslashishini ta'minlash.
- O'zaro bog'lovchi qurilmalar orasidagi o'zgaruvchilarni almashish.

Quyidagi standartlar bir nechta WiFi protocolini ifodalaydi:

1. **IEEE 802.11a:** 5 GHz bandida ishlaydigan qurilmalar uchun.
2. **IEEE 802.11b:** 2.4 GHz

S58. Bluetooth yoki IEEE 802.15.1 standarti

Bluetooth - bu IEEE 802.15.1 simsiz lokal tarmoq aloqa standartiga asoslangan texnologiya. Bluetooth qurilmalar orasida qisqa masofada ma'lumot almashinuvi, boshqaruv va sinxronizatsiya uchun mo'ljallangan.

IEEE 802.15.1 standarti quyidagi asosiy versiyalardan iborat:

1. **Bluetooth 1.0 va 1.0B (1999-2000):** Dastlabki versiyalar, 1 Mbit/s tezlik.
2. **Bluetooth 2.0 + EDR (2004):** Tezligi 3 Mbit/s ga yetadi, energiya sarfi kamayradi.
3. **Bluetooth 3.0 + HS (2009):** Tezligi 24 Mbit/s ga yetadi, AES shifrlanishi qo'llaniladi.
4. **Bluetooth 4.0 (2010):** Yangi qurilmalar uchun optimal, energiya sarfi juda past, 1 Mbit/s tezlik.
5. **Bluetooth 5.0 (2016):** Menzili 4 barobar oshgan, tezligi 2 Mbit/s, energiya sarfi kichik.
6. **Bluetooth 5.1 (2019):** Lokatsiya aniqlash imkoniyatlari yaxshilangan, energiya sarfi kamaygan.
7. **Bluetooth 5.2 (2020):** LE Audio texnologiyasi qo'llab-quvvatlanadi, ishlash oralig'i uzaygan.

Bluetooth qurilmalar orasidagi aloqani mavjud tarmoq infratuzilmasiga bog'liq bo'lmagan holda ta'minlaydi. Ular kichik masofadagi ma'lumot uzatish va boshqaruv maqsadlarida keng qo'llaniladi. Yangi versiyalar bilan tezlik, energiya samaradorligi, xavfsizlik kabi xususiyatlari doimo takomillashib boradi.

Bluetooth, qisqa masofaviy radio aloqa standarti, qurilmalar orasidagi ma'lumot almashish va ulanishni ta'minlaydi. U, qurilmalar orasidagi ishoratni almashish, qurilmalarni bog'lash, ko'rsatkichlarni almashish va tizimlar orasidagi ma'lumot almashishni osonlashtiradi.

Bluetooth (IEEE 802.15.1) standarti asosiy xususiyatlari:

1. **Qisqa masofaviy aloqa:** Bluetooth, qisqa masofaviy orqali (odamlar yoki qurilmalar orasida 10 metrga qadar) ma'lumot almashish va ulanishni ta'minlaydi.
2. **Eng yirik ta'minot xizmatlari:** Bluetooth, qurilmalar orasidagi ma'lumot almashish va ulanishning ko'p ta'minot xizmatlarini (masalan, talqin etuvchilar, printerlar, klaviaturalar, musiqi qurilmalari va boshqalar) ta'minlaydi.
3. **Uzluksiz foydalanish:** Bluetooth, qurilmalar orasidagi uzluksiz aloqani ta'minlash uchun moslashtirilgan protokollardan foydalanadi.
4. **Batareya uchun samarali:** Bluetooth, qurilmalar orasidagi aloqani ochildi va batareya yaxshiligini ta'minlash maqsadida samarali foydalanishni ta'minlaydi.

5. **To'g'ri bog'lanish uchun qulaylik:** Bluetooth, bog'liq qurilmalar uchun avtomatik ravishda qo'llanishga imkon beradi, shuningdek, bog'lovchi va qabul qiluvchi qurilmalar orasidagi o'zgaruvchilar ko'rib chiqishni yaxshi o'rganadi.
6. **Yuqori xavfsizlik darajasi:** Bluetooth standarti, ma'lumotlar almashishni himoya qilish uchun yuqori darajada xavfsizlikni ta'minlaydi.
7. **Avtomatik o'rnatish va boshqarish:** Bluetooth, qurilmalar orasidagi aloqani ochildi va avtomatik ravishda o'rnatish va boshqarish uchun moslashtirilgan protokollardan foydalanadi.
8. **Qisqa foydalanish protokollari:** Bluetooth, qurilmalar orasidagi ma'lumot almashish va ulanishni osonlashtirish uchun qisqa foydalanish protokollari bilan ta'minlaydi.

Bluetooth standarti foydalanish maqsadlari:

- Qisqa masofaviy (10 metrgacha) aloqalarni ta'minlash.
- Uzluksiz bog'liq qurilmalar uchun qulaylik yaratish.
- Ko'p sonli ta'minot xizmatlarini ta'minlash.
- Batareya yaxshiligini ta'minlash.

Bluetooth standarti turli xil protokollarga ega:

1. **Bluetooth Classic:** Klassik Bluetooth, qisqa masofaviy (10 metrgacha) aloqalar uchun moslashtirilgan standart Bluetooth versiyasi.
2. **Bluetooth Low Energy (BLE):** Qisqa batareya hayotini qo'llaydigan, kam quvvat sarf etuvchi Bluetooth protocoli.
3. **Bluetooth Mesh:** Ko'pta'minli IoT (Internet of Things) tizimlarini birlashtirish uchun BLE (Bluetooth Low Energy) asosida yaratilgan jadvaldagi aloqa protokol.

Quyidagi standartlar bir nechta Bluetooth protokollarni ifodalaydi:

- **IEEE 802.15.1-2002:** Asosiy Bluetooth standarti.
- **IEEE 802.15.1-2005:** Bluetooth versiyasi 2.0 ni joriy etadi.
- **IEEE 802.15.1-2017:** Ommaviy yordamchi profillarini (Public Access Profiles) qo'shdi

S59 Mikrokontrollerni tashkillashtirish.

Mikrokontrollerlar o'rnatilgan tizimlar asosidagi apparatli-dasturiy qurilmalarning asosiy boshqaruv elementi hisoblanadi. Mikrokontrollerni tashkillashtirish uchun quyidagi komponentlar zarur:

1. **Protsessor:** Mikrokontrollerdagi asosiy hisoblash qurilmasi. Odatda 8, 16 yoki 32 bitli RISC arxitekturasiga ega.
2. **RAM xotira:** Dastur va o'zgaruvchilar uchun ishlatiladi, mikrokontrollerdagi protsessor tomonidan to'g'ridan-to'g'ri kiritiladigan.
3. **ROM yoki Flash xotira:** Doimiy dastur kodini saqlaydi, protsessor tomonidan o'qiladigan.
4. **Kirib-chiqish portlari:** Tashqi qurilmalar va sensorlar bilan aloqa o'rnatish uchun.
5. **Taqvimlovchi generators:** Protsessor ichki taktirovka uchun, shuningdek tashqi qurilmalar uchun.
6. **Analog-raqamli va raqamli-analog konvertorlar:** Sensor signallarini raqamli ko'rinishga aylantirish uchun.
7. **Interfeys shinalari:** I²C, SPI, UART kabi protsessor va qurilmalar orasidagi aloqa.
8. **Kuchirishtirgichlar:** Sensor yoki aktuatorlarni boshqarish uchun kuchli signallar yaratadi.
9. **Qo'shimcha modullar:** Vaqt, xavfsizlik, energiya boshqaruvi kabi maxsus funktsiyalarni amalga oshirish uchun.

Mikrokontrollerlar bu komponentlarni bir yig'ma sxemada birlashtirib, o'rnatilgan tizimlarning asosiy "miya"si vazifasini bajaradi. Dasturiy ta'minot yordamida ularning ishlashi tartiblashtiriladi va o'zgaruvchan vazifalar bajariladi.

Mikrokontrollerlar tashkillashtirish odatda quyidagi bosqichlardan o'tkaziladi:

1. Dasturlash Muhitini Tuzing:

Mikrokontrollerlarni tashkillashtirishdan oldin, dasturlash muhitini o'rnatish zarur. Ushbu muhit, mikrokontrollerga kod yozish va uningga kiritishni osonlashtiradi. Odatda, dasturlash uchun quyidagi vositalar ishlatiladi:

- **Dasturlash Tillari va Asboblari Kitobxonasi (IDE):** Bir nechta dasturlash tillarini (masalan, C, C++, Assembly) qo'llash mumkin. IDE, dasturlash jarayonini osonlashtiradi va dasturlarni yozish, tekshirish va yuklash imkonini beradi. Mashhur IDE lar Visual Studio Code, Arduino IDE, MPLAB X IDE va Keil uchun IDE ni misol qilsak bo'ladi.
- **Tasvirlik Dastur Qo'llanmalari:** Bu dasturlar, grafik interfeys orqali elektronika loyihalarini tashkillashtirishni osonlashtiradi. Misol uchun, Proteus,

KiCad, Eagle va Altium Designer tasvirlik dasturlar orasida keng tarqalgan dasturlar.

2. Mikrokontrollerni Tanlash va Ilovani Tanlang:

Mikrokontrollerlar juda turli xil xususiyatlarga ega. Ilovani to'g'ri tanlash uchun kerakli funksiyalarni, xususiyatlarini va mahsulot ko'rsatuvlarini baholashingiz kerak bo'ladi.

- **Funksiyalarni Baholash:** Kerakli portlar, to'lov bollari, mahsulotning o'lchamlari va boshqa xususiyatlar bo'yicha kerakli funksiyalarni baholang.
- **Birinchi o'lcham:** Qanchalik qisqa vaqt davomida dastur tuzish uchun, qanday qismlarni tayyorlash kerakligini aniqlang.

3. Qurilishni Tashkil Etish:

Mikrokontrollerlarni tashkillashtirish, loyiha uchun kerakli elektronik qurilmani tashkil etishni o'z ichiga oladi. Bu bosqichda quyidagi vazifalarni bajarish kerak:

- **Taqsimlash va Joylashtirish:** Dastlabki loyihani qanchalik qisqa bo'lishi, kerakli komponentlarni joylashuvini belgilang.
- **Komponentlarni Joylashtirish:** Mikrokontroller, sensorlar, portlar, tizimlar, kuch ta'minoti boshqa elektronik qurilmalar bilan birgalikda joylashtiriladi.
- **Boshqa loyihaga to'g'ri ishlovchi qurilmalarni tanlash:** To'g'ri komponentlarni tanlash uchun, loyihani to'liq tushunarli va ishlashini ta'minlash kerak.

4. Dastur Tuzish va Tekshirish:

Mikrokontrollerga dastur yozish, uni yuklash va tekshirish muhim qismi hisoblanadi. Bu bosqich quyidagi vazifalardan iborat:

- **Dastur Tuzish:** Tanlangan dasturlash tilida, kerakli funksiyalarni dasturlaydigan va loyihada amalga oshirilishiga imkon beradigan dasturlar tuziladi.
- **Dasturni Yuklash:** Dastur, tanlangan mikrokontrollerga yuklanadi.
- **Tekshirish va To'liqlash:** Mikrokontrollerga yuklangan dasturni tekshirish uchun moslashtirilgan protokollar va vositalar ishlatiladi. Qurilishni sinash, ma'lumotlarni tekshirish va dastur bajarishni o'z ichiga oladi.

5. Jarayonni Boshqarish:

Mikrokontroller loyihalari jarayonini boshqarish, u ishga tushirilgandan so'ng dasturni bajarish va boshqa dasturlar bilan ma'lumot almashishni o'z ichiga oladi. Bu bosqichda quyidagi vazifalar o'tkaziladi:

- **Jarayonni Boshqarish Protokollari:** Ilovani ishlatish uchun ma'lumot almashish va bog'lanishning protokollarini va algoritmlarini aniqlang.
- **Ma'lumotlar O'qish va Yozish:** Mikrokontroller bilan bog'liq sensorlar, portlar va boshqa qurilmalar bilan ma'lumot almash va uxlashning barcha protokollarni o'z ichiga oladi.
- **Avtomatik Rejimlar:** Mikrokontrollerning avtomatik rejimlarni aniqlash va amalga oshirish imkonini ta'minlash.
- **Xavfsizlik:** Ma'lumotlar o'z ichiga olindi va shifrlangan yoki boshqa xavfsizlik protokollari bilan himoyalangan.

Mikrokontrollerlarni tashkillashtirishni boshlashdan avval, har bir bosqichni diqqat bilan o'tkazish, dasturlash tillari va elektronika sohasida tajribaga ega bo'lish juda muhimdir.

S60 Axborotlarga parallel ishlov berish.

Axborotlarga parallel ishlov berish, bir nechta vazifalarni yuzlab yoki minglab mikrokontrollerlarda bajarish imkonini beradi. Bu, barcha vazifalarni bir vaqtning o'zida parallel ravishda bajarishni ta'minlaydi va jarayonni tejash va tizimni samarali ishga solishni ta'minlaydi. Quyidagi bosqichlar, axborotlarga parallel ishlov berish jarayonini o'tkazishda kerak bo'lgan amaliyatlardir:

1. Vazifalarni Tanlash va Taqsimlash:

- **Vazifalarni Tahlil qilish:** Qilinadigan har bir vazifa, jarayonlarning ta'minoti, tekshirish, ma'lumotlarni amalga oshirish va boshqa amaliyotlar bo'yicha tahlil qilinadi.
- **Vazifalarni Taqsimlash:** Asosiy vazifani bir nechta subvazifalarga ajratish, har bir subvazifani alohida o'rnatish va uni bajarish uchun ma'lum bir qurilmaga (mikrokontrollerga) topshirish.

2. Ma'lumot Almashtirish va O'xish Tizimlarini Tuzing:

- **Komunikatsiya Interfeyslarini Tanlash:** Kerakli kommunikatsiya interfeyslarini (masalan, UART, SPI, I2C) tanlash.
- **Protokollarni Tuzish:** Ma'lumot almashishni amalga oshirish uchun moslashtirilgan protokollarni tanlash va o'rnatish.

3. Vazifalarni Bajarish va Parallel ishlov berish:

- **Mikrokontrollerlarni tanlash va tashkil etish:** Har bir vazifani bajarish uchun moslashtirilgan mikrokontrollerlarni tanlash va tashkil etish.
- **Vazifalarni Parallel Ishlov Berish:** Har bir mikrokontrollerda bir vazifani bajarish.
- **Ma'lumot Almashtirish va O'xish:** Ma'lumotlar o'qish va yozish uchun moslashtirilgan kommunikatsiya interfeyslaridan foydalanish.

4. Ma'lumotlarni Yig'ish va Natijalarni Ishlovchi Boshqaruv Markazi Bilan Ulash:

- **Ma'lumotlarni Yig'ish:** Har bir mikrokontroller tomonidan bajarilgan vazifalar natijalari, boshqaruv markazi bilan ulash uchun bir ma'lumot to'plamiga yig'iladi.
- **Natijalarni Ishlovchi Boshqaruv Markazi:** Barcha ma'lumotlarni qabul qilish, tahlil qilish va moslashtirilgan boshqaruv markaziga natijalarni yetkazish.

5. Jarayonni Tekshirish va Optimallashtirish:

- **Vazifalarni Tekshirish:** Parallel ishlov berishni tekshirish va natijalarni ma'lum etish.
- **Ishlov berishni Optimallashtirish:** Har bir mikrokontrollerdagi vazifalarni parallel ishlov berishni optimallashtirish, ma'lumot almashish va boshqarish davomiyligi orqali.

6. Xavfsizlik:

- **Ma'lumotlar Xavfsizligi:** Ma'lumotlar almashish va yozish jarayonlarida xavfsizlikni ta'minlash uchun moslashtirilgan xavfsizlik protokollaridan foydalanish.

Misol:

Agar siz bir mavjud tarmoq loyihangizning barcha komponentlarini boshqarish uchun parallel ishlov berishni o'rganmoqchisiz, quyidagi bosqichlar sizga yordam beradi:

1. **Vazifalarni Taqsimlash:** Har bir komponentni boshqarish uchun asosiy vazifani ta'min etish. Masalan, sensorlarni boshqarish, ma'lumotlarni saqlash, qurilmalarni boshqarish va h.k.
2. **Mikrokontroller Tanlash:** Har bir vazifa uchun moslashtirilgan mikrokontroller tanlash.
3. **Birga Tegishli Ma'lumot Almashtirish Protokollari Tuzish:** Moslashtirilgan kommunikatsiya protokollari (masalan, I2C, SPI) orqali barcha komponentlar bilan bog'lanishni ta'minlash.
4. **Ma'lumotlarni Boshqarish:** Barcha sensorlardan olingan ma'lumotlarni tekshirish, tahlil qilish va moslashtirilgan boshqaruv markazi bilan ulash uchun xavfsizlikni ta'minlash.
5. **Parallel Ishlov Berish:** Har bir mikrokontrollerda boshqariladigan vazifani o'rnatish.
6. **Natijalar va Ma'lumotlar Bilan Ishlov Berish:** Har bir mikrokontrollerdan olingan natijalar boshqaruv markazi bilan ulanadi va boshqaruv markazi natijalarni tekshiradi, tahlil qiladi va boshqarish bilan davomiyligi natijalarni yuboradi.

Ushbu qadamli loyihalar, komponentlarni parallel ishlov berish uchun mo'ljallangan loyihani tuzishda yordam beradi.

Parallel ishlov berish - bu bir vaqtning o'zida bir nechta jarayonning bir xil vaqt oralig'ida bajarilishidir. Bunda axborot bir necha mustaqil bir-biridan ajralgan yo'nalishlar bo'yicha qayta ishlangan bo'ladi.

Axborotlarga parallel ishlov berish quyidagi asosiy usullar orqali amalga oshiriladi:

1. **Konveyer (konveyer qatori) usuli:** Bir jarayonning chiqish signaliga darhol keyingi jarayonning kirish signali ulangan bo'ladi. Bu usul maxsus konveyer qurilmalar yordamida amalga oshiriladi.
2. **Tarmoq usuli:** Bir-biriga o'xshash jarayonlar bir vaqtning o'zida bajariladi, ular o'rtasida ma'lumotlar almashinuvi amalga oshiriladi.
3. **SIMD (bir xil instruktsiyalar - ko'p ma'lumotlar) usuli:** Bir xil instruktsiyalarni bir vaqtda ko'p ma'lumotlarga nisbatan qo'llash orqali parallel ishlov berish.
4. **MIMD (ko'p instruktsiyalar - ko'p ma'lumotlar) usuli:** Har xil instruktsiyalarni bir vaqtda ko'p ma'lumotlarga nisbatan qo'llash orqali parallel ishlov berish.

Parallel ishlov berish:

- Hisoblash samaradorligini oshiradi
- Katta hajmdagi ma'lumotlarni tezroq qayta ishlashga imkon beradi
- Eng murakkab masalalarni yechishda qo'l keladi (kriptografiya, molekulyar biologiya, fizika va h.k.)

Zamonaviy kompyuter va mikrokontroller arxitekturalari parallel ishlov berishni qo'llab-quvvatlaydi. Bu ularning hisoblash qobiliyatlarini sezilarli darajada oshiradi.

S61.Ma'lumotlarni parallel qayta ishlash.

Ma'lumotlarni parallel qayta ishlash - bu bir vaqtning o'zida bir nechta mustaqil jarayonlar orqali ma'lumotlarni qayta ishlash jarayonidir. Bunda ma'lumotlar guruhlanib, har bir guruh alohida protsessor yoki yadro tomonidan qayta ishlangan bo'ladi.

Asosiy parallel ma'lumot qayta ishlash usullari quyidagilar:

1. **SIMD (Single Instruction, Multiple Data)**: Bir xil instruksiyalar bir vaqtda ko'p ma'lumotlarga qo'llaniladi. Masalan, yuqori samarali grafik protsessorlar (GPU) va vektorli protsessorlar.
2. **MIMD (Multiple Instruction, Multiple Data)**: Har xil instruksiyalar bir vaqtda ko'p ma'lumotlarga qo'llaniladi. Asosan ko'p yadrolik protsessorlarda qo'llaniladi.
3. **Konveyer arxitekturas**i: Ma'lumotlar qayta ishlash jarayoni ketma-ket bosqichlarga bo'lingan bo'lib, har bir bosqich bir vaqtda bajariladi.
4. **Tarmoq arxitekturas**i: Ma'lumotlar bir-biridan mustaqil bo'lgan parallel jarayonlar orasida taqsimlanadi.
5. **GPGPU (General-Purpose computing on Graphics Processing Units)**: GPU yadrolari orqali umumiy maqsadlar uchun parallel ma'lumotlarni qayta ishlash.

Parallel ma'lumot qayta ishlash samaradorligi quyidagi omillarga bog'liq:

- Jarayonlar o'rtasidagi ma'lumot almashish tezligi
- Parallel jarayonlar soni
- Ular o'rtasidagi mos kelish (sinkronizatsiya)
- Yuk taqsimotining muvozanatlanganligi

Zamonaviy mikrokontrollerlar, protsessorlar va GPU-lar parallel qayta ishlashni qo'llab-quvvatlaydi, bu ularning hisoblash qobiliyatini sezilarli oshiradi.

Ma'lumotlarni parallel qayta ishlash, bir nechta parallel vazifalarni o'z ichiga oladi va ularni bir-biriga bog'liq qiladi, shuningdek, ma'lumotlar orasida o'zaro aloqani optimallashtiradi. Bu jarayon, boshqa o'tgan ma'lumotlarni ishlov berish va natijalarni to'plam bo'lib, ularni boshqa amalga oshirish, tekshirish va boshqarish uchun moslashtirilgan protokollar va algoritmlar yordamida amalga oshiriladi. Quyidagi bosqichlar, ma'lumotlarni parallel qayta ishlash jarayonida kerak bo'lgan asosiy amaliyatlarni ta'minlaydi:

1. Ma'lumotlarni Paralel Ishlov Berish va Qayta Ishlash Uchun Taqsimlash:

- **Ma'lumotlar Tahlil qilish**: Paralel ishlov berish jarayonini boshlash uchun bajarilishi kerakli vazifalarni aniqlash va ma'lumotlar tahlilini o'tkazish.
- **Ma'lumotlar Taqsimi**: Paralel ishlov berish uchun, ma'lumotlarni ajratib chiqish va ularni har bir parallel vazifaga topshirish uchun moslashtirilgan algoritmlar va protokollar ishlatiladi.

2. Paralel Ishlov Berishning Tizimi va Infrastrukturini Tuzish:

- **Paralel Vazifalar Tuzilishi:** Har bir vazifani o'zlashtirish uchun to'plamni tuzish, har bir vazifaga alohida bo'lmish parallel ishlov berish tizimini tuzish.
- **Infrastruktura Tuzish:** Vazifalar orasidagi ma'lumot almashish va o'xish uchun zarur protokollar va aloqalar tuziladi. Bu aloqalar, parallel ishlov berishning hamda ma'lumotlarni to'plab olish va ularga murojaat qilish jarayonlarini boshqarishga imkon beradi.

3. Ma'lumotlarni Paralel Ishlov Berish:

- **Bir nechta vazifalarni parallel bajarish:** Ma'lumotlarni parallel qayta ishlash jarayonida, boshqa ma'lumotlar bilan bog'liq qilinishi kerak bo'lgan bir nechta parallel vazifalar o'zlashtiriladi.
- **Tahlil va Ishlov Berish:** Har bir parallel vazifada, ma'lumotlarni qayta ishlash uchun moslashtirilgan algoritmlar ishlatiladi.

4. Natijalarni To'plash va Boshqarish:

- **Natijalarni To'plash:** Parallel vazifalar o'zlarining natijalarini topish uchun moslashtirilgan boshqaruv markaziga natijalarni to'plash.
- **Natijalarni Tahlil va Ishlov Berish:** Natijalar boshqaruv markaziga topshiriladi, tekshiriladi va ma'lumotlar qayta ishlash jarayonida kerakli bo'lgan ishlov berish jarayonlari o'tkaziladi.

5. Xavfsizlik:

- **Xavfsizlikni Ta'minlash:** Ma'lumotlarni parallel qayta ishlash jarayonida, ma'lumotlarning xavfsizligi juda muhimdir. Shuning uchun ma'lumotlarni himoyalash uchun shifrlash va boshqa xavfsizlik protokollari qo'llaniladi.

6. Ma'lumotlarni To'plab Qo'yish:

- **Ma'lumotlar To'plami:** Har bir vazifa uchun topshirilgan ma'lumotlar boshqaruv markaziga to'plab qo'yiladi.

Misol:

Agar bir tarmoq loyihangizning barcha sensorlaridan ma'lumotlarni to'plash va ularga boshqa amalga oshirish uchun parallel ishlov berishni o'rganmoqchisiz, quyidagi bosqichlar sizga yordam beradi:

1. **Ma'lumotlarni Paralel Ishlov Berish:** Har bir sensorning ma'lumotlarini parallel ravishda to'plash uchun paralel vazifalarni o'zlashtirish.
2. **Parallel Vazifalar Tuzilishi:** Har bir sensor uchun parallel vazifalarni tuzish va ma'lumotlarni to'plash.
3. **Infrastruktura Tuzish:** Har bir sensor uchun moslashtirilgan kommunikatsiya protokollarini va aloqalarni tuzish.
4. **Ma'lumotlarni Ishlov Berish:** Har bir parallel vazifa uchun ma'lumotlarni to'plash, tekshirish va ma'lumotlarni boshqaruv markaziga topshirish.
5. **Natijalar va Ma'lumotlar Bilan Ishlov Berish:** Har bir parallel vazifa natijalarini boshqaruv markaziga topshirish.

Ushbu bosqichlar, ma'lumotlarni parallel qayta ishlash uchun kerak bo'lgan asosiy amaliyatlarni ta'minlaydi va parallel ishlov berishning samarali va samarali qo'llanilishini ta'minlaydi.

S62.O'rnatilgan tizimlarga uzilish.

O'rnatilgan tizimlarga uzilish (interruption) - bu tizim ishlayotgan vaqtda ichki yoki tashqi omillar tomonidan yasalgan "to'satdan" yuzaga kelgan voqea bo'lib, uni darhol e'tiborga olish va zarur harakatlar qilish talab etiladi.

Uzilishlar quyidagi asosiy turlarga bo'linadi:

1. **Tashqi uzilishlar:** Tashqi sensorlar, interfeys portlari, taymerlar va boshqa tashqi qurilmalarning signallari orqali yuzaga keladi.
2. **Ichki uzilishlar:** Protsessor, xotira, timing va boshqa mikroskopik komponentlarning ichki holatlariga bog'liq ravishda yuzaga keladi.
3. **Dasturiy uzilishlar:** Unga qo'shimcha ravishda dasturiy ta'minotda belgilangan shartlar asosida yasaladi.

Uzilishlar tizim ishlashiga qiyinlik tug'dirmasligi uchun quyidagi talablarga javob berishi kerak:

- **Eng qisqa muddatda bajarilishi:** Uzilishni qayta ishlash dasturi tezkor bajarilishi lozim.
- **Minimal resurs sarfi:** Uzilishni qayta ishlash minimal protsessor vaqi, xotira hajmi va boshqa resurslarni talab qilishi kerak.
- **Tizim ishlashiga ta'sir qilmasligi:** Uzilish tizimning asosiy ishini buzmasligi kerak.

O'rnatilgan tizimlarda uzilishlar muhim ahamiyatga ega. Ular tizimning real vaqtda ishlashiga, optimizatsiya qilishga va keng imkoniyatlar yaratishga imkon beradi.

O'rnatilgan tizimlarga uzilish, o'rnatilgan operatsion tizimning boshqa tizimlar yoki qurilmalar bilan integratsiyasini anglatadi. Bu jarayon, o'rnatilgan tizimning o'z funksiyalarini bajarish va boshqa tizimlar bilan ma'lumot almashish, ulashish va boshqa interaktsiyalarni o'z ichiga oladi. Quyidagi bosqichlar, o'rnatilgan tizimlarga uzilish jarayonida kerak bo'lgan asosiy amaliyatlarni ta'minlaydi:

1. Tizim Uchun Kerakli Interfeyslarni Tuzish:

- **Aloqa Interfeysi Tuzish:** O'rnatilgan tizimni boshqa tizimlar bilan aloqada bo'lish uchun kerakli aloqa protokollarini (masalan, UART, SPI, I2C) tuzish.
- **Aloqa Uzatish:** Kerakli aloqa interfeyslarini tashkil etish va boshqa tizimlar bilan aloqa o'rnatish.

2. Tizimni Sozlash va Integrlash:

- **Protokollar Tuzilishi:** Ma'lumot almashishni amalga oshirish uchun kerakli protokollarni tuzish, shunday qilib, ma'lumotlarni to'g'ridan-to'g'ri almashish.
- **Parametrlarni Sozlash:** Boshqa tizimlar bilan ma'lumot almashish uchun kerakli parametrlarni sozlash.

3. Uzilayotgan Tizimni Testlash:

- **Testlash va Tekshirish:** O'rnatilgan tizimlarga uzilishdan so'ng, integratsiya jarayonini tekshirish va testlash.
- **To'liqlash va Optimallashtirish:** O'rnatilgan tizimning to'g'ri ishlashi va samarali ishlovchi bo'lishi uchun parametrlarni to'liqlash va optimallashtirish.

4. Tizim Uzatilishi va Qo'llanilishi:

- **O'rnatilgan Tizimlarni Ishga Solish:** O'rnatilgan tizimlarni ishga solish va ularning so'nggi versiyalarini o'rnatish.
- **Boshqa Tizimlar bilan Integratsiya qilish:** O'rnatilgan tizimlarni boshqa tizimlar bilan integratsiya qilish.

5. Xavfsizlik:

- **Xavfsizlikni Ta'minlash:** Ma'lumotlar o'tkazish va almashish jarayonlarida xavfsizlikni ta'minlash.

Misol:

Agar bir tarmoq loyihangizga yangi o'rnatilgan xizmat qo'shishni rejalashtirayotgan bo'lsangiz, quyidagi bosqichlar sizga yordam beradi:

1. **Tizim Uchun Aloqa Interfeyslarni Tuzing:** Aloqa protokollarini (masalan, TCP/IP, HTTP) tuzish va tarmoq loyihangizga bog'liq boshqa tizimlar bilan aloqani o'rnatish.
2. **Protokollar Tuzilishi:** Ma'lumot almashishni boshqa tizimlar bilan amalga oshirish uchun kerakli protokollarni tuzish. Masalan, RESTful API yoki SOAP protokollari bilan integratsiya qilish.
3. **Testlash va Tekshirish:** Yangi o'rnatilgan xizmatni tekshirish va testlash, shu jumladan, ma'lumotlarni to'g'ridan-to'g'ri almashishni va qayta ishlashni tekshirish.
4. **Integratsiya va Testlash:** Yangi xizmatni boshqa tizimlar bilan integratsiya qilish va uning to'g'ri ishlashini test qilish.
5. **Xavfsizlikni Ta'minlash:** Ma'lumotlar o'tkazish va almashish jarayonlarida xavfsizlikni ta'minlash.

O'rnatilgan tizimlarga uzilish jarayonida amaliyotlarning har biri, loyihangizning maqsadlariga mos ravishda bajarilishi kerak. Ushbu bosqichlar, yangi xizmatlarni tarmoq loyihangizga muvaffaqiyatli qo'shish uchun kerakli bo'lgan asosiy amaliyatlarni ta'minlaydi

S63.Xotiraga bevosita ega bo'lish.

Xotiraga bevosita ega bo'lish - bu protsessor yoki mikrokonrollarning xotira manzillarini to'g'ridan-to'g'ri o'qish va yozish jarayonidir. Bu protsess xotira bilan to'g'ridan-to'g'ri muloqot qilish imkonini beradi.

Xotiraga bevosita ega bo'lish asosiy usullari:

1. **Birma-bir manzillash:** Protsessor xotira manzillarini to'g'ridan-to'g'ri ko'rsatadi va ma'lumotlarni o'qiydi yoki yozadi.
2. **Indekslangan manzillash:** Protsessor indekslangan xotira joylaridan ma'lumotlarni o'qiydi yoki yozadi.
3. **Qurilmalar interfeysi orqali manzillash:** Protsessor boshqa qurilmalar yordamida (masalan, DMA) xotiraga kirishadi.

Xotiraga bevosita ega bo'lishning asosiy afzalliklari:

- **Tezkor o'qish/yozish:** Xotira bilan to'g'ridan-to'g'ri muloqot qilinishi natijasida ma'lumotlar almashinuvi juda tez amalga oshiriladi.
- **Resurslardan to'liq foydalanish:** Protsessor xotira manzillarini to'g'ridan-to'g'ri boshqarishi natijasida resurslardan unumli foydalanish ta'minlanadi.
- **Minimal kechikish:** Xotira bilan muloqot uchun minimal kechikish yuzaga keladi.

Xotiraga bevosita ega bo'lish o'rnatilgan tizimlarda, real vaqt tizimlari, signal qayta ishlash tizimlari va boshqa ehtiyojlari yuqori bo'lgan tizimlarda qo'llaniladi.

Xotiraga bevosita ega bo'lish, operatsion tizimdagi ma'lumotlar bilan ishlashning asosiy tizimlari haqida. Bu, ma'lumotlarni yozish, o'qish va o'chirish jarayonlarini boshqarish uchun kerakli tizimlarni ta'minlashni anglatadi. Quyidagi bosqichlar, xotiraga bevosita ega bo'lish jarayonida kerak bo'lgan asosiy amaliyatlarni ta'minlaydi:

1. Ma'lumotlar Yozish va O'qish Interfeysi:

- **Ma'lumotlar O'qish va Yozish Qayta Ishlovini Boshqarish:** Xotiraga ma'lumot yozish, o'qish va o'chirish uchun kerakli interfeyslar va funksiyalarni ta'minlash.
- **Fayl Ochildi Boshqaruvchi:** Fayl ochilgan, yozilgan va yopilgan bo'lsa, bu protsessni boshqarish.

2. Ma'lumotlar Strukturalash va Yig'ish:

- **Ma'lumotlar Tizimini Tashkil Qilish:** Ma'lumotlarni organizatsiyalash va jadval, ro'yxat yoki boshqa tizimlarni ishlatish.
- **Ma'lumotlar Strukturalash:** Ma'lumotlarni strukturatseya qilish uchun kerakli tizimlarni ta'minlash, masalan, o'lchovlar, satrlar va jadvallar.

3. Ma'lumotlar Qayta Ishlov Berish va Taxminiy Ishlovlar:

- **Fayl Tizimi:** Ma'lumotlar o'qilgan yoki yozilgan bo'lsa, ularni yuklab olish, taxminiy ishlovlar va taxminiy qayta ishlovlar.
- **Ma'lumotlar Qayta Ishlovlarini Tashkil Etish:** Taxminiy ishlovlar va qayta ishlovlar uchun kerakli tizimlarni ta'minlash.

4. Ma'lumotlarni Boshqarish va Qayta Ishlov Berish:

- **Ma'lumotlarni Qayta Ishlov Berish va Taxminiy Ishlovlarini Boshqarish:** Taxminiy ishlovlar va qayta ishlovlar bo'yicha ma'lumotlarni to'plash va boshqarish.
- **Ma'lumotlar Uchun Boshqaruv Markazi:** Barcha ma'lumotlar uchun boshqaruv markazini tashkil etish va unga kirish.

Misol:

Agar siz yangi fayllarni yuklash, ularni qayta ishlab chiqish va qayta ishlov berishning taxminiy ma'lumotlarini boshqarishni rejalashtirayotgan bo'lsangiz, quyidagi bosqichlar sizga yordam beradi:

1. **Ma'lumotlar O'qish va Yozish Interfeysi:** Fayllarni o'qish va yozish uchun kerakli interfeyslarni ta'minlash, masalan, fayl ochish, yozish va yopish.
2. **Fayl Tizimi:** Fayllarni taxminiy qayta ishlovlar uchun yuklab olish va boshqarish.
3. **Ma'lumotlar Qayta Ishlovlarini Boshqarish:** Taxminiy qayta ishlovlar uchun ma'lumotlarni boshqarish va ularga murojaat.
4. **Ma'lumotlar Uchun Boshqaruv Markazi:** Barcha ma'lumotlar uchun boshqaruv markazini tuzish va unga kirish.

Ushbu bosqichlar, ma'lumotlar bilan ishlashning asosiy tizimlarini o'rnatishni va boshqarishni ta'minlaydi. Bu jarayon, operatsion tizimning ma'lumotlar bilan ishlash bo'yicha asosiy funksiyalarini ta'minlash uchun muhimdir.

S64.Oʻrnatilgan tizimlarning kiritish-chiqarish interfeyslari

Xotiraga bevosita ega bo'lish - bu protsessor yoki mikrokonrollarning xotira manzillarini to'g'ridan-to'g'ri o'qish va yozish jarayonidir. Bu protsess xotira bilan to'g'ridan-to'g'ri muloqot qilish imkonini beradi.

Xotiraga bevosita ega bo'lish asosiy usullari:

1. **Birma-bir manzillash:** Protsessor xotira manzillarini to'g'ridan-to'g'ri ko'rsatadi va ma'lumotlarni o'qiydi yoki yozadi.
2. **Indekslangan manzillash:** Protsessor indekslangan xotira joylaridan ma'lumotlarni o'qiydi yoki yozadi.
3. **Qurilmalar interfeysi orqali manzillash:** Protsessor boshqa qurilmalar yordamida (masalan, DMA) xotiraga kirishadi.

Xotiraga bevosita ega bo'lishning asosiy afzalliklari:

- **Tezkor o'qish/yozish:** Xotira bilan to'g'ridan-to'g'ri muloqot qilinishi natijasida ma'lumotlar almashinuvi juda tez amalga oshiriladi.
- **Resurslardan to'liq foydalanish:** Protsessor xotira manzillarini to'g'ridan-to'g'ri boshqarishi natijasida resurslardan unumli foydalanish ta'minlanadi.
- **Minimal kechikish:** Xotira bilan muloqot uchun minimal kechikish yuzaga keladi.

Xotiraga bevosita ega bo'lish o'rnatilgan tizimlarda, real vaqt tizimlari, signal qayta ishlash tizimlari va boshqa ehtiyojlari yuqori bo'lgan tizimlarda qo'llaniladi.

64 Oʻrnatilgan tizimlarning kiritish-chiqarish interfeyslari

O'rnatilgan tizimlarning kiritish-chiqarish interfeyslari

O'rnatilgan tizimlar kiritish-chiqarish (input-output, I/O) interfeyslari - bu tizimning tashqi qurilmalar bilan aloqa o'rnatish vositalaridir. Bu interfeyslarsiz o'rnatilgan tizimlar mustaqil ishlashi mumkin emas.

Asosiy o'rnatilgan tizim I/O interfeyslari quyidagilar:

1. **Parallel portlar:** Bir vaqtda bir nechta bitlar uzatiladi. Masalan, Printer porti (LPT), PCI, ISA.
2. **Ketma-ket portlar:** Ketma-ket ravishda bir bitdan uzatiladi. Masalan, UART (RS-232), USB, Ethernet.
3. **Sensor va aktuatorlar uchun interfeyslari:** Sensor va aktuatorlar bilan bevosita aloqa o'rnatish uchun. Masalan, ADC, DAC, PWM.
4. **Real vaqt soatlari:** Tizimda real vaqt hisobini yuritish uchun.
5. **Xotira interfeyslari:** Tashqi xotiralar bilan aloqa o'rnatish uchun. Masalan, SDRAM, SRAM, EEPROM.
6. **Display interfeyslari:** LCD, OLED yoki boshqa displeylarni boshqarish uchun. Masalan, HDMI, VGA, DVI.
7. **Audio interfeyslari:** Ovoz kiritish va chiqarish uchun. Masalan, I2S, AC'97.
8. **Tarmoq interfeyslari:** Tizimni tarmoqqa ulash uchun. Masalan, Ethernet, Wi-Fi, Bluetooth.

Ushbu interfeyslardagi asosiy talablar quyidagilar:

- Tezkor, ishonchli va samarali ishlash
- Real vaqtda ma'lumot uzatish
- Minimal resurs sarfi
- Mos keladigan qurilmalar bilan uyg'unlashuv

O'rnatilgan tizimlarning I/O imkoniyatlari ularning qo'llanish doirasi va ehtiyojlarini belgilaydi.

O'rnatilgan tizimlarning kiritish-chiqarish interfeyslari, o'rnatilgan tizimning boshqa tizimlar yoki foydalanuvchilar bilan ma'lumot almashish uchun kerak bo'lgan interfeyslarni ta'minlashni anglatadi. Bu interfeyslar, ma'lumotlarni kiritish, o'qish, va boshqa tizimlar yoki foydalanuvchilar bilan ma'lumot almashishni o'z ichiga oladi. Quyidagi bosqichlar, o'rnatilgan tizimlarning kiritish-chiqarish interfeyslarini ta'minlash jarayonida kerak bo'lgan asosiy amaliyatlarni ta'minlaydi:

1. Boshqaruv Paneli va Grafik Interfeyslar:

- **Boshqaruv Paneli:** O'rnatilgan tizimni boshqarish uchun grafik boshqaruv panelini ta'minlash. Bu panel orqali tizim sozlash, konfiguratsiyalash, va boshqa muhim amallarni bajarish mumkin bo'ladi.
- **Grafik Interfeyslar (GUI):** Grafik interfeyslar orqali foydalanuvchilar o'rnatilgan tizimning barcha funktsiyalaridan foydalanishlari mumkin bo'ladi. Bu, tugma, menu, oyna va boshqa GUI komponentlari orqali amalga oshiriladi.

2. Komand Line Interfeyslar (CLI):

- **Komand Line Interfeysi:** O'rnatilgan tizimni komand line interfeysi orqali boshqarishni ta'minlash. Foydalanuvchilar komand qatorlarini kiritib, tizimning barcha funktsiyalaridan foydalanishlari mumkin.
- **Komand Line Argumentlari:** O'rnatilgan tizim uchun kerakli komand line argumentlarini ta'minlash. Bu argumentlar orqali tizim sozlash va boshqarish amalga oshiriladi.

3. API (Application Programming Interface):

- **RESTful API:** O'rnatilgan tizimning RESTful API sifatida ta'minlash. Bu, boshqa tizimlar yoki ilovalar bilan o'rnatilgan tizim orasidagi ma'lumot almashishni o'zlashtirishga imkon beradi.
- **SOAP API:** O'rnatilgan tizimning SOAP API sifatida ta'minlash. Bu, boshqa tizimlar yoki ilovalar bilan o'rnatilgan tizim orasidagi ma'lumot almashishni o'zlashtirishga imkon beradi.

4. Tizim fayl tizimi:

- **Fayl tizimi:** Fayllarni yuklash, o'qish va o'chirish uchun kerakli interfeyslarni ta'minlash.
- **Fayl Ochildi Boshqaruvchi:** Fayl ochilgan, yozilgan va yopilgan bo'lsa, bu protsessni boshqarish.

Misol:

Agar siz yangi tizimni boshqarish interfeysini tuzishni rejalashtirayotgan bo'lsangiz, quyidagi bosqichlar sizga yordam beradi:

1. **Boshqaruv Paneli va Grafik Interfeyslar:** O'rnatilgan tizimni boshqarish uchun grafik boshqaruv panelini va GUI komponentlarini ta'minlash.
2. **Komand Line Interfeyslar (CLI):** O'rnatilgan tizimni komand line interfeysi orqali boshqarishni ta'minlash, shuningdek, komand line argumentlarini ta'minlash.
3. **API (Application Programming Interface):** O'rnatilgan tizimning RESTful yoki SOAP API sifatida ta'minlash.
4. **Fayl Tizimi:** Fayllarni yuklash, o'qish va o'chirish uchun kerakli interfeyslarni ta'minlash.

Ushbu bosqichlar, o'rnatilgan tizimlarning foydalanuvchilar yoki boshqa tizimlar bilan ma'lumot almashish uchun kerakli interfeyslarni ta'minlashni ta'minlaydi. Bu jarayon, o'rnatilgan tizimning foydalanishini oson va samarali qilishga yordam beradi.

S65.O'rnatilgan tizimlarning kirish -chiqish (chiqish -chiqish) arxitekturasini.

O'rnatilgan tizimlarning kirish-chiqish (Input-Output, I/O) arxitekturasini - bu tizimning qo'shimcha qurilmalar (sensor, aktuator va boshqalar) bilan aloqa o'rnatish yondashuvidir. I/O arxitekturasini tizimning umumiy arxitekturasining asosiy qismi hisoblanadi.

O'rnatilgan tizimlar uchun qo'llaniladigan asosiy I/O arxitekturalari:

1. **Bevosita manzillash:** Protssessor I/O qurilmalarga to'g'ridan-to'g'ri manzil beradi va ma'lumotlar almashinuvini amalga oshiradi.
2. **Xotira-asoslangan manzillash:** Protssessor I/O manzillar oralig'idagi xotira joylariga murojaatlar orqali ma'lumotlar almashinuvini amalga oshiradi.
3. **Tashqi kontroller asoslangan:** I/O qurilmalari alohida kontroller yordamida boshqariladi, protssessor esa faqat bu kontroller bilan muloqot qiladi.
4. **DMA (bevosita xotira kirish-chiqish) asoslangan:** Protssessor I/O ma'lumotlarini bevosita xotiraga almashtirib turadi, bu esa protssessorni yengillashtirib, samaradorligini oshiradi.

I/O arxitekturaning tanlovi quyidagi omillarga bog'liq:

- Tizimning murakkabligi va miqyosi
- Tezkorlik, ishonchoqlik va ishlash samaradorligi talablari
- Resurs sarfi (protssessor, xotira, tezlik)
- I/O qurilmalarining turli-tumanligi va ularni boshqarish imkoniyatlari

I/O arxitekturasini o'rnatilgan tizimlarning samaradorligi, tezkorligi va ishonchliligini belgilaydi. Qulay arxitektura tanlovi muhim ahamiyat kasb etadi.

O'rnatilgan tizimlarning kirish-chiqish arxitekturasini, tizimning boshqa tizimlar yoki foydalanuvchilar bilan ma'lumot almashish, ulashish va boshqa interaktsiyalarni o'z ichiga oladi. Bu arxitektura, tizimning ma'lumotlar olish, ma'lumotlarni qayta ishlash, boshqarish va boshqa tizimlar bilan aloqani boshqarish jarayonlarini ta'minlaydi. Quyidagi bosqichlar, o'rnatilgan tizimlarning kirish-chiqish arxitekturasini ta'minlash uchun kerakli asosiy amaliyatlarni ta'minlaydi:

1. Kirish-Chiqish Protokollari:

- **Aloqa Protokollari:** O'rnatilgan tizimning boshqa tizimlar bilan aloqasi uchun kerakli aloqa protokollarini (masalan, TCP/IP, HTTP, SOAP, REST) ta'minlash.
- **Tarmoq Interfeyslari:** Tarmoq interfeyslarini (Ethernet, Wi-Fi, Bluetooth, etc.) ta'minlash.

2. Qabul-Qilish va Qayta Ishlov Jarayonlari:

- **Ma'lumotni Qabul Qilish:** O'rnatilgan tizimning boshqa tizimlar yoki foydalanuvchilar tomonidan yuborilgan ma'lumotlarni qabul qilish.
- **Ma'lumotni Qayta Ishlov Qilish:** Qabul qilingan ma'lumotlarni to'plab qo'lish, ulardan foydalanib ishlov bajarish, natijalarni tayyorlash va qayta yuborish.

3. Ma'lumotlar Strukturasini va Formatlash:

- **Ma'lumotlar Strukturasini Tashkil Etish:** Ma'lumotlarni kerakli strukturaga olib borish, masalan, JSON, XML, yoki boshqa ma'lumot strukturalari.
- **Ma'lumotlar Formatini Formatlash:** Ma'lumotlarni joynab olish va joynatish uchun kerakli formatlash va kiritish.

4. Kiminchi Asosiy Xususiyatlari:

- **Autentifikatsiya va Xavfsizlik:** Foydalanuvchini kimligini tekshirish va ma'lumotlarni xavfsiz tarzda uzatishni ta'minlash.
- **Kiminchi Asosiy Xususiyatlar (API):** API interfeyslarni ta'minlash, masalan, RESTful yoki SOAP API.

Misol:

Agar siz o'rnatilgan tizimning boshqa tizimlar bilan aloqasini tuzishni rejalashtirayotgan bo'lsangiz, quyidagi bosqichlar sizga yordam beradi:

1. **Aloqa Protokollari:** O'rnatilgan tizimning boshqa tizimlar bilan aloqasini o'rnatish uchun kerakli aloqa protokollarini ta'minlash, masalan, TCP/IP yoki HTTP.
2. **Ma'lumotni Qabul Qilish va Qayta Ishlov Qilish:** O'rnatilgan tizimning ma'lumotlarni qabul qilish va ulardan foydalanib ishlov bajarish, natijalarni tayyorlash va boshqa tizimlar bilan aloqani boshqarish uchun kerakli protsesslarni ta'minlash.
3. **Ma'lumotlar Strukturasini va Formatlash:** Qabul qilingan ma'lumotlarni kerakli strukturaga olib borish va ulardan foydalanish uchun kerakli formatlashni ta'minlash.
4. **Kiminchi Asosiy Xususiyatlari (API):** API interfeyslarni ta'minlash, masalan, RESTful yoki SOAP API ni o'rnatish.

Ushbu bosqichlar, o'rnatilgan tizimlarning boshqa tizimlar yoki foydalanuvchilar bilan ma'lumot almashish va ulashish uchun kerakli asosiy amaliyatlarni ta'minlashni ta'minlaydi. Bu jarayon, o'rnatilgan tizimning boshqa tizimlar yoki foydalanuvchilar bilan muvaffaqiyatli aloqasini ta'minlashga yordam beradi

S66.Shinaga ulanadigan kirish -chiqish interfeysi uskunasi.

Shinaga ulanadigan kirish-chiqish (input-output, I/O) interfeysi uskunalari - bu o'rnatilgan tizimlar bilan tashqi qurilmalar o'rtasidagi aloqa o'rnatish vositalaridir. Bu uskunalar shinaga ulangan protsessor yoki mikrokontroller bilan bog'lanib, I/O amallarini amalga oshiradi.

Shinaga ulanadigan asosiy I/O interfeysi uskunalari quyidagilar:

1. **Parallel portlar:** Parallel portlar bir vaqtda bir necha bitlarni uzatish imkoniyatiga ega. Masalan, PCI, ISA, Printer poriti (LPT).
2. **Ketma-ket portlar:** Ketma-ket portlar bir bitdan ma'lumotni ketma-ket uzatadi. Masalan, UART (RS-232), USB, Ethernet.
3. **Sensor va aktuatorlar uchun interfeyslari:** Sensor va aktuatorlar bilan bevosita aloqa o'rnatish uchun. Masalan, ADC, DAC, PWM.
4. **Real vaqt soat modullari:** Tizimning real vaqt hisobini yuritish uchun.
5. **Xotira interfeyslari:** Tashqi xotiralar bilan aloqa o'rnatish uchun. Masalan, SDRAM, SRAM, EEPROM.
6. **Display interfeyslari:** LCD, OLED yoki boshqa displeylar bilan aloqa o'rnatish uchun. Masalan, HDMI, VGA, DVI.
7. **Audio interfeyslari:** Ovoz kiritish va chiqarish uchun. Masalan, I2S, AC'97.
8. **Tarmoq interfeyslari:** Tizimni tarmoqqa ulash uchun. Masalan, Ethernet, Wi-Fi, Bluetooth.

Bu I/O interfeyslari qurilmalarini tizimga ulash va ulardan foydalanishni ta'minlaydi. Shinaga ulanadigan interfeyslardagi asosiy talablar - tezkorlik, ishonchoqlik, real vaqt hisobi va minimal resurs sarfidir.

Shinaga ulanadigan kirish-chiqish interfeysi uskunasi (SCSI), bilan ma'lumot almashish va ma'lumotlarni uzatish uchun yana bir avlodli tarmoq protokolidir. SCSI uskunasi, asosan disk yoki boshqa qurilmalarni tarmoqqa ulanish va ulanishni boshqarish uchun ishlatiladi. Quyidagi bosqichlarda SCSI uskunasi haqida ko'proq ma'lumot berilgan:

1. SCSI uskunasi nima uchun ishlatiladi?

- **Disk tarmoqlari:** SCSI uskunasi, oddiy va katta hajmdagi disk tarmoqlari bilan ishlaydi. Bu tarmoqlar, ma'lumotlarni saqlash va o'qish uchun ishlatiladi.
- **Periferiya qurilmalari:** SCSI uskunasi, skanerlar, printerlar, optik disk vositalari va boshqa periferiya qurilmalariga ulanish uchun ham ishlatiladi.

2. SCSI uskunasi turlari:

- **SCSI-1:** Asosiy SCSI uskunasi bo'lib, uning tezligi 5 MB/s va bir qurilmaga yoki ustunaga ulanish imkoniyati mavjud.

- **SCSI-2:** SCSI-1 ning mustahkamlashgan versiyasi, uning tezligi 10 MB/s va 8 qurilmaga ulanish imkoniyati mavjud.
- **SCSI-3 (Ultra SCSI):** Qadamlashtirilgan SCSI versiyasi, uning tezligi 20 MB/s va 16 qurilmaga ulanish imkoniyati mavjud.
- **SCSI-4 (Ultra2 SCSI):** Ultra SCSI-ning mustahkamlangan versiyasi, uning tezligi 40 MB/s va qo'shimcha qurilma ulanishi uchun tashqi buslarda ishlatilgan.
- **SCSI-5 (Ultra3 SCSI):** Ultra2 SCSI-ning yanada mustahkamlangan versiyasi, uning tezligi 80 MB/s va ishlab chiqarish va tashqi buslarda o'zgarishlar mavjud.
- **SCSI-6 (Ultra160 SCSI):** Ultra3 SCSI-ning yanada mustahkamlangan versiyasi, uning tezligi 160 MB/s va yana bir necha mustahkamlashlar qo'shilgan.
- **SCSI-7 (Ultra320 SCSI):** Ultra160 SCSI-ning yanada mustahkamlangan versiyasi, uning tezligi 320 MB/s va yanada mustahkamlashlar qo'shilgan.

3. SCSI uskunasi amaliyoti:

- **Daisy Chain:** SCSI tizimi, bir narsa (masalan, bir kompyuter)dan boshqa bir narsaga (masalan, bir disk bo'limiga) ulanish uchun daisy chain texnologiyasidan foydalanadi.
- **Terminatsiya:** Ulanishning har ikki tomondagi soha chiqishida, tarmoqning oxiridagi va boshida qurilgan terminatorlar orqali tarmoqni tugatish uchun zarur.
- **Adresslangan qurilmalar:** SCSI tarmoqlar, har bir qurilmaning SCSI turiga mos adressga ega bo'lishi uchun to'plam transportirovka tarmoqlarini qo'llab-quvvatlaydi.

4. SCSI uskunasi kirish - chiqish interfeysi:

- **Paralel Interfeys:** Paralel SCSI uskunasi, oddiy paralel interfeys orqali kompyuter bilan ulanadi. Ushbu interfeysda bir nechta kabel va oddiy elektrik signal tizimlari ishlatiladi.
- **Serial Interfeys:** Serial SCSI (SAS) uskunasi, sinovli interfeysni taqdim etadi va oddiy kabel va elektrik signal tizimlari ishlatilmaydi. Bu interfeys, tezkorlik, ustunlik va keng doirasini taqdim etadi.

Misol:

Agar siz disk tarmoqlar yoki boshqa periferiya qurilmalar bilan ulanish uchun SCSI uskunasi kirish - chiqish interfeysini o'rnatishni rejalashtirayotgan bo'lsangiz, SCSI uskunasi tizimini ishlatish uchun kerakli harakatlar quyidagicha bo'ladi:

1. **Paralel Interfeys o'rnatish:** Paralel SCSI uskunasi tizimini kompyuter bilan ulanish uchun kerakli paralel interfeysni o'rnatish.
2. **Serial Interfeys o'rnatish:** Serial SCSI (SAS) uskunasi tizimini kompyuter bilan ulanish uchun kerakli serial interfeysni o'rnatish.

Ushbu bosqichlar, SCSI uskunasi tizimini kompyuter bilan ulanish va ulanishni boshqarish uchun kerakli interfeyslarni ta'minlashni ta'minlaydi. Bu jarayon, disk tarmoqlar va boshqa periferiya qurilmalari bilan SCSI uskunasi orqali muvaffaqiyatli aloqalarni ta'minlashga yordam beradi.

S67.Ikkinchi avlod mikroprotessorli shina (PCI).

PCI (Peripheral Component Interconnect), kompyuter komponentlarini bir-biriga bog'lash uchun ishlatiladigan avtomatlashtirilgan tarmoq standarti. Ikkinchi avlod mikroprotessorli shina (PCI) yaxshi ko'rsatishli grafika kartalari, har disk qurilmalari va boshqa ko'p yadroli tarmoq qurilmalari bilan birga ishlovchi tizimlar uchun standart interfeysni ta'minlashda foydalaniladi. Quyidagi bosqichlarda PCI tarmoq standarti haqida ko'proq ma'lumot berilgan:

1. PCI tarmoq standarti nima uchun ishlatiladi?

- **Komponentlar orasidagi bog'lanish:** PCI tarmoq standarti, kompyuter komponentlarini bog'lash uchun ishlatiladi. Bu qurilmalar grafika kartalari, hard disk qurilmalari, USB qurilmalari, sifatli muzlatkichlar va boshqa qurilmalarni o'z ichiga oladi.
- **Har xil qurilmalar uchun xizmat ko'rsatish:** PCI standarti, har xil tarmoq komponentlarining (audio, video, disk va boshqalar) bog'lanishini ta'minlaydi.

2. PCI standarti turli avlodlarda bo'ladi:

- **PCI:** Asosiy PCI standarti 32-bit o'zgaruvchanlarni qo'llaydi va bu o'zgaruvchilar 33 MHz tezligida ishlaydi.
- **PCI-X:** Kattalashtirilgan PCI, PCI-X standarti, standartni 64-bit o'zgaruvchilarga va 66 MHz tezligiga kengaytirdi.
- **PCI Express (PCIe):** Yangi avlod, PCI Express (PCIe), katta tezlikda o'zgaruvchilarni, keng qo'llanish va yetarli kuch ta'minlash uchun xususiyatlar bilan ta'minlanadi.

3. PCI standarti asosiy xususiyatlari:

- **Paralel tarmoq:** PCI tarmoq, paralel tarmoq usuli orqali komponentlar orasida ma'lumot almashish va almashish imkonini ta'minlaydi.
- **Avlodi o'zgaruvchanlar:** PCI standarti, 32-bit va 64-bit o'zgaruvchilar bilan ishlaydi va boshqa qurilmalarga bog'lash uchun keng ko'lamli bog'liqlikni ta'minlaydi.
- **Boshqarish va bloklovchi qurilmalar:** PCI tarmoqi, boshqarish va bloklovchi qurilmalar bilan ishlashga imkon beradi, shuningdek, ularga xavfsizlikni ta'minlash uchun xavfsizlik funktsiyalarni taqdim etadi.

4. PCI tarmoqning kirish-chiqish interfeysi:

- **Paralel interfeys:** PCI tarmoq standarti, oddiy paralel interfeysni o'z ichiga oladi. Komponentlar tomonidan yuborilgan ma'lumotlar parallel o'zgaruvchilarda o'tkaziladi.
- **Kengaytirilgan Interfeys:** PCI Express (PCIe) yoki PCI-X kabi turli versiyalari, komponentlar orasida ma'lumot almashish uchun kengaytirilgan interfeysni ta'minlaydi.

Misol:

Agar siz kompyuter komponentlarini bog'lash uchun ikkinchi avlod mikroprotssessorli shinani (PCI) o'rnatishni rejalashtirayotgan bo'lsangiz, quyidagi bosqichlar sizga yordam beradi:

1. **PCI kartlarni o'rnatish:** PCI xoblariga mos keladigan komponentlarni o'rnatish.
2. **To'plam tarmoqni tuzish:** To'plam tarmoqni tuzish va avlodning kerakli interfeyslarini qo'llab-quvvatlash.

Ushbu bosqichlar, ikkinchi avlod mikroprotssessorli shina (PCI) orqali kompyuter komponentlarini bog'lash uchun kerakli tarmoq interfeyslarini ta'minlashni ta'minlaydi. Bu jarayon, kompyuter komponentlarining to'plamini oson va samarali qilishga yordam beradi.

Ikkinchi avlod mikroprotssessorli shina - PCI (Peripheral Component Interconnect)
PCI - bu mikroprotssessorli tizimlar uchun ishlab chiqilgan ikkinchi avlod shinalaridan biridir. PCI shina 1992 yilda IBM, Digital, NCR, Northern Telecom va DEC kompaniyalari tomonidan taklif qilindi.

PCI shinaning asosiy xususiyatlari:

1. **Parallel shina:** PCI - 32-bitli parallel shina bo'lib, ma'lumotlarni parallel uzatadi.
2. **Plug and Play:** PCI qurilmalar avtomatik ravishda aniqlanadi va konfiguratsiya qilinadi. Operatsion tizim tomonidan avtomatik ravishda qurilmalar menejmenti amalga oshiriladi.
3. **Tezkorlik:** PCI shina tezligi 33 yoki 66 MHz tashkil etadi.
4. **Quvvat menejmenti:** PCI qurilmalar o'chirib qo'yilganda yoki kam quvvatda ishlash rejimiga o'tadi.
5. **Keng tarqalgan:** PCI shina PC, server va boshqa elektronika mahsulotlarida keng qo'llanildi.
6. **Kam resurs sarfi:** PCI - kam protssessor, xotira va kabel resurslaridan foydalanadi.

PCI shinasini protsessor, xotira va boshqa tizim komponentlari bilan uyg'un ishlaydi. PCI qurilmalar (grafik karta, tezkor SSD, tarmoq kartasi va boshqalar) protsessor bilan ketma-ket bog'lanadi.

PCI shinasining asosiy afzalliklari:

- Plug and Play qo'llab-quvvatlash
- Tezkorlik va ishonchlilik
- Kam resurs sarfi
- Keng tarqalganligi

PCI shina hozirgi kungacha kompyuter va server texnologiyalarida keng qo'llanilmoqda, ammo yangi PCI Express shinasini bilan yanada rivojlanib bormoqda.

S68.Real vaqt OT foydalanish

Real vaqt operatsion tizimlari (Real-Time Operating Systems - RTOS)

Real vaqt operatsion tizimlari - bu muayyan vaqt chegarasida bajariladigan vazifalarni qat'iy amalga oshirishni ta'minlaydigan maxsus OTlar. Bunda har bir vazifa o'z muddatida bajarilishi shart.

RTOS-ning asosiy xususiyatlari:

1. **Vaqt ajratish:** RTOS jarayonlar uchun qattiq vaqt cheklovlarini qo'llaydi. Har bir vazifa uchun minimal bajarish vaqti belgilanadi.
2. **Prediktivlik:** RTOS rejalashtirish va bajarish jarayonlarini oldindan bashorat qilish qobiliyatiga ega.
3. **Ishonchlilik:** RTOS vazifalarning bajarilishi ustidan qat'iy nazorat olib boradi va muddatida bajarilishini ta'minlaydi.
4. **Determinizm:** RTOS harakatining natijasi oldindan aytib beriladi.
5. **Kam resurs sarfi:** RTOS uchun kam protsessor quvvati, xotira va kuchda ishlashi kifoya.

RTOS-ning asosiy qo'llanish sohasi:

- Embedded tizimlari (mikrokontroller, FPGA, SoC)
- Avtomatika, robotlar, transport vositalari
- Tibbiy asbob-uskunalar
- Telekomunikatsiya va aloqa tizimlar
- Muxandislikdagi boshqaruv tizimlari

Mashhur RTOS-lar: Linux RT, FreeRTOS, VxWorks, QNX, TinyOS, eCos, RTEMS.

Real vaqt operatsion tizimlaridan foydalanish real vaqt tizimlarida muhim ahamiyatga ega, chunki ular vazifalarning haqiqiy vaqtda bajarilishini ta'minlaydi. Bu tizimlarda yuqori darajadagi ishonchlilik, determinizm va resurs samaradorligi muhim talablar hisoblanadi.

Real vaqt (Real-time) operatsion tizimlarda (OT) foydalanish, amalni o'z vaqtida bajarishga yordam beradi. Bu tizimlar ma'lumotlarni o'z vaqtida qabul qilish, tahlil qilish va javob berishga murojaat qiladi. Real vaqt operatsion tizimlar, maslahat berish, monitoring qilish, avtomatlashtirilgan kontrollarni boshqarish va boshqa kritik jarayonlarni bajarishda o'z vaqti yoki kiritilgan vaqtga muvofiq ravishda faoliyat yuritishga qodir. Quyidagi bosqichlarda real vaqt operatsion tizimlarining foydalanishiga misol keltirilgan:

1. Soat va ma'lumot qaytarish tizimlari:

- **Transport tizimlari:** Real vaqt tizimlar, transport tizimlarida avtomatlashtirilgan tizimlar, masalan, metro, temir yo'llarida yoki avtomobil sotuv tizimlarida amalga oshiriladi.

- **Bank tizimlari:** Banklar va moliya institutlari, ma'lumotlarni o'z vaqtida qabul qilish, taqsimlash va ma'lumotlarini avtomatik ravishda ishlab chiqish uchun real vaqt tizimlaridan foydalanishadi.

2. Monitoring va Avtomatlashtirilgan Boshqarish Tizimlari:

- **Ilovalar monitoringi:** Real vaqt tizimlar, texnik jarayonlarni monitoring qilish uchun ilovalar ishga tushiriladi. Masalan, temperaturani nazorat qilish, hisoblash va avtomatik monitoring vaqtli ravishda bajariladi.
- **Avto boshqarish tizimlari:** Real vaqt tizimlar, qattiq bosqichda avtomatik boshqarish tizimlari, masalan, maydon, tez-tez ishlab chiqarilgan qurilmalar, fabrikalar va energiya tarmoqlari uchun ishlatiladi.

3. Moliyaviy va Bank Tizimlari:

- **Boshqaruv va hisobkitob jarayonlari:** Moliyaviy institutlar va banklar, muhlatni buzmaslik va tranzaksiyalarni o'z vaqtida ishlab chiqish uchun real vaqt tizimlaridan foydalanadi.
- **Bank xizmatlari:** Banklar, ATM, kart xizmatlari va elektronik to'lov sistemlarini monitoring qilish va ishlatishda real vaqt tizimlaridan foydalanishadi.

Misol:

Agar siz bir korxona boshqaruv tizimini loyihalashtirayotgan bo'lsangiz, quyidagi bosqichlar sizga yordam beradi:

1. **Monitoring tizimlari:** Korxona jarayonlarini va hisobotlarni o'z vaqtida monitoring qilish uchun monitoring tizimlarini tuzish.
2. **Avtomatik boshqarish:** Energiya taqsimoti, ishlab chiqarish va boshqa jarayonlarni avtomatik ravishda boshqarish uchun avtomatik boshqarish tizimlarini ishga tushirish.

Ushbu bosqichlar, korxona jarayonlarini monitoring qilish va avtomatik boshqarishni ta'minlash uchun real vaqt operatsion tizimlaridan foydalanishni ta'minlaydi. Bu jarayon, korxonani ishlab chiqarish va xizmatlarni o'z vaqtida va samarali boshqarishga yordam beradi.

S69.Foydalanuvchi ilovalari Real vaqtda O'rnatilgan OS arxitekturasida.

O'rnatilgan tizimlar (Embedded systems) uchun real vaqt operatsion tizimlari (RTOS) ancha murakkab arxitekturaga ega. Bu arxitektura quyidagi asosiy komponentlardan iborat:

1. **Real vaqt kernel:** Real vaqt vazifalarini qat'iy muddatda bajarish uchun javobgar bo'lgan asosiy komponent. Real vaqt kernel quyidagi funksiyalarni bajaradi:
 - Protessorni vaqtinchalik bo'lib, har bir vazifa uchun alohida vaqt ajratish
 - Qattiq muddatlar chegarasida vazifalarni bajarilishini nazorat qilish
 - Yuqori ustuvorlikdagi vazifalarning bajarilishini ta'minlash
2. **Qurilmalar drayverlari:** Qurilmalar bilan aloqa o'rnatish uchun yozilgan drayverlari. Bu drayverlarga kirish RTOS-ning real vaqt kerneli orqali amalga oshiriladi.
3. **Amaliy ilovalar:** O'rnatilgan tizim uchun yozilgan ilovalar. Bu ilovalar real vaqt kerneli tomonidan boshqariladi va ularga ajratilgan vaqt chegarasida bajariladi.
4. **Xavfsizlik va konfiguratsiya:** RTOS-da xavfsizlik va tizim konfiguratsiyasi bilan bog'liq funksiyalar amalga oshiriladi.
5. **Tarmoq protokollari:** Tarmoq aloqasi uchun TCP/IP, CAN, LIN va boshqa protokollar qo'llab-quvvatlanadi.
6. **Foydalanuvchi interfeysi:** Tizim holati va boshqaruv funksiyalarini taqdim etuvchi grafik yoki matn interfeyslar.

Real vaqt kernel ushbu komponentlarni integratsiyalangan holda boshqaradi va ularning qat'iy muddatda bajarilishini ta'minlaydi. Bu arxitektura o'rnatilgan tizimlarning real vaqt talablariga javob beradi.

O'rnatilgan real vaqt operatsion tizimlaridan asosiy foydalanish sohalari:

- Avtomobillar va transport vositalari
- Tibbiy asbob-uskunalar
- Sanoat avtomatlashtirilgan tizimlari
- Robotlar va boshqa muxandislik tizimlar

Foydalanuvchi ilovalari real vaqtda o'rnatilgan operatsion tizim (OS) arxitekturasida, foydalanuvchilar uchun samarali, tez ishlash va tizimni qat'iy yoki to'liq to'xtatmasdan faol ishga tushirishni ta'minlaydi. Bu tizimlar maslahat berish, monitoring, avtomatik boshqarish, foydalanuvchilar interfeysi va boshqa kritik jarayonlarni o'z vaqti yoki kiritilgan vaqtga muvofiq bajarishga qodir. Quyidagi bosqichlarda foydalanuvchi ilovalari real vaqtda o'rnatilgan operatsion tizimlarining arxitekturasida haqida ko'proq ma'lumot berilgan:

1. Real vaqtda o'rnatilgan OS arxitekturasida yoki foydalanuvchi ilovalari:

- **Real vaqt operatsion tizimlari:** Foydalanuvchi ilovalari real vaqtda o'rnatilgan operatsion tizimlar, ma'lumotlarni o'z vaqtida qabul qilish, tahlil qilish va javob berishga yordam beradi.
- **Monitoring tizimlari:** Foydalanuvchi ilovalari real vaqtda o'rnatilgan operatsion tizimlar, monitoring va avtomatik boshqarish uchun muvofiqlashgan monitoring tizimlarini ta'minlashda ishlatiladi.
- **Avtomatik boshqarish tizimlari:** Foydalanuvchi ilovalari real vaqtda o'rnatilgan operatsion tizimlar, avtomatik boshqarish uchun muvofiqlashgan avtomatik boshqarish tizimlarini o'rnatishda yordam beradi.

2. Foydalanuvchi ilovalari real vaqtda o'rnatilgan OS arxitekturasini keng qamrovli tizimlarga:

- **Bank tizimlari:** Banklar va moliya institutlari, ma'lumotlarni o'z vaqtida qabul qilish, tahlil qilish va javob berishga yordam berish uchun foydalanuvchi ilovalari real vaqtda o'rnatilgan operatsion tizimlardan foydalanishadi.
- **Transport tizimlari:** Transport tizimlari, masalan, metro, temir yo'llarida yoki avtomobil sotuv tizimlarida real vaqt operatsion tizimlaridan foydalanishadi.
- **Boshqaruv va hisob-kitob jarayonlari:** Foydalanuvchi ilovalari real vaqtda o'rnatilgan operatsion tizimlar, korxona jarayonlarini monitoring qilish, hisob-kitob jarayonlarini avtomatik ravishda boshqarish uchun ishlatiladi.

Misol:

Agar siz transport tizimini boshqaruv tizimini loyihalashtirayotgan bo'lsangiz, quyidagi bosqichlar sizga yordam beradi:

1. **Monitoring tizimlari:** Transport tizimini vaqtli ravishda monitoring qilish uchun monitoring tizimlarini tuzish.
2. **Avtomatik boshqarish:** Energiya taqsimoti, ishlab chiqarish va boshqa jarayonlarni avtomatik ravishda boshqarish uchun avtomatik boshqarish tizimlarini ishga tushirish.

Ushbu bosqichlar, transport tizimini monitoring qilish va avtomatik boshqarishni ta'minlash uchun foydalanuvchi ilovalari real vaqtda o'rnatilgan operatsion tizimlardan foydalanishni ta'minlaydi. Bu jarayon, transport tizimini samarali va tezkor boshqarishga yordam beradi.

S70.API tizimi va Yadro.

O'rnatilgan tizimlar uchun

arida (RTOS) asosiy ahamiyatga ega bo'lgan komponentlar - **API tizimi** va **Yadro**dir.

API tizimi (Application Programming Interface):

- API tizimi RTOS-ning tashqi interfeysi hisoblanib, ilovalar bilan interaksioni ta'minlaydi.
- API orqali ilovalar RTOS xizmatlari va funksiyalaridan foydalanadi: vazifalarni yaratish, blokirovka, sinxronizatsiya, taqsimot, xotira boshqaruvi, qurilmalar bilan ishlash va hokazo.
- API standartlashtirilgan bo'lib, barcha RTOS-larda deyarli bir xil bo'ladi. Masalan, POSIX API, Win32 API, OSEK API.
- API tizimi RTOS yadrosidan ajratilgan bo'lib, yadro bilan interfeys orqali aloqa qiladi.

Yadro (Kernel):

- RTOS yadro komponentining asosiy vazifasi - real vaqt talablarini qanoatlantirish uchun jarayonlarni boshqarishdir.
- Yadro quyidagi asosiy funksiyalarni bajaradi:
 - Vaqt taqsimoti - har bir vazifa uchun ajratilgan protsessor vaqtini boshqaradi
 - Tizim resurslarini boshqarish - xotira, qurilmalar, interruptlar va boshqalar
 - Tasklar/threadlar boshqaruvi - yaratish, yopish, blokirovka, sinxronizatsiya
 - Interruptlar boshqaruvi - real vaqt talablarini qanoatlantirish uchun
- Yadro kam hajmga ega bo'lishi va tezkor ishlashi kerak. Shuning uchun yadro funksiyalari aniq cheklangan.
- Yadro API funksiyalar to'plamini ta'minlaydi, ilovalar ushbu funksiyalardan foydalanadi.

API tizimi va yadro o'zaro integratsiyalashgan bo'lib, RTOS-ning asosini tashkil etadi. Ushbu komponentlar RTOS-ning real vaqt xususiyatlarini ta'minlashda muhim rol o'ynaydi.

API (Application Programming Interface) tizimi, dasturlar, ilovalar va tizimlar o'rtasida ma'lumot almashish va almashish uchun moslashtirilgan interfeyslardir. Yadro (Kernel) esa operatsion tizimning asosiy qismidir va dasturlar uchun ma'lumotlar va resurslarga murojaatni ta'minlaydi. Quyidagi qismda API tizimi va Yadro haqida ko'proq ma'lumot berilgan:

API tizimi:

- **API:** Application Programming Interface, dasturlar va ilovalar o'rtasida ma'lumot almashish va almashish uchun moslashtirilgan interfeysdir. API, dasturlar va tizimlar uchun aloqador funksiyalar va protokollar to'plamini ta'minlaydi.
- **Foydalanish joylari:** Dastur yaratishda, kodni yozishda yoki tizimlarni integratsiyalashda foydalanuvchi interfeysi sifatida ishlatiladi. Bu, dastur olish uchun funksiyalarni va protokollarni aniqlash, ma'lumot almashish va ma'lumotlarni yuborish, hisob-kitobni bajarish va boshqa keng qamrovli amallarni bajarish uchun ishlatiladi.
- **Turi:** API turli shakllarda bo'lishi mumkin, masalan, dastur, veb-saytlar, dasturiy interfeyslar, tarmoq protokollari va boshqalar uchun API mavjud.

Yadro:

- **Yadro:** Operatsion tizimning asosiy qismidir va asosiy funksiyalarni bajaradi. Bu, to'g'ridan-to'g'ri qurilmadagi ilovalar bilan murojaat qilishga ruxsat beradi. Yadro, qurilma, operatsion tizimning asosiy ishlab chiqariladigan qismi hisoblanadi.
- **Funksiyalari:** Yadro asosiy funksiyalarni bajaradi, masalan, ma'lumotlar o'qilishi, yozilishi, qurilmaga murojaat qilish, boshqa dasturlar bilan murojaat qilish va boshqalar.
- **Ishtirok etish:** Yadro, boshqa ilovalar bilan o'rtasida ishlaydigan va foydalanuvchilarga tizim resurslariga murojaat qilish imkonini ta'minlaydigan API lar orqali ishlaydi.

API tizimi va Yadro munosabatlari:

- **Tavzil:** API, dasturlar va tizimlar o'rtasida aloqa o'rnatish uchun interfeysni ta'minlaydi. Yadro esa operatsion tizimning asosiy qismi hisoblanadi va asosiy funksiyalarni bajaradi.
- **Tegishli bo'lish:** Yadro, API tizimi orqali dasturlar bilan murojaat qilish uchun imkoniyat yaratadi. API, Yadroga murojaat qilishni tizimlar va dasturlarga ta'minlaydi.
- **Tarmoq va boshqa ilovalar bilan ishlash:** API tizimi, Yadro orqali boshqa tizimlar va ilovalar bilan ishlash imkoniyatini ta'minlaydi.

Misol:

Agar siz veb-sayt yaratayotgan bo'lsangiz, sizning veb-saytingiz va server o'rtasida ma'lumot almashish va almashishni o'z ichiga oladigan API tizimiga ega bo'lishi kerak. Bu API tizimi serverning Yadrosi bilan bog'liq bo'ladi va sizning veb-saytingiz o'zgarishlar uchun server bilan murojaat qilish uchun bu API ni ishlatadi.

S71.Xotira arxitekturası.

O'rnatilgan tizimlar uchun real vaqt operatsion tizimlarda xotira arxitekturası

O'rnatilgan real vaqt operatsion tizimlarida (RTOS) xotira arxitekturası ancha muhim ahamiyatga ega. Xotira arxitekturası quyidagi asosiy komponentlardan iborat:

1. Xotira boshqaruv tizimi (Memory Management Unit - MMU):

- RTOS yadrosining asosiy vazifalaridan biri - xotira resurslarini samarali boshqarishdir.
- MMU xotira sahifalash, qo'llab-quvvatlash, blokirovka va boshqa xotira bilan bog'liq operatsiyalarni bajaradi.
- MMU orqali yoki uning qo'llab-quvvatlanmasligi tufayli RTOS yoki ilovalar virtual xotiradan foydalanishlari mumkin.

2. Xotira segmentlari:

- RTOS xotira segmentlari - yadro, drayverlari, ilovalar va boshqa tizimli komponentlar uchun ajratiladi.
- Har bir segment uchun xotira himoyasi, prioriteti, ochilish/yopilish huquqlari belgilanadi.
- Segmentlarni to'g'ri taqsimlash va konfiguratsiyalash RTOS xavfsizligi va samaradorligida muhimdir.

3. Dinamik xotira boshqaruvi:

- Ilovalar va tizimli komponentlarga dinamik xotira ajratish va erkin qilish imkonini beradi.
- Xotira bo'sh qolmasligi, fragmentatsiya va boshqa muammolarni oldini olish uchun samarali algorithm ishlatiladi.

4. Xotira himoya:

- Xotira segmentlarini yoki butun xotiraning himoyasi uchun mexanizmlar mavjud.
- Masalan, Access Control List, Privileged/Unprivileged rejimlari.

Ushbu komponentlar RTOS yadrosida integratsiyalangan holda ishlatiladi va xotira resurslarini samarali boshqarish va xavfsizligini ta'minlaydi. Xotira arxitekturası RTOS-ning real vaqt talablarini qanoatlantirishi uchun juda muhimdir.

Xotira arxitekturası, operatsion tizimlar va dasturlar tomonidan foydalaniladigan xotira (memory) tizimi o'rganilishi va boshqarilish usullari va qonunlarini ta'minlaydi. Xotira, ma'lumotlarni saqlash, o'qish va yozish uchun kompyuterda fizikaviy saqlash oynasi hisoblanadi. Quyidagi qismda xotira arxitekturası haqida ko'proq ma'lumot berilgan:

1. Kesh (Cache) Xotirasi:

- **Kesh (Cache):** Kesh, samarali ma'lumot o'qish va yozish uchun yuqori tezlikda xotira hisoblanadi. U operatsion tizim tomonidan amalga oshiriladigan qo'shimcha xotira darajasi hisoblanadi.
- **Qurilmalar uchun qo'llanish:** Kesh, processorlar, disk qurilmalari va boshqa qurilmalar uchun samarali ma'lumotlarni saqlash va unga tez foydalanish uchun qo'llaniladi.

2. Asosiy (Main) Xotira:

- **Asosiy (Main) Xotira:** Asosiy xotira, RAM (Random Access Memory) deb ataladi. Bu xotira, faqat processorlar va dasturlar tomonidan keng ko'lamli amalga oshiriladi.
- **Tezlik va keng ko'lamli xotira:** Asosiy xotira, samarali dasturlarni ishga tushirish uchun tez foydalanish imkonini ta'minlaydi.

3. Disk Xotirasi:

- **Disk Xotirasi:** Disk xotirasi, ma'lumotlarni saqlash uchun disk qurilmalari yoki SSD larda o'zlashtiriladi. U o'zlashtirilgan xotira bo'lib, qurilma qabul qilgan barcha ma'lumotlarni saqlaydi.
- **Tayanch (Virtual) Xotira:** Disk xotirasi o'zida tayanch xotira hisoblanadi, bu esa operatsion tizim tomonidan asosiy xotira sifatida ishlatiladi.

4. Qo'shimcha Xotira Turi:

- **Piksel (Pixel):** Piksel xotirasi, grafikani saqlash va qayta ishlash uchun ishlatiladi.
- **Bufferni (Buffer):** Bufferni xotirasi, boshqa qurilmalardan ma'lumotlarni o'qish va yozish uchun ishlatiladi.

5. Fizikaviy Xotira:

- **Fizikaviy Xotira:** Bu xotira, kompyuterda asosiy xotira hisoblanadi va barcha faoliyatlar uchun yuqori tezlikda ma'lumotlar saqlanadi.

6. Xotira hierarxchasi:

- **Xotira hierarxchasi:** Xotira hierarxchasi, kesh, asosiy xotira va disk xotirasi kabi turli xotira turlarini o'z ichiga oladi. Bu, ma'lumotlarni saqlash, o'qish va yozish uchun moslashtirilgan xotira turlarining qatorini ta'minlaydi.

7. Tayanch (Virtual) Xotira:

- **Tayanch (Virtual) Xotira:** Tayanch xotira, asosiy xotira kengaytirilgan versiyasi hisoblanadi. Agar asosiy xotira to'la bo'lsa, operatsion tizim tayanch xotiradan foydalanadi.

Misol:

Agar siz bir kompyuter arxivini yaratmoqchi bo'lsangiz, sizning arxiv dasturi fizikaviy xotiraga yoki disk xotirasiga ma'lumotlarni saqlab turishi kerak bo'ladi. Lekin, dasturingiz foydalanuvchilari tez ishlayotgan ma'lumotlarga dasturingizga qaytish imkonini ta'minlash uchun kesh xotirasidan foydalanishadi. Bunda asosiy xotira faqat arxivning kattaligini saqlash uchun ishlatiladi.

S72.Asosiy operatsion tizim (OS) xizmatlari, Fayl tizimlari

Real vaqt operatsion tizimlarida (RTOS) quyidagi asosiy OS xizmatlari mavjud:

1. **Vaqt boshqaruvi:**
 - Vazifalar uchun vaqt taqsimoti
 - Vaqt sinxronizatsiyasi, kechikish, uyg'onish va hokazo
2. **Jarayon/Vazifa boshqaruvi:**
 - Vazifalarni yaratish, yopish, blokirovka, sinxronizatsiya
 - Prioritet, kontekst almashtirish, planlashtirish
3. **Xotira boshqaruvi:**
 - Xotira ajratish, qaytarish, qo'riqash
 - Dinamik xotira boshqaruvi
4. **Qurilma boshqaruvi:**
 - Qurilmalarni ro'yxatga olish, konfiguratsiyalash
 - Qurilmalar uchun drayverlar
5. **Sinxronizatsiya va o'zaro ishlash:**
 - Semafora, muteks, event flag kabi sinxronizatsiya ob'ektlari
 - Interprotssessor muloqoti, uzatish va qabul qilish
6. **Xavfsizlik va himoya:**
 - Foydalanuvchi/tizim rejimlarini ajratish
 - Xotira, umumiy resurslarni qo'riqlab, himoyalash
7. **Fayl tizimlari:**
 - Tizimli va foydalanuvchi fayllari uchun fayl tizimi
 - Fayl tizimi uchun xotira tarqatish va boshqarish

RTOS-da asosiy vazifa - ilovalar ishlatishni ta'minlash, ularning real vaqt talablarini qanoatlantirish. Shuning uchun ushbu OS xizmatlari aniq ta'riflanadi va minimal hajmga ega bo'ladi.

Fayl tizimi RTOS-da kam taraqqiy etgan bo'ladi. Odatda, tizim holati va konfiguratsiya ma'lumotlari uchun to'g'ridan-to'g'ri xotira yoki qattiq disk emulyatsiyasi asosida soddalashtirilgan fayl tizimi ishlatiladi. Bunda sifat parametrlari yaxshi bo'lishi, tezkor ishlashi kerak.

Asosiy operatsion tizim (OS) xizmatlari, kompyuter tizimlarida kelayotgan asosiy funksiyalarni bajarishda yordam beradi. Fayl tizimlari esa ma'lumotlarni saqlash, boshqarish va qo'llash uchun mo'ljallangan tizimlardir. Quyidagi qismda asosiy operatsion tizim xizmatlari va fayl tizimlari haqida ko'proq ma'lumot berilgan:

Asosiy Operatsion Tizim (OS) Xizmatlari:

1. **Protssess boshqarish:** Protssesslar (dasturlar)ni boshqarish uchun xizmatlar, ularni yaratish, boshqarish, to'xtatish va ko'chirishni ta'minlaydi.

2. **Xotira boshqarish:** Xotira boshqarish xizmatlari, xotiraning qo'llanishini boshqarish, xotira yig'ish va boshqarish, asosiy xotira yuklash va tark etishni ta'minlaydi.
3. **Energiya boshqarish:** Energiya boshqarish xizmatlari, kompyuter qurilmasining energiya isrofi va batareyani ta'minlashni boshqarish, batareyani quvvatlanish va energiya saqlashni ta'minlaydi.
4. **Tarmoq boshqarish:** Tarmoq boshqarish xizmatlari, tarmoq interfeyslarini boshqarish, aloqa yo'nalishlarini aniqlash, aloqa jarayonlarini amalga oshirish va boshqarishni ta'minlaydi.
5. **Foydalanuvchi interfeysi (UI):** Foydalanuvchi interfeysi xizmatlari, foydalanuvchi bilan kommunikatsiyani ta'minlash uchun grafik interfeyslarni, ko'rsatkichlar va tugmalar, menyu, shakllar va boshqa elementlarni taqdim etishni boshqarishni ta'minlaydi.
6. **Aniq vaqtni taqdim etish:** Aniq vaqt xizmatlari, kompyuter tizimining joriy vaqtni aniqlash, vaqt o'qish va yozish, vaqt sozlamalarni boshqarishni ta'minlaydi.
7. **Qurilmalar bilan aloqa:** Qurilmalar bilan aloqa xizmatlari, kompyuter tizimining qurilmalarini boshqarish, to'g'ridan-to'g'ri qurilmalar va periferik qurilmalarga murojaat qilishni ta'minlaydi.

Fayl Tizimlari:

1. **Fayl tizimi boshqarish:** Fayl tizimi boshqarish, fayl va kataloglarni yaratish, o'qish, yozish, ko'chirish, o'chirish va boshqarish imkoniyatini ta'minlaydi.
2. **Fayl ishlash:** Fayl ishlash xizmatlari, fayl va kataloglarni o'qish, yozish, ko'chirish, o'chirish va boshqarish uchun xizmatlarni ta'minlaydi.
3. **Fayl himoyalash:** Fayl himoyalash xizmatlari, fayllarni va ma'lumotlarni himoyalash, fayllarga kirishni cheklash va ma'lumotlarni shifrlashni ta'minlaydi.
4. **Fayl tizimi ma'lumotlarini o'qish va yozish:** Fayl tizimi ma'lumotlarini o'qish va yozish xizmatlari, fayllar bilan ishlash uchun funksiyalarni ta'minlaydi.
5. **Fayl tizimini boshqarish:** Fayl tizimi boshqarish xizmatlari, fayllarni jadvallash, izlash, filtirlash va tartiblashni ta'minlaydi.
6. **Fayl tartiblash:** Fayl tartiblash xizmatlari, fayllarni turli usullarda tartiblashni ta'minlaydi, masalan, alifbo, sanalar, kengaytma, kichiklik va boshqalar bo'yicha.

Misol:

Agar siz kompyuter operatsion tizimini loyihalashtirayotgan bo'lsangiz, quyidagi bosqichlar sizga yordam beradi:

1. **Foydalanuvchi interfeysi (UI):** Grafik foydalanuvchi interfeysini (GUI) taqdim etish uchun UI xizmatlaridan foydalaning.
2. **Protsess boshqarish:** Dasturlar va ilovalarni boshqarish uchun protsess boshqarish xizmatlaridan foydalaning.
3. **Fayl tizimi boshqarish:** Fayl tizimini yaratish, fayllarni yuklash, o'qish, yozish, ko'chirish va o'chirishni ta'minlash uchun fayl tizimlari xizmatlaridan foydalaning.
4. **Tarmoq boshqarish:** Kompyuter tarmog'iga ulanish va unga murojaat qilish uchun tarmoq boshqarish xizmatlaridan foydalaning.

Ushbu bosqichlar, kompyuter operatsion tizimi loyihangizda ishlayotgan asosiy operatsion tizim xizmatlarini va fayl tizimlarini boshqarishda foydalanishingiz mumkin.

S73.Arduino tizimi asosida o'rnatilgan tizimlarning apparat vositalarini loyihalashtirish.

Arduino tizimi o'rnatilgan tizimlar loyihalashda keng qo'llaniladi. Arduino platformasi quyidagi asosiy apparat vositalarini o'z ichiga oladi:

1. **Mikroprotssessor:**

- Odatda, 8-bit AVR (ATmega328P) yoki 32-bit ARM (SAM3X8E) mikroprotssessorlari ishlatiladi.
- Tezligi 16 MHz dan 84 MHz gacha bo'lishi mumkin.

2. **Xotira:**

- Flash xotira: 32 KB dan 256 KB gacha
- SRAM: 2 KB dan 96 KB gacha
- EEPROM: 1 KB dan 4 KB gacha

3. **Raqamli va analogli kiritish/chiqarish pinlari:**

- 14 dan 54 taga qadar raqamli kiritish/chiqarish pini
- 6 dan 16 taga qadar analog kiritish pini

4. **Qurilmalar uchun interfeyslari:**

- UART, I2C, SPI kabi standard interfeyslari
- PWM, ADC kabi xususiy modullar

5. **Quvvatlash :**

- Odatda, 5V/3.3V, 1A dan ortiq quvvatlash
- USB orqali quvvatlash yoki tashqi quvvatlash

6. **Tuzilish:**

- Turli o'lchamdagi PCB platalari
- Qurilma korpusi

Arduino platformasining asosiy afzalliklari:

- **Keng tarqalgan:** Juda keng tarqalgan, arzon, oddiy platalar.
- **Oson dasturlash:** C/C++ tillarida dasturlash, Arduino IDE qulay dasturlash muhiti.
- **Keng ko'lamli kutubxonalar:** Sensor, aktuatorlar, kommunikatsiya va boshqa kutubxonalar mavjud.
- **Oson prototiplash:** Tez prototiplash va ishlab chiqish uchun qulay.

Arduino tizimida apparat vositalarini loyihalashda quyidagilarga e'tibor berish kerak:

- Loyiha talablariga moslash: quvvatlash, tezlik, xotira, kiritish/chiqarish.
- Qurilmalarni to'g'ri ulanish va sinxronlash.
- Dasturlash va Arduino kutubxonalaridan foydalanish.
- Fizik o'lchamlarga, PCB plata yul chamlariga e'tibor berish.
- Ishonchlilik, xavfsizlik va energiya tejamkorligini ta'minlash.

Umumiy qilib aytganda, Arduino tizimi o'rnatilgan tizimlar loyihasi uchun juda qulay platforma hisoblanadi.

Arduino tizimi, mikrokontrollerlarga asoslangan va o'z ichiga samarali tarmoq, tizim boshqarish va dasturlashning amaliy vositasi hisoblanadi. Bu tizimlarda yana boshqa apparatlar bilan bog'liq ko'p vositalar ishlay oladi. Quyidagi vositalar ko'p toifali dasturlarni loyihalashtirishda ishlatilishi mumkin:

1. **LEDlar:** LEDlar, elektr energiyasi yoki boshqa signalni korsatish uchun keng qo'llaniladi.
2. **Sensorlar:** Sensorlar, harorat, yorug'lik, tovush va boshqa o'zgaruvchilarni kuzatish uchun ishlatiladi.
3. **Motorlar:** Motorlar, mexanik harakatni amalga oshirish uchun ishlatiladi, masalan, dc motorlar, servomotorlar va qo'l ajratgichlar.
4. **Relaylar:** Relaylar, elektr ustida ishlaydigan buyruqlarni amalga oshirish uchun ishlatiladi.
5. **LCD ekranlar:** LCD ekranlar, ma'lumotlar va xabarlar ko'rsatish uchun ishlatiladi.
6. **Bluetooth va Wi-Fi modullari:** Bu modullar, arduino tizimini boshqa qurilmalar va tarmoqqa bog'lash uchun ishlatiladi.
7. **SD kartalari:** SD kartalari, ma'lumotlarni saqlash va o'qish uchun ishlatiladi.
8. **RTC (Real-Time Clock) modullari:** RTC modullari, vaqtni saqlash va amalga oshirish uchun ishlatiladi.
9. **Servo motorlar:** Servo motorlar, aniqlikni boshqarib harakatni amalga oshirish uchun ishlatiladi.
10. **GPS modullari:** GPS modullari, geografik o'rinlarni aniqlash va ularga murojaat qilish uchun ishlatiladi.
11. **Step motorlar:** Step motorlar, to'g'ri yo'naltirish uchun ishlatiladi.

Bu faoliyatlar asosida Arduinoga bog'liq ko'p loyihalar va vositalar yaratilmoqda. Masalan, temperatur, harorat, yorug'lik, vaqtni avtomatik ravishda boshqarish, Smart home loyihalari, Robotlar, qidirish va saqlash tizimlari, ozbek liboslari qilish, musiqa qurilmalari va ko'p qo'llanmalarga ko'ra dasturlar yaratish mumkin. Arduining oddiy dasturlash va qulayliklari bilan, bu tizimni o'rganish eng oson va ilg'or yo'l bo'ladi.

S74.Arduino mikrokontrollerlari.

Arduino mikrokontrollerlari o'rnatilgan tizimlar loyihalashda juda keng qo'llaniladi. Ular quyidagi asosiy xususiyatlarga ega:

1. **Mikroprotsessorlar:**
 - Asosan, 8-bit AVR (ATmega328P) yoki 32-bit ARM (SAM3X8E) mikroprotsessorlari ishlatiladi.
 - Tezligi 16 MHz dan 84 MHz gacha bo'lishi mumkin.
2. **Xotira:**
 - Flash xotira: 32 KB dan 256 KB gacha
 - SRAM: 2 KB dan 96 KB gacha
 - EEPROM: 1 KB dan 4 KB gacha
3. **Kiritish/Chiqarish pinlari:**
 - 14 dan 54 taga qadar raqamli kiritish/chiqarish pini
 - 6 dan 16 taga qadar analog kiritish pini
4. **Standart interfeyslari:**
 - UART, I2C, SPI kabi standard interfeyslari
 - PWM, ADC kabi xususiy modullar
5. **Quvvatlash:**
 - Odatda, 5V/3.3V, 1A dan ortiq quvvatlash
 - USB orqali quvvatlash yoki tashqi quvvatlash
6. **Tuzilish:**
 - Turli o'lchamdagi PCB platalari
 - Qurilma korpusi

Arduino mikrokontrollerlarining asosiy turlari:

1. **Arduino Uno:** 8-bit ATmega328P mikroprotsessori, 14 raqamli pin, 6 analog pin, 32 KB flash xotira.
2. **Arduino Mega 2560:** 8-bit ATmega2560 mikroprotsessori, 54 raqamli pin, 16 analog pin, 256 KB flash xotira.
3. **Arduino Due:** 32-bit Atmel SAM3X8E ARM Cortex-M3 mikroprotsessori, 54 raqamli pin, 12 analog pin, 512 KB flash xotira.
4. **Arduino Nano:** 8-bit ATmega328 mikroprotsessori, 14 raqamli pin, 8 analog pin, 32 KB flash xotira.
5. **Arduino Leonardo:** 8-bit ATmega32u4 mikroprotsessori, 20 raqamli pin, 12 analog pin, 32 KB flash xotira.
6. **Arduino Yun:** 8-bit ATmega32u4 va 400 MHz AR9331 mikrokontrollerlari, Ethernet, Wi-Fi va USB bilan uyg'unlashtirilgan.

Arduino mikrokontrollerlari orasida Uno va Mega eng ko'p qo'llaniladi. Ular bir-biridan tezligi, xotirasining hajmi, kiritish/chiqarish pinlari soni bilan farqlanadi. Arduino platformasining keng tarqalganligi, qulay dasturlash muhiti va keng ko'lamli kutubxonalar mavjudligi uni o'rnatilgan tizimlar loyihalashda juda qulay qiladi.

Arduino mikrokontrollerlari, elektronika va dasturlash sohasida ta'minotchilar, talabalar, va hobbistlar o'rtasida mashhur bo'lgan Arduinoning asosiy qismidir. Bu mikrokontrollerlar, ma'lumotlarni o'qish, hisoblash va harakatlar amalga oshirish uchun moslashtirilgan toifadagi apparatlar hisoblanadi. Arduinoning eng mashhur qismlaridan ba'zi majmualar quyidagilardir:

1. **Arduino Uno:** Arduino Uno, dasturlash uchun ommabop bo'lgan va o'rganishni osonlashtiradigan toifadagi modeldir. 14 pinli, USB porti, 6 analog kiritish, 6 dijital kiritish/chiqish portlari va boshqa yordamchi funksiyalarga ega.
2. **Arduino Nano:** Arduino Nano, kichik hajmdagi qulay dasturlash uchun mo'ljallangan toifa. Uni qanday to'g'ri yoki yo'nalish, dasturlarni tekshirish, va boshqa kamchiliklar uchun yaxshi foydalanish mumkin.
3. **Arduino Mega:** Arduino Mega, ko'p ulgurji dasturlash loyihalarini boshqarish uchun xizmat qiladi. Uda, kiritish/chiqish portlari, dasturiy UART, SPI va I2C interfayslar bilan birga 54 dijital kiritish va 16 analog kiritish portlari mavjud.
4. **Arduino Due:** Arduino Due, ARM Cortex-M3 mikrokontrolleri bilan taminlangan va samarali dasturlash uchun mo'ljallangan model. Uni tajribali foydalanuvchilar va keng doirasidagi loyihalar uchun mo'ljallangan bo'lib, bir nechta katta ulg'ayish va kiritish/chiqish portlari mavjud.
5. **Arduino Mini:** Arduino Mini, kichik hajmdagi boshqaruvchi mikrokontroller hisoblanadi, lekin uning quyidagi funksiyalari mavjud: 14 pinli, USB porti, 6 analog kiritish, 6 dijital kiritish/chiqish portlari.
6. **Arduino Leonardo:** Arduino Leonardo, ATmega32U4 mikrokontroller bilan taminlangan, samarali dasturlash va boshqarish uchun mo'ljallangan modeldir. Bu modelda 20 dijital kiritish/chiqish portlari va 12 analog kiritish portlari mavjud.
7. **Arduino Pro Mini:** Arduino Pro Mini, qulaylik va qulaylik uchun dizayn qilingan modeldir. U kompakt hajmi va samarali ishlovchi xususiyatlariga ega bo'lgan oddiy va yuzlablar dasturlashni o'rganish uchun mo'ljallangan.
8. **Arduino Nano Every:** Arduino Nano Every, ATmega4809 mikrokontrolleri bilan taminlangan va kichik hajmdagi Arduinolardan biridir. Bu model, keng qamrovli dasturlash loyihalari uchun mo'ljallangan va uni barcha odatiy qurilmalar bilan foydalanish imkonini beradi.
9. **Arduino MKR1000:** Bu model WiFi va Bluetooth ta'minotlari bilan taminlangan va IoT loyihalari uchun mo'ljallangan. Uni internetga bog'lanishni yoki boshqa qurilmalar bilan bog'lanishni yordamchi modulalar mavjud.
10. **Arduino Nano 33 IoT:** Bu Arduino modeli, WiFi, Bluetooth, low energy Bluetooth (BLE) va 9-aksli IMU bilan ta'minlangan kompakt va samarali dasturlash uchun mo'ljallangan.

Bu Arduinoning qanday va qaysi mikrokontrolleri kerakligiga qarab qaror qilinganini xisobga olgan holda, ular o'rganish, tajribani oshirish va qiziqarli loyihalarni boshlash uchun yaxshi imkoniyatlar ta'minlaydi.

S75.Arduino mikrokontrolleriga asoslangan avtomatlashtirilgan tizimlar

Arduino mikrokontrollerlari avtomatlashtirilgan tizimlarni loyihalashda keng qo'llaniladi. Arduino platformasi quyidagi kabi avtomatlashtirilgan tizimlar yaratishda qulay:

1. Uy avtomatikasi tizimlari:

- Yorug'lik, muzlatgich, eshiklar va boshqalar ustidan nazorat.
- Issiqlik, namlik, harorat kabi parametrlarni monitoring qilish.
- Alarm tizimlarini boshqarish.

2. Sanoat avtomatizatsiyasi:

- Sensor va aktuatorlarni nazorat qilish.
- Texnologik jarayonlarni monitoring va boshqarish.
- Ishlab chiqarish xavfsizligini ta'minlash.

3. Qishloq xo'jaligi avtomatikasi:

- Suv, o'g'it va dorilarni boshqarish.
- Yog'in-sochin, harorat va namlikni monitoring qilish.
- Yalpi hosildorlikni oshirish.

4. Transport avtomatikasi:

- Transport vositalarini nazorat qilish va boshqarish.
- Yo'llar va transport harakatini monitoring qilish.
- Transport xavfsizligini ta'minlash.

5. Robototexnika va dronlar:

- Manipulyatorlar va qo'l-oyoq harakatlarini boshqarish.
- Sensor ma'lumotlarini qayta ishlash va sharhlash.
- Navigatsiya va avtopilot tizimlarini amalga oshirish.

6. Kommunal xizmatlar avtomatikasi:

- Suv, gaz va elektr tizimlarini nazorat qilish.
- Yo'l yorug'ligi, chiqindilar va boshqalarni boshqarish.
- Resurs sarfini kamaytirish va tejamkorlikni ta'minlash.

Arduino mikrokontrollerlari apparat va dasturiy ta'minotni oson integratsiya qilish imkonini beradi. Ular keng tarqalgan, arzon, ishonchliligi yuqori va dasturlanishi oson bo'lganligi uchun avtomatlashtirilgan tizimlarni loyihalashda juda qo'l keladi.

Arduino mikrokontrollerlari, avtomatlashtirilgan tizimlarni o'rganish va amalga oshirish uchun yaxshi asos bo'lib xizmat qiladi. Ular yordamida bir nechta turdagi avtomatlashtirilgan tizimlar yaratish mumkin. Quyidagi misollar avtomatlashtirilgan tizimlar uchun Arduino mikrokontrollerlarini qo'llashning muhim mavzularini ta'kidlashni maqsad qilgan:

1. Smart Home Tizimlari:

- **Qulayliklar Boshqaruvi:** Arduino orqali yonuvchi yo'l, og'irlik, va yorug'likning avtomatik boshqarilishini ta'minlash mumkin. Misol

uchun, xonadon ichida yorug'likni o'tkazish va qaytarish uchun avtomatik tizim.

- **Havo Tizimi:** Arduinoni temperatur, namlik, va havo sharoitini o'qib, qurilma ishini boshqarish uchun ishlab chiqish mumkin.
- **Xavfsizlik Tizimi:** Ko'chaga kirish/chiqishlar, oynalar, yonuvchilar va boshqa xavfsizlik qurilmalari uchun Arduinoni qo'llab-quvvatlash mumkin.

2. Avto Tizimlar:

- **Yordamchi Parklov:** Parklov sensorlari va motorlar orqali avtomobilni parklov qilish.
- **Qulayliklar:** Muhim talablar uchun kalendar va eslatma vositalari bilan aloqa.
- **Tezlik va Yukseklik Boshqaruv:** Arduinoni tezlik va yuksaklikning avtomatik boshqarilish tizimini yaratish.

3. Fermerlik Tizimlari:

- **Su Darajasi Boshqaruv:** Arduino mikrokontrollerlar orqali avtomatik su boshqaruv tizimi yaratish.
- **Temperatur va Namlik Boshqaruv:** Kichik fermada temperatur va namlikni avtomatik boshqarish.

4. Oziq-ovqat Tizimlari:

- **Vaqtni Boshqarish:** Oziq-ovqat saqlash va vaqtida chiqarish uchun tizim yaratish.
- **Temperatur va Namlikni Boshqaruv:** Yengil tomizdan foydalanib temperatur va namlikni boshqaruv tizimi.

5. Oziq-ovqat Tizimlari:

- **Vaqtni Boshqarish:** Oziq-ovqat saqlash va vaqtida chiqarish uchun tizim yaratish.
- **Temperatur va Namlikni Boshqaruv:** Yengil tomizdan foydalanib temperatur va namlikni boshqaruv tizimi.

6. Haroratni Boshqaruv Tizimlari:

- **Konditsioner Boshqaruv:** Arduinoni konditsioner va issiqlik qurilmalarini avtomatik ravishda boshqarish.
- **Bo'ron tizimi:** Haroratni boshqarib, isitish va sovurishni boshqarish.

7. Oqimli Tizimlar:

- **Ma'shina Boshqarish:** Arduinoni ma'shina qismi va sensorlar orqali ma'shinani avtomatik ravishda boshqarish.
- **Robotlar:** Arduino yordamida robotlar va mekhatronika qurilishlarini boshqarish.

Bu hamma misollardan foydalanib, Arduino mikrokontrollerlarini, yangi tarmoq, avtomatlashtirilgan tizimlar yaratishda mashxur qilish mumkin. Bu turdagi loyihalar, elektronika va dasturlash sohasida qiziqarli va maqbuldir, chunki ular

ko'plab sohalarga taalluqli bo'lishi mumkin va ko'p toifali mahsulotlar ishlab chiqarish imkonini beradi

S76. MK Arduino asosida jarayonlarni avtomatlashtirish.

Arduino mikrokontrollerlari o'rnatilgan tizimlar loyihalashda juda keng qo'llaniladi. Ular apparat va dasturiy ta'minotni oson integratsiya qilish imkonini beradi. Arduino platformasi turli xil jarayonlarni avtomatlashtirish uchun quyidagi imkoniyatlarga ega:

1. **Sensor ma'lumotlarini to'plash va monitoring qilish:**

- Harorat, namlik, bosim, yoritilganlik, oqim va boshqa ko'rsatkichlarni o'lchash.
- Sensorlarni Arduino pinlari orqali o'qish va ma'lumotlarni qayta ishlash.
- Ob-havo, suv sathi, ekologiya kabi parametrlarni onlayn kuzatish.

2. **Aktuatorlarni nazorat qilish:**

- Motorlar, elektr klapanlar, relele, yoritish qurilmalari va boshqalarni boshqarish.
- PWM signallaridan foydalanib, aktuatorlarni doimiy ravishda moslashtirish.
- Xavfsizlik va energiya tejamkorligi talablarini amalga oshirish.

3. **Kommunikatsiya va ma'lumot uzatish:**

- UART, I2C, SPI, Ethernet, Wi-Fi kabi standart protokollardan foydalanish.
- Ma'lumotlarni PC, server yoki bulutli platformalarga uzatish.
- Mobil ilovalar yoki veb-xizmatlar bilan aloqa o'rnatish.

4. **Yangi vaziyatlarga moslanish:**

- Mikrokontroller dasturini yangilash va o'zgartirish imkoniyati.
- Qurilmalarni osongina almashtirish va yengillashtirish.
- Zamonaviy texnologiyalar bilan birgalikda qo'llash.

5. **Oson dasturlash va prototiplab yaratish:**

- Bepul va ochiq kodli Arduino IDE dasturiy ta'minoti.
- Katta jamoat tomonidan qo'llab-quvvatlanib, ko'plab kutubxonalar mavjud.
- Prototiplab sinab ko'rish uchun qulay va tezkor.

Arduino mikrokontrollerlari arzon, keng tarqalgan va ishonchli bo'lganligi uchun turli xil jarayonlarni avtomatlashtirish uchun juda qo'l keladi. Ular real vaqtdagi ma'lumotlarni yig'ish, boshqarish, kommunikatsiya o'rnatish va yangi vaziyatlarga moslashish imkonini beradi.

Arduino MKR seriyasida qo'llanadigan mikrokontrollerlar, IoT (Internet of Things) loyihalari uchun mo'ljallangan va avtomatlashtirilgan jarayonlarni amalga oshirishga yordam beradi. Bu mikrokontrollerlar, WiFi yoki Bluetooth modullari bilan ta'minlangan, keng qamrovli loyihalarni dasturlash uchun ideal vositalardir. Quyidagi misollar MKR Arduino asosida jarayonlarni avtomatlashtirishda ishlatiladigan foydali funksiyalardan ba'zi ko'rsatilgan:

1. **MKR WiFi 1010:**

- **WiFi Ulashuvi:** WiFi moduli orqali IoT qurilmalarini WiFi tarmoqlariga bog'lash.
- **Sensorlar bilan Bog'lanish:** Harorat, namlik, va havo sharoitini o'qib sensorlar orqali bog'lanish.
- **Avtomatlashtirish va Monitoring:** Harorat, yorug'lik, va namlikning avtomatik monitoring va boshqarilishi.

2. **MKR GSM 1400:**

- **Mobil Aloqa:** GSM moduli orqali IoT qurilmalarini mobil aloqa tarmoqlariga bog'lash.
- **Mahsulot Monitoring:** Xaridorlar ma'lumotlari, mahsulotlarning monitoringi va avtomatik buyurtma berish.

3. **MKR NB 1500:**

- **Narrowband IoT Ulashuvi:** NB-IoT moduli orqali IoT qurilmalarini telecom tarmoqlariga bog'lash.
- **Mahsulot Monitoring:** Uzlüksiz monitoring va sensorlar orqali mahsulot holati monitoringi.

4. **MKR FOX 1200:**

- **Sigfox Ulashuvi:** Sigfox moduli orqali IoT qurilmalarini Sigfox tarmoqlariga bog'lash.
- **Batareya Monitoring:** Batareyani o'z vaqtida monitoring qilish va batareya statusini avtomatik ravishda boshqarish.

5. **MKR WAN 1300:**

- **LoRa Ulashuvi:** LoRa moduli orqali IoT qurilmalarini LoRaWAN tarmoqlariga bog'lash.
- **Harorat Monitoring:** Harorat, namlik va havo sharoitini o'qib, monitoring qilish va boshqarish.

6. **MKR Zero:**

- **Yengil Dasturlash:** Zero moduli, bas icchiq mavjudiyat bilan energiya xarajatini kamaytirib, avtomatik harakatlarni boshqarish uchun yengil dasturlash imkonini beradi.
- **IoT Monitoring:** Sensorlar orqali ma'lumotlar oqib, IoT monitoring tizimlarini dasturlash.

Bu misollar, Arduino MKR seriyasidagi mikrokontrollerlarni avtomatlashtirilgan IoT tizimlarini yaratish uchun qanday qilib foydalanishingizni ko'rsatadi. Bu mikrokontrollerlar, keng qamrovli IoT loyihalarini boshqarish, harorat, namlik, yorug'lik va boshqa o'zgaruvchilarni avtomatik ravishda boshqarish, va mahsulot monitoringi kabi ko'plab vazifalarni bajarish uchun samarali vositalardir.

S77. O'rnatilgan tizimlarning dasturiy ta'minoti.

O'rnatilgan tizimlar, shu jumladan Arduino asosidagi tizimlar, ularning ish faoliyatini ta'minlaydigan maxsus dasturiy ta'minotga ega bo'ladi. Bu dasturiy ta'minotlar quyidagi funktsiyalarni amalga oshiradi:

1. **Apparat ta'minotni nazorat qilish:**
 - Sensorlarni o'qish, aktuatorlarni boshqarish.
 - Quvvat boshqaruvi, xavfsizlik cheklovlari.
 - Vaqt va kalendar funktsiyalarini amalga oshirish.
2. **Dasturiy ta'minotni boshqarish:**
 - Mikrokontroller dasturini yozish, yangilash, debug qilish.
 - Ob'ektga yo'naltirilgan yoki modulli dasturlash.
 - Qo'shimcha kutubxonalar va drayverlari yuklash.
3. **Ma'lumot boshqaruvi:**
 - Sensor ma'lumotlarini yig'ish, qayta ishlash, saqlab qolish.
 - Mahsulot parametrlarini tasdiqlash va nazorat qilish.
 - Bulutli platformalar yoki boshqa tizimlar bilan aloqa o'rnatish.
4. **Foydalanuvchi interfeysi:**
 - Mahsulot parametrlarini sozlash va boshqarish.
 - Ogohlantirish va xizmat ko'rsatish xabarlarini ko'rsatish.
 - Mobil ilovalar yoki web-saytlar orqali boshqarish.
5. **Xavfsizlik va xizmat ko'rsatish:**
 - Xavfsizlik siyosatlarini amalga oshirish.
 - Uzluksiz ishlashini ta'minlash, nosozliklarni bartaraf etish.
 - Yangilanishlarni avtomatik yoki qo'lda qabul qilish.

O'rnatilgan tizimlarning dasturiy ta'minoti, shu jumladan Arduino mikrokontrollerlari uchun yozilgan dasturlar, o'z funktsionalligi, doimiyliigi, xavfsizligi va xizmat ko'rsatishi bilan ajralib turadi. Bu dasturiy ta'minotlar qurilmalarning samarali va ishonchli ishlashini ta'minlaydi.

O'rnatilgan tizimlar dasturiy ta'minoti, ularga qarashli xizmatlarni amalga oshirishda o'rin olgan dasturlar, ilovalar va qo'shimcha xizmatlar to'g'risida ma'lumotlarni o'z ichiga oladi. Dasturiy ta'minot, bitta operatsion tizimdan boshqa o'giriladigan dasturlar va qo'shimcha yordamchi dasturlarni o'z ichiga oladi. Quyidagi misollar o'rnatilgan tizimlarning dasturiy ta'minotini aks ettiradi:

1. **Ma'muriy panel (Control Panel):** Foydalanuvchi uchun o'rnatilgan tizimni boshqarish uchun ma'muriy panel o'rnatish tavsiya etiladi. Bu panel o'rnatilgan tizimni boshqarishga yordam beradi, sozlashlar va konfiguratsiyalarni o'zgartirishga imkon beradi.
2. **Bundan tashqari, operatsion tizimni boshqarish uchun:**

- **Yozilgan dasturlar va ilovalar:** Foydalanuvchilarga xizmat qilish uchun o'rnatilgan dasturlar va ilovalar.
 - **Operatsion tizimni sozlash qo'llanmalari:** Operatsion tizimning ichki sozlashlarini o'zgartirishga imkon beradigan qo'llanmalar.
 - **Dasturiy ta'minot yordamchilari:** Foydalanuvchilarga o'rnatilgan tizimlarni o'rganish va ularga o'rganishga yordam beradigan yordamchilar, o'rganish materiallari, onlayn yordam va ko'rsatmalar.
3. **Xizmat dasturlari:** O'rnatilgan tizimga bog'liq bo'lgan yordamchi xizmat dasturlari, masalan, o'rnatilgan tizim uchun murakkab ta'minotni saqlash, monitoring va diagnosika, ta'minot sozlashlari va boshqalar.
 4. **Qo'shimcha dasturlar va ilovalar:** Operatsion tizimga qo'shimcha xizmatlarni taqdim etish uchun qo'shimcha dasturlar va ilovalar, masalan, ko'p qo'llaniladigan dasturlar, turli xil ta'minotlarni qo'shish va boshqalar.
 5. **O'rnatilgan tizim uchun qo'shimcha xizmatlar va yordamchilar:** Foydalanuvchilarga o'rnatilgan tizimni o'rganish va unga murojaat qilish uchun ko'rsatmalarga qo'shimcha xizmatlar va yordamchilar.
 6. **O'rnatilgan tizimni saqlash uchun yedeklovchi vositalar:** O'rnatilgan tizimning ma'lumotlarini saqlash va takomillashtirish uchun yedeklovchi dasturlar va vositalar.

O'rnatilgan tizimlarning dasturiy ta'minoti, foydalanuvchilarga ma'lumotlarni amalga oshirish va boshqarishga imkon beruvchi dasturlar, qo'shimcha xizmatlar va yordamchilar o'z ichiga oladi. Bu ta'minot o'rnatilgan tizimni samarali ishlatishga imkon beradi va foydalanuvchilarga ma'lumotlarini boshqarish va boshqarish imkoniyatlarini yaxshilaydi.

S78.Mikrokontrollerli tizimlarni sozlash.

Mikrokontrollerli tizimlar, shu jumladan Arduino asosidagi tizimlar, ularni osonroq sozlash va moslash uchun bir qator funktsiyalarga ega:

1. Apparat ta'minoti sozlamalari:

- Mikrokontroller tezligi, xotira hajmi, USB kabi parametrlarni tanlov.
- Pin konfiguratsiyasi: kiritish/chiqarish, analog/raqamli, PWM.
- Sensorlar, aktuatorlar va boshqa periferiya qurilmalarini ulanishi.

2. Dasturiy ta'minot sozlamalari:

- Arduino IDE yoki boshqa dasturlash muhitini o'rnatish va sozlash.
- Mikrokontroller dasturini yozish, yangilash, sinab ko'rish.
- Kerakli kutubxonalar va drayverlari yuklash va ulash.

3. Foydalanuvchi interfeysi:

- LCD, LED, tugmalar yoki boshqa elementlar orqali interaktiv interfeys yaratish.
- Sensor ko'rsatkichlarini, xizmat xabarlarini, sozlamalarni ko'rsatish.
- Mikrokontroller parametrlarini foydalanuvchi tomonidan o'zgartirish.

4. Ma'lumot ulash va kommunikatsiya:

- UART, I2C, SPI, Ethernet, Wi-Fi kabi aloqa protokollarini sozlash.
- Mikrokontroller va boshqa tizimlar o'rtasida ma'lumot almashinuvi.
- Bulut xizmatlar yoki mobil ilovalar bilan bog'lanish.

5. Quvvat boshqaruvi:

- Quvvat manbalari, batareyalar, quvvat tejash rejimlarini sozlash.
- Uzluksiz ishlashni ta'minlash uchun quvvat taqsimoti.
- Foydalanuvchi talablariga moslash va samaradorlikni oshirish.

Mikrokontroller tizimlarini sozlash, ularning apparat va dasturiy ta'minoti, foydalanuvchi interfeysi, kommunikatsiya va quvvat boshqaruvi kabi komponentlarini moslash orqali, qurilmalarni samarali, ishonchli va maksimal darajada moslashtirilgan holda ishlashini ta'minlaydi.

Mikrokontrollerli tizimni sozlash jarayoni, avvalo, kerakli vositalarni tanlash va dasturiy ta'minotni o'rnatishni o'z ichiga oladi. Keyin esa, tizimni dasturlash uchun muhim parametrlarni belgilash, asosiy funktsiyalarni amalga oshirish, dasturiy xatolarni aniqlash va tuzatish, sozlash va optimallashtirish qat'iyot bilan o'tkaziladi. Quyidagi bosqichlar mikrokontrollerli tizimni sozlash jarayonida odatda muhim bo'ladi:

1. Kerakli Vositalarni Tanlash:

- Mikrokontroller: Mos mikrokontroller turini tanlash, masalan, Arduino, Raspberry Pi, ESP32, STM32 kabi.
- Sensorlar: Ma'lumotlarni o'qish uchun zarur sensorlarni tanlash, masalan, temperatur sensori, harorat sensori, yorug'lik sensori va boshqalar.

- Aktiv vositalar: Jarayonni amalga oshirish uchun zarur aktuatorlarni tanlash, masalan, motorlar, relenlar, LEDlar, va boshqalar.
- 2. Dasturiy Ta'minotni O'rnatish:**
- Boshlang'ich Dasturlash Muhitini O'rnatish: Tanlangan mikrokontroller uchun mos dasturlash muhiti (masalan, Arduino IDE, PlatformIO, ke'da dasturlash muhiti) o'rnatish.
 - Dasturlarni Yuklash: O'zlashtirilgan vositalar uchun dasturlarni yuklash va tayyorlangan dasturlardan foydalanish.
- 3. Asosiy Funksiyalarni Amalga Oshirish:**
- Harorat, namlik, yorug'lik va boshqa o'zgaruvchilarni o'qish uchun sensorlarni qo'llash.
 - Boshlang'ich sozlash va konfiguratsiyalarni o'zlashtirish.
 - Ma'lumotlarni o'qish va ularni ko'rish.
 - Sensorlardan olingan ma'lumotlarga asoslangan qarorlarni qabul qilish va ularni amalga oshirish.
 - Aktuatorlarni boshqarish va harakatlar amalga oshirish.
- 4. Dasturiy Xatolarni Aniqlash va Tuzatish:**
- Dasturiy xatolarni aniqlash uchun debug chipti va bog'lovchi dasturlashda qo'llaniladigan qo'shimcha vositalardan foydalanish.
 - Dasturda aniqlangan xatolarni to'g'rilash va tuzatish.
- 5. Sozlash va Optimallashtirish:**
- Sozlashlar va konfiguratsiyalarni o'zgartirish va mikrokontrollerli tizimni sozlash.
 - Dastur va algoritmlarni optimallashtirish uchun kodni tezlashtirish va yengillashtirish.
- 6. Test qilish va taqdim etish:**
- Tizimni test qilish va uning ishlashini tekshirish.
 - Foydalanuvchilarga taqdim etish uchun tizimni tayyorlash va tizimni tanishtirish.

Mikrokontrollerli tizimni sozlash, qurilish bosqichidan dasturiy ta'minot o'rnatish va tizimning to'liq ishlashi uchun dasturlarni yuklash va test qilish jarayonidan o'tkaziladi. Bu jarayon, avvalgi tanlov, sozlash, dasturlash va test qilish bosqichlaridan iborat

S79 Arduino platformalarining turlari.

Arduino - mikroprotsessor asosida qurilgan va ochiq kodli elektron platforma bo'lib, uning bir nechta turlari mavjud:

1. Klassik Arduino platalari:

- Arduino Uno, Nano, Mega, Leonardo kabi eng mashhur modellar.
- AVR mikrokontrollerlari (ATmega328P, ATmega2560) asosida qurilgan.
- Turli o'lcham va xususiyatlarga ega, umumiy Arduino SDK bilan ishlaydi.

2. ARM asosidagi Arduino platalari:

- Arduino Due, Zero, 101, 33 kabi modellar.
- 32-bit ARM Cortex-M mikroprotsessorlari (SAM, nRF52) asosida.
- Kuchli hisoblash va ulash imkoniyatlariga ega.

3. WiFi/Bluetooth Arduino platalari:

- Arduino Yun, MKR1000, Nano 33 IoT kabi platalarda WiFi moduli bor.
- Arduino 101, Nano 33 BLE kabi platalarda Bluetooth modulli.
- Tarmoqlarga ulash va mobil aloqa uchun qulay.

4. Maxsus maqsadli Arduino platalari:

- Arduino Gemma, Trinket, Lily Pad - kichik o'lchamli modifikatsiyalar.
- Arduino Industrial 101, Pro/Pro Mini - sanoat yoki professional qo'llanish uchun.
- Arduino Portenta H7 - kuchli protsessor, keng imkoniyatlar.

5. Maxsus Arduino klonlari:

- SparkFun, Adafruit, Seeed kabi kompaniyalarning Arduino klonlari.
- Turli xil qo'shimcha imkoniyatlari, interfeyslari bor.
- Arduino SDK va ilovalarini mos keluvchi qurilishiga ega.

Arduino platformasining ushbu turli xil turlari, turli maqsadlar, odam toifasi va imkoniyatlar uchun mo'ljallangan. Foydalanuvchining talablariga qarab, mos keluvchi Arduino modelini tanlash mumkin.

Arduino dasturlash platformalari, turli maqsadlar va talablarga moslashtirilgan mikrokontrollerli tizimlarni dasturlash uchun moslashtirilgan turli vositalardir. Quyidagi Arduino platformalari eng mashhur va keng qo'llaniladigan turlardan ba'zilaridir:

1. Arduino Uno:

- Arduino Uno, asosiy va eng mashhur Arduino mikrokontrolleridir.
- Uno, ma'lumotlarni o'qish va yozish uchun 14 ta pin bilan ta'minlangan.
- Bu platforma, boshlang'ich va o'rtacha darajadagi dasturlarni yaratish uchun moslashadi.

2. Arduino Mega:

- Arduino Mega, ko'p ta pinlarga ega bo'lgan va ko'p ta sensor va aktuatorlarni qo'llab-quvvatlaydigan katta miqdorda portlarga ega.
 - Uni keng ko'lamli dasturlash loyihalari uchun yaxshi tanlash mumkin.
- 3. Arduino Nano:**
- Arduino Nano, kichik hajmli va qulay dasturlash uchun mo'ljallangan.
 - Bu model, Arduino Uno'ga o'xshash funksiyalarni taqdim etadi, ammo kichik hajmi va to'g'riyoq loyihalarni yaratishga yordam beradi.
- 4. Arduino Leonardo:**
- Arduino Leonardo, ATmega32U4 mikrokontrolleri bilan taminlangan, boshqa Arduino modellari bilan farq qiladigan keyingi darajada funksiyalarga ega.
 - Bu modelda USB klaviatura va mouse emulatorlarini boshqarishga imkon beradi.
- 5. Arduino Due:**
- Arduino Due, ARM Cortex-M3 mikrokontrolleri bilan taminlangan.
 - Bu model, yuqori darajadagi dasturlash uchun ishlab chiqilgan va ko'p ta sensor va aktuatorlarni qo'llash imkonini beradi.
- 6. Arduino MKR Series:**
- Arduino MKR seriyasi, IoT loyihalari uchun mo'ljallangan, WiFi, Bluetooth yoki LoRa modullari bilan ta'minlangan.
 - MKR seriyasidagi modellar, sensorlardan ma'lumotlarni o'qish va internetga ulanish uchun qulayliklar taqdim etadi.
- 7. Arduino Pro Mini:**
- Arduino Pro Mini, soddaligi va kichik hajmi tufayli ko'plab joylarda qo'llaniladi.
 - Bu model, odatiy dasturlash loyihalari uchun mo'ljallangan.
- 8. Arduino Yun:**
- Arduino Yun, mikrokontroller va Linux protsessori birlashmasidir.
 - Bu platforma, Internetga bog'lanish va bulut xizmatlariga kirish imkonini beradi.
- 9. Arduino Nano Every:**
- Arduino Nano Every, ATmega4809 mikrokontrolleri bilan taminlangan.
 - Bu model, ko'p funksiyalarga ega va Arduino Nano bilan o'xshash o'z hajmini taqdim etadi.
- 10. Arduino Robot:**
- Arduino Robot, robotlar va mekhatronika dasturlash loyihalari uchun mo'ljallangan.
 - Bu platforma, robotlar yaratish va ulardan foydalanishga imkon beradi.

Bu, faqatgina bir qancha mashhur Arduino platformalardir. Har biri, maqsad va talablarga moslashtirilgan turli vositalarga ega, shuningdek, mavjud tizimlarni dasturlash va IoT loyihalari uchun yaxshi tanlovdir.

S80. Shaxsiy kompyuterida testlarni o'tkazish.

Shaxsiy kompyuteringizda turli xil testlarni o'tkazish uchun quyidagi yondashuv va usullardan foydalanish mumkin:

1. Tizim diognostikasi:

- CPU, xotira, disk, tarmoq kabi komponentlarni tekshirish.
- Windows Diagnostics Tool, CPU-Z, HWMonitor kabi dasturlardan foydalaning.
- Tizim barqarorligini, ishlash tezligini, qizish darajasini baholash.

2. Disk va xotira testlari:

- HDD/SSD sog'lom ekanligini sinab ko'rish.
- CrystalDiskInfo, HD Tune Pro kabi dasturlar yordamida tekshiring.
- Tezlikni, xatoliklar va yomon sektorlarni aniqlash.

3. Tarmoq sinovlari:

- Internetga ulanish tezligini, ping, paket yo'qotilishi kabi ko'rsatkichlarni tekshiring.
- Speedtest, PingPlotter, Network Ping Tester dasturlari yordamida.
- Wi-Fi sifati, tarmoq keskinligi va boshqa parametrlarni baholang.

4. Quvvat ta'minoti (PSU) sinovlari:

- PSU samaradorligi, o'zgaruvchan kuchlanish, qizish kabi xususiyatlarini tekshiring.
- CPUID HWMonitor, Systool kabi dasturlar yordamida.
- Kompyuter turli yuklamalar ostida barqaror ishlashini sinab ko'ring.

5. Grafik va video sinovlari:

- 3DMark, FurMark, Cinebench kabi dasturlar yordamida grafikani baholang.
- Video ko'rsatkichlarni, video xotira tezligini sinab ko'ring.
- Yog'ma va issiqlik parametrlarini ham kuzatib boring.

Testlarni o'tkazish orqali siz kompyuteringizning holati, ishlash samaradorligi va muammolarini aniqlay olasiz. Bu ma'lumotlar asosida yanada optimallashtirish va yangilash ishlari olib borish mumkin.

Shaxsiy kompyuteringizda testlarni o'tkazish uchun turli usullar mavjud bo'lib, kompyuteringizning faoliyatini tekshirish va to'g'ri ishlashi uchun muhim bo'lishi mumkin. Quyidagi asosiy testlar kompyuteringizning ustuvor taraflarini, xotirani, va faoliyatini tekshirishga yordam berishi mumkin:

1. Antivirus Skaneri:

- Antivirus dasturi orqali kompyuteringizni viruslar, malware, va boshqa zararli dasturlarga qarshi tekshirish.

- Qo'shimcha ma'lumotlar uchun, kompyuteringizga virus skaneri dasturlar, masalan, Avast, Kaspersky, Norton, Malwarebytes, kabi yuklab oling.
- 2. **Disk Tekshiri:**
 - Kompyuteringizdagi diskni tekshirib ko'rish uchun murakkab dasturlardan foydalaning. Masalan, Windows uchun "Disk Check" yoki "Chkdsk" dasturi mavjud.
- 3. **Sistem Tekshiri:**
 - Operatsion tizimning to'la ishlashi uchun "System File Checker" (SFC) dasturini ishga tushiring. Bu dastur Windows tizim fayllarini tekshiradi va to'g'ri qo'yishda yordam beradi.
 - Linux operatsion tizimi uchun "fsck" dasturini ishga tushiring.
- 4. **Xotira Tekshiri:**
 - Operatsion tizim qo'shimcha dasturlari orqali xotiradagi muammolarini aniqlang va bartaraf eting. Windows uchun "Disk Cleanup" yoki "CCleaner" dasturlaridan foydalaning.
- 5. **Qurilma Ishtiyoq Tekshiri:**
 - Qurilma ishtiyoqini tekshirish uchun "Task Manager" yoki "Activity Monitor" dasturlarini oching. Bu dasturlar kompyuterning qanday ishlayotganligini va qaysi dasturlar yoki protsesslar ko'p ishtiyoqni sarflaydi, hisoblang.
- 6. **Internet Tezligi Tekshiri:**
 - Internet tezligini tekshirish uchun, internetni qo'llash maqsadida ma'lumotlarni ko'chirib olishga qodir dasturlardan foydalaning. Misol uchun, "Speedtest" sayti yoki dasturi.
- 7. **Qulayliklar:**
 - Kompyuterni tozalash, dastur va draiverlarni yangilash, va ishga tushirish ommaviy maslahatlaridir.

Bu testlar, kompyuterning asosiy taraflarini, dasturiy ta'minotni, xotirani, va faoliyatini tekshirishda yordam beradi. Agar hech qanday muammolar topilmasa, kompyuter savdo qilinayotgan bo'lsa yoki o'z vaqtida ishlamayotgan bo'lsa, mutlaqo mutaxassisga murojaat qiling.

S81.Zamonaviy vositalar asosida o'rnatilgan tizimlarni amaliy apparatdasturiy joriy etish.

Zamonaviy vositalar yordamida o'rnatilgan tizimlarni amaliy tatbiq etish uchun quyidagi asosiy bosqichlar ko'zda tutiladi:

1. Talablar tahlili va dizayn:

- Tizimning maqsad va funksiyalarini aniq belgilash.
- Zarur dasturiy ta'minotlar, apparat komponentlari, aloqa vositalari tanlash.
- Tizim ma'lumotlarini qayta ishlash, saqlash, uzatish usullarini loyihalash.

2. Apparatli qurilmalarni o'rnatish:

- Protsessor, xotira, yig'uvchi plata, sensor va aktlyatorlarni o'rnatish.
- Kuchlanish manbasi, quvvat taqsimlovchilar, quvvat yo'llari jihozlash.
- Tashqi o'lchovchi, boshqaruvchi va aloqa qurilmalarini ulash.

3. Dasturiy ta'minot yaratish:

- Mikrokontroller, FPGA yoki boshqa platforma uchun kod yozish.
- Sensorlardan ma'lumotlar olish, ularni qayta ishlash, boshqaruv komandalarini shakllantirish.
- Aloqa, ma'lumotlar saqlash, inson-mashina interfeysi kabi funksiyalarni amalga oshirish.

4. Tizimni sinash va sozlash:

- Apparat va dasturiy ta'minotning o'zaro bog'liqligini tekshirish.
- Barcha funksiyalar va parametrlarni sinab ko'rish.
- Tizimning samaradorligi, xavfsizligi, barqarorligini baholash.

5. Tizimni o'rnatish va ishlatishga tayyorlash:

- Apparat va dasturiy ta'minotni o'rnatish, konfiguratsiyalash.
- Foydalanuvchilarga o'rgatish, dokumentatsiya tayyorlash.
- Muntazam tekshiruv, yangilash va xizmat ko'rsatish rejasini tuzish.

Zamonaviy tizimlarni samarali amaliy joriy etish uchun apparat va dasturiy ta'minot komponentlarini muvofiqlashtirish, ularni qo'yilgan talablarga moslash, sinash va ishlatishga tayyorlash muhim.

Zamonaviy vositalar orqali o'rnatilgan tizimlarni amaliy apparatdasturiy joriy etish uchun quyidagi asosiy bosqichlar mavjud:

1. Kompyuter Arxivchisi:

- Kompyuter arxivchisining o'rnatilishi va sozlashi. Ushbu asbob, xotira birligini, protsessor va operatsion tizimni ta'minlaydi.
- O'rnatilgan tizimni arxivchiga ulashing va operatsion tizimni o'rnatish.

2. IoT qurilmasi:

- IoT qurilmasi, kompyuter arxivchisini va boshqa IoT vositalarini birlashtiradi.

- IoT qurilmasini operatsion tizimga bog'lang va u bilan birga ishlatiladigan dasturlarni o'rnatish.
- 3. **Mikrokontrollerlar:**
 - Mikrokontrollerlarni (masalan, Arduino, Raspberry Pi) va ularga bog'liq qurilmalarni (masalan, sensorlar, aktuatorlar) o'rnatish.
 - Dastur va ilovalarni yaratish yoki o'zlashtirish.
- 4. **Robotika:**
 - Robotik tizimlarni qurish, masalan, robot qurilmasini (kuzatish, robotlar, manipulyatorlar) tayyorlash.
 - Robotning boshqarilishi uchun dastur yozish.
- 5. **Avtoelektrik tizimlari:**
 - Avtoelektrik tizimlarni tayyorlash va o'rnatish (masalan, avtomobil diagnosika vositasi).
 - Ma'lumotlarni o'qish, o'zlashtirish va chiqarish uchun dasturlarni yaratish.
- 6. **O'qitish / O'rganish platformalari:**
 - O'qitish tizimlarini (masalan, Smart Board, Interaktiv o'qitish vositalari) va onlayn o'qitish platformalarini tayyorlash.
 - O'qitish materiallarini yaratish va onlayn o'qitish dasturlarini o'rnatish.
- 7. **Bog'liqlik tizimlari:**
 - Bog'liqlik tizimlari (masalan, qurilmaning ishlab chiqarish jarayonlarini boshqarish uchun ERP tizimlari) va loyihalarni tayyorlash.
 - Biznes jarayonlarini avtomatlashtirish va ma'lumotlarni birlashtirish dasturlarini o'rnatish.
- 8. **O'quv tizimlari:**
 - O'quv tizimlarini (masalan, ta'lim tizimlari, interaktiv testlar va taqdimotlar) o'rnatish.
 - O'quv materiallarini tayyorlash va o'quvchi ma'lumotlarini boshqarish dasturlarini yaratish.
- 9. **Tibbiy tizimlar:**
 - Tibbiy tizimlarni (masalan, elektron terapevtik va diagnostik vositalar) o'rnatish.
 - Xotirani, ma'lumotlarni, vaqt boshqarish dasturlarini yaratish.
- 10. **Veb-hizmatlar:**
 - Veb-saytlarni (masalan, onlayn do'konlar, portallar) va veb-hizmatlarni tayyorlash.
 - Ma'lumotlar bazalarini boshqarish va foydalanuvchilar bilan interaktiv muloqot dasturlarini yaratish.

Amaliy apparatdasturiy joriy etish bosqichlaridan birini tanlash uchun kompaniya yoki shaxsiy loyihangizning talablari va maqsadlari asosida boshqalarini konsultatsiya

qilish juda muhimdir. Bunday tizimlar tuzilishi, yaratilishi va sozlashi murakkab bo'lishi mumkin, shuning uchun maqsad va loyihangizni aniqroq tanlash kerak.

S82.BC qurilmalari, Uskuna-dasturiy ta'minot kompleksi

BC (Business Computers) qurilmalari - bu korporativ va sanoat tarmoqlarida keng qo'llaniluvchi kompyuter texnologiyalari turkumidir. Ular "uskuna-dasturiy ta'minot" yondashuvi asosida ishlab chiqilgan bo'lib, quyidagi asosiy tarkibiy komponentlardan iborat:

1. Apparat qurilmalari (Hardware):

- Protessorlar (CPU): Intel, AMD, ARM, POWER kabi arxitekturalardagi mikrokontrollerlar.
- Xotira qurilmalari: RAM, SSD, HDD, flash xotira, fayllar uchun massiv qurilmalari.
- Kiritish/chiqarish vositalari: sensorlar, aktyatorlar, displeylar, klaviatura, sichqoncha.
- Aloqa interfeyslari: Ethernet, Wi-Fi, Bluetooth, CAN, ModBus, RS-485 va boshqalar.
- Quvvat bloki, sovutish tizimi, kengaytirish platalari.

2. Dasturiy ta'minot (Software):

- Operatsion tizimlar: Windows Embedded, Linux, VxWorks, QNX.
- Dasturlash muhitlari: .NET, Java, C/C++, Python.
- Supervisory Control and Data Acquisition (SCADA) tizimlar.
- Aloqa protokollari: ModBus, OPC, PROFIBUS, CAN.
- Grafik interfeysi, ma'lumotlar bazasi, axborot xavfsizlik vositalari.

3. Integratsiya va boshqaruv:

- Apparat va dasturiy ta'minotning o'zaro aloqasi.
- Tizimlar, qurilmalar va dasturlar o'rtasidagi ma'lumotlar almashish.
- Markaziy boshqaruv, monitoring, diagnostika va xizmat ko'rsatish.

BC qurilmalari komponentlarining moslashuvchanligini, ularning tezkor yangilanishini, shuningdek, yuklamani qismlashtirish imkoniyatlarini ta'minlash uchun "uskuna-dasturiy ta'minot" yondashuvi qo'llaniladi. Bu yondashuvning asosiy afzalliklari - katta miqyosli, kasbiy darajadagi tizimlarni modernizatsiya qilish va rivojlantirish imkonini beradi.

BC qurilmalari (Boshlang'ich Kirish) kompyuter tizimlarini o'rganish va dasturiy ta'minotni o'rnatishda juda muhim vazifalarni o'z ichiga oladi. Bu uskuna-dasturiy ta'minot kompleksi, boshlang'ich kirish qurilmalari, masalan, klaviatura, maus, monitor, va o'zgaruvchilarga bog'liq komponentlardan iborat bo'lib, kompyuter tizimlarini tayyorlash va sozlash jarayonida yordam beradi. Quyidagi asosiy qismlar BC qurilmalari bo'lib, ularni tushuntirish mumkin:

1. Klaviatura (Keyboard):

- Klaviatura kompyuterga matn va belgilar kiritish uchun ishlatiladi.

- Harflar, raqamlar, belgilar va funktsiyalarni boshqarish uchun tugmalar mavjud.
 - Qo'llanish yuzi klaviaturaga qarab ishlaydi.
2. **Maus (Mouse):**
- Maus kompyuterda oyna shaklida ishlaydigan qurilmadir.
 - Ekranga ko'rsatiladigan kursorning o'girishini va boshqarishini ta'minlaydi.
 - O'yinlar, dasturlar va boshqa interaktiv jarayonlarda ishlatiladi.
3. **Monitor (Monitor):**
- Monitor kompyuterning ekrani hisoblanadi va tasvirlar, matnlar va boshqa grafik ma'lumotlarini chiqarish uchun ishlatiladi.
 - Ko'rsatilayotgan ma'lumotlarni foydalanuvchi tomonidan ko'rish va boshqarishga imkon beradi.
4. **O'zgaruvchilar (Peripherals):**
- O'zgaruvchilar, kompyuter bilan bog'liq qurilmalar bo'lib, ular klaviatura, maus, printerlar, skannerlar, flash yodlar, va boshqa qurilmalar bo'lishi mumkin.
 - Ular kompyuterga ma'lumot kiritish, qayta ishlash, yoki ma'lumotni chop etish uchun ishlatiladi.

BC qurilmalari, kompyuter tizimini boshqarish uchun qo'llaniladigan boshlang'ich qurilmalardan iborat bo'lib, uskuna-dasturiy ta'minot kompleksi tizimning ishini boshqarish va kompyuterda foydalanuvchi interfeysini ta'minlashda juda muhim rol o'ynaydi. Bu qurilmalar tuzilishi va sozlashi murakkab bo'lishiga qaramay, kompyuter tizimining ishlashi uchun zarur bo'lgan asosiy vositalardan biridir.

S83 Fotorezistorlar, ishlash tamoyillari, fotorezistorning tuzilishi bilan tanishish va uni loyihalarda qo'llash.

Fotorezistor (yoki fotoelement) - bu yorug'lik ta'sirida elektrik qarshiligi o'zgaruvchi devidir. Ular yorug'lik intensivligini elektr signalga aylantirish uchun qo'llaniladi.

Ishlash tamoyili:

Fotorezistorlar yorug'lik ta'sirida ularning elektr o'tkazuvchanligi ortishi yoki kamayishiga asoslangan. Bu hodisa yarim o'tkazgichlardagi valent va o'tish elektron-kovaklar juftliklarining paydo bo'lishi va rekombinatsiyasi hisobiga sodir bo'ladi. Yorug'lik yoki qorong'ilikda fotorezistorlarning qarshiligi ancha farq qiladi.

Tuzilishi:

Fotorezistor odatda ikki elektrodga ega bo'lib, ular orasidagi yarim o'tkazgich materialdan (kadmiy sulfid, kadmiy selenid, germaniy, kremniy va boshqalar) iborat sezuvchi element joylashtirilgan. Elektrodlar orasidagi masofani yetarli qilib tanlash orqali fotorezistorning yorug'lik ta'sirida qarshiligi o'zgarishi ta'minlanadi.

Loyihalarda qo'llash:

- Yorug'lik yoki havo oqimi o'zgarishlarini aniqlash uchun (yorug'lik integrallash, vaqtinchalik yoritilish sezgichlari, o'zgaruvchan yorug'lik detektorlari).
- Yorug'lik yordamida boshqariladigan avtomatlashtirilgan tizimlar (yog'dulanish sezgichlari, yoritilish boshqaruvi, qorong'ilik sezgichlari).
- Optik raqamli kodlashtirish va o'lchash qurilmalari.
- Optik aloqa va opto-elektron qurilmalarda tutqichlar, modulyatorlar.
- Fotointensivlikka sezgir kameralar, radiometrlar, spektrometrlar.

Fotorezistorlar oson foydalanish, sodda tuzilish va arzonligi bilan ajralib turadi. Ular avtomatika, elektron, optik va boshqa sohalarda keng qo'llaniladi.

Fotorezistor, chiroq yoki yorug'lik kuchini qabul qilish uchun ishlatiladigan fotosensorni ifodalovchi rezistor turidir. U, chiroqning kuchlanish darajasi o'zgarishi bilan o'zgaruvchili fotoelektrik qismlar orqali o'zgaradi. Fotorezistorlar, chiroqning intensivligi o'zgarishi bilan rezistentliklarida o'zgarishni hisobga oladilar.

Fotorezistorning tuzilishi:

- Fotorezistorlar, semiconductor materiallar, masalan, kadmiy selenid yoki silikon, bilan ishlab chiqiladi.
- Uning tuzilishi ikki o'q, chiroqning kuchlanish darajasi o'zgarishi bilan o'zgaruvchi materialdan iborat.
- Chiroqning kuchlanish darajasi oshib ketganda, fotosensorni ifodalovchi rezistorning rezistentlik miqdori kamayadi.

- Shu bilan birga, chiroqning kuchlanish darajasi kamayganda, fotosensorni ifodalovchi rezistorning rezistentlik miqdori oshadi.

Fotorezistorning ishlash tamoyillari:

1. **Chiroq intensivligini o'lchash:** Fotorezistorlar, chiroq intensivligini o'lchash uchun foydalaniladi. Masalan, otomatik qorong'ulikdan foydalanadigan avtomobillar, kundalik yoki kechqurunggi chiroq intensivligini o'lchash uchun fotorezistorlar ishlatiladi.
2. **Avtomatik chiroqlar:** Fotorezistorlar, avtomatik chiroqlarni yo'nash uchun ishlatiladi. Masalan, tungi paytlarda shaharni yoritish uchun ko'p joyda fotorezistorlar qo'llaniladi. Kunduzgi va kechqurunggi vaqtlarda, chiroqlar avtomatik ravishda yo'nalishadi.
3. **Optik sensorlar:** Fotorezistorlar, optik sensorlar sifatida ham ishlatiladi. Masalan, printerlarda, skannerlarda va boshqa optik qurilmalarda rasm yoki matnning ko'rish jarayonini boshlash uchun fotorezistorlar qo'llaniladi.
4. **Chiroq terishini boshqarish:** Fotorezistorlar, chiroq terishini boshqarish uchun ishlatiladi. Masalan, chiroq terishini avtomatik ravishda boshqarish uchun uni qurilma yoki asbob uskunalarda ishlatish mumkin.

Loyihalarda fotorezistorlarni qo'llash:

- **Avtochiroqlar:** Avtochiroqlarda, kunduzgi va kechqurunggi vaqtlarda chiroqlarning yo'nalishi uchun fotorezistorlar ishlatiladi. Masalan, kunduzgi paytda yorug'likni yengilaydigan va kechqurunggi vaqtlarda chiroqlarni yo'qotadigan avtomobil farolga yorug'lik sensorlari.
- **Avtomatik qorong'uliklar:** Avtomatik qorong'ulik qurilmasida, tungi va kechqurunggi vaqtlarda qorong'ulik intensivligini o'lchash uchun fotorezistorlar qo'llaniladi. Chiroqlarning intensivligi past bo'lganda, qorong'uliklar avtomatik ravishda yo'qotiladi.
- **Optik sensorlar:** Fotorezistorlar, optik sensorlar sifatida ham ishlatiladi. Masalan, printerlarda, skannerlarda, va boshqa optik qurilmalarda fotosensorni ifodalovchi komponentlar qo'llaniladi.

Fotorezistorlar, juda keng qo'llanma maydoniga ega va turli sohalar uchun qulaylik yaratadi. Ular, chiroqlarning kuchlanish darajasini va intensivligini o'lchashda, avtomatik qorong'ulik qurilmasida va optik sensorlar sifatida foydalaniladi.

S84 Rele va transistor ishlash imkoniyatlari.

Rele - elektromagnitli kommutatsiya qurilmasidir. U elektr zanjirida o'zi qo'llanilayotgan tokdan boshqa kattalikdagi tokni kommutatsiya qila oladi.

Rele ishlash imkoniyatlari:

- Bir zanjirdan ikkinchi zanjirni kommutatsiya qilish (kuchlanish, tok, yuk faktorini uzatish).
- Signallarni kuchaytirish (elektromagnit relening o'zi kuchlanish yoki tok signalini kuchaytira oladi).
- Zanjirlarni bir-biridan galvanik ajratish.
- Chekli kattalikdagi boshqaruv signali yordamida kattalikdagi yuk zanjirini ishlata olish.
- Avtomatik boshqaruv, himoya, monitoring tizimlarida qo'llash.

Tranzistor

Tranzistor - yarim o'tkazgichli aktivsiz qurilma bo'lib, elektronlar oqimini boshqarish uchun ishlatiladi.

Tranzistor ishlash imkoniyatlari:

- Kuchlanish, tok va quvvatni kuchaytirish.
- Elektr zanjirlarni kommutatsiya qilish (yoqish/o'chirish).
- Chekli signallar yordamida kattalikdagi yuk zanjirini boshqarish.
- Digital va analogik signallar ishlov berish.
- Avtomatik boshqaruv tizimlarida keng qo'llaniladi (amplifikatorlar, kommutatorlar, generatorlar, chastota o'zgartiruvchilar).
- Mikroelektronika, kompyuter texnologiyalari, radioe'lonlar, sensorlar, robotlar tizimlarida qo'llaniladi.

Umumiy qilib aytganda, rele va tranzistorlar boshqaruv, kuchlanishni kuchaytirish, kommutatsiya va ajratish kabi vazifalarni amalga oshira oladilar. Lekin tranzistorlar relelarga nisbatan ancha kichik, tezkor va kamenergiyadir.

Rele va tranzistorlar, elektr energiyasini boshqarishda juda keng qo'llaniladigan elektronik qurilmalar. Ularning har birining o'zining afzalliklari va ishlatilish sohalari mavjud. Quyidagi qismlarda, har birining ishlash imkoniyatlari va foydalanish sohalari haqida batafsil ma'lumot beraman:

Rele:

1. **Ishlaydigan prinsip:** Rele, kichik miqdordagi elektr energiyasini o'rtacha yoki ko'p miqdorda elektr energiyasiga aylantirishda ishlatiladi. Bu, boshqaruv tizimlarida amalga oshirish uchun juda mashhur qurilma.
2. **Imkoniyatlari:**

- **Kuchli elektr to'g'ridan to'g'ri (SPST) rele:** Bir elektr chiqishini yo'q qilish va uchun to'g'ridan to'g'ri qo'laydigan osishchisiz kontakti yo'q qilish uchun ishlatiladi.
- **Kuchli elektr kattaligi (DPST) rele:** Ikki elektr chiqishini bir vaqtda yo'q qilish uchun ishlatiladi.
- **Kuchli elektr tarmoq (SPDT) rele:** Bitta elektr chiqishni ikki ta odam orasida tanlov qilish uchun ishlatiladi.

3. Foydalanish sohalari:

- Avtomobil elektronika-larida oyna yoki mehnat qilish vositalari uchun
- Elektr qurilma vositalarini boshqarish uchun boshqaruv paneliga bog'langan dasturlarda
- Quvvatli qurilma vositalarini yo'q qilish uchun kuchli elektr tizimlari

Tranzistor:

1. **Ishlaydigan prinsip:** Tranzistor, elektr to'g'ridan-to'g'ri (DC) ta'minoti orqali boshqariladigan, o'zgaruvchan miqdordagi elektr ta'minotlarni boshqarishda ishlatiladi. Bu, amperliyatni kuchaytirish va o'zgaruvchiligi oshirish imkoniyatiga ega.

2. Imkoniyatlari:

- **NPN tranzistorlar:** Elektr ta'minoti emitor, kollektor va baza orqali o'tadi. Elektr to'g'ridan-to'g'ri ta'minoti baza bilan birgalikda o'rnatilgan bo'lganda, emitor va kollektor orasidagi oqim ochiq.
- **PNP tranzistorlar:** Elektr ta'minoti emitor, kollektor va baza orqali o'tadi. Elektr to'g'ridan-to'g'ri ta'minoti baza bilan birgalikda o'rnatilgan bo'lganda, emitor va kollektor orasidagi oqim yopiq.

3. Foydalanish sohalari:

- Kuchli elektriy ta'minotlarni boshqarish uchun
- Ovqat korxonalari, ushbu damlarni boshqarish
- Avtomobil elektrik tizimlari, avtomobil lampalarni yonish
- Amplifikatsiya va elektron dasturlash

Tranzistorlar va relenin har birining o'zining afzalliklari va foydalanish sohalari mavjud. Rele kuchli elektr ta'minotlarini boshqarishda va kontrolli qurilma vositalarini yo'q qilishda, tranzistor esa o'zgaruvchan elektr ta'minotlarini boshqarish va amplifikatsiya uchun yaxshi variant bo'ladi. To'g'ri tanlov, foydalanish maqsadi va tizimning zaruratiga bog'liq.

S85 Arduino tizimini haqida ma'lumot. Komponentlar va ularning imkoniyatlari.

Arduino - ochiq kod va ochiq qurilma asosida ishlovchi mikrokontroller platforma. U oson dasturlashtiriladi va ko'plab sensorlar, motorlar va boshqa komponentlar bilan ishlash imkoniyatiga ega.

Arduino tarkibiy qismlari:

1. **Mikroprotessor:** Odatda Atmel AVR oilasidan bo'lgan 8-bit yoki 32-bit mikrokontrollerlar (ATmega328P, ATmega2560 va boshqalar).
2. **Digital I/O:** Raqamli kirish/chiqish uchun portlar.
3. **Analog I/O:** Analog signallarni o'lchash uchun portlar.
4. **UART, I2C, SPI:** Seriya portlar, interfeyslar.
5. **PWM:** Pulsli kenglik modulyatsiyasi uchun portlar.
6. **USB:** Platforma va komputerning o'zaro aloqa qilishi uchun USB port.
7. **Quvvat tizimi:** Quvvatlash uchun portlar.
8. **Dasturlash chipi:** Mikroprotessorni dasturlab olish uchun.

Arduino platforma imkoniyatlari:

- Oson dasturlash va bir qator tillarni qo'llab-quvvatlash (C, C++).
- Ko'plab sensorlar, motorlar, ekranlar va boshqa komponentlar bilan o'zaro aloqa.
- Internetga ulash, WiFi, Bluetooth, Ethernet kabi kengaytmalar.
- 3D printer, CNC, robotlar, IoT va boshqa loyihalar uchun platforma.
- Ochik kod va keng hamjamiyat tomonidan qo'llab-quvvatlanadi.

Arduino komponentlar va ularning imkoniyatlari:

- Sensorlar: Yorug'lik, harorat, namlik, harakat, kuch, tezlik va boshqa sezgichlar.
- Aktlyatorlar: Motorlar, servolar, releler, displeylar, svetodiodlar.
- Kommunikatsiya: WiFi, Bluetooth, Ethernet, RFID modul va boshqalar.
- Qo'shimcha plata va kengaytmalar.

Arduino - oson, arzon va keng tarqalgan mikrokontroller platforma bo'lib, turli elektron loyihalar uchun keng qo'llaniladi.

Arduino Tizimi haqida Ma'lumot

Arduino - bu ochiq kodli apparat va dasturiy platforma bo'lib, u sensorlar va aktuatorlarni boshqarish uchun ishlatiladi. Arduino platformasi, mikrokontroller platalari, dasturlash muhitlari va kutubxonalaridan iborat. Bu platforma elektronikani o'rganish, tezkor prototiplash va turli xil loyihalarni amalga oshirish uchun juda qulay.

Arduino Tizimining Komponentlari

1. **Arduino Board (Boshqaruv Paneli):**

- **Arduino UNO:** Eng mashhur va keng tarqalgan model. Atmega328P mikrokontrolleridan foydalanadi, 14 raqamli va 6 analog pinlarga ega.
 - **Arduino Mega:** Katta loyihalar uchun mo'ljallangan. Atmega2560 mikrokontrolleridan foydalanadi, 54 raqamli va 16 analog pinlarga ega.
 - **Arduino Nano:** Kichik va kompakt loyihalar uchun. Atmega328P mikrokontrolleridan foydalanadi, 14 raqamli va 8 analog pinlarga ega.
 - **Arduino Leonardo:** USB interfeysi orqali kompyuter bilan oson integratsiya qilinadi. Atmega32u4 mikrokontrolleridan foydalanadi.
2. **Mikrokontroller (Microcontroller):**
- Arduino platalarida turli mikrokontrollerlar ishlatiladi (masalan, Atmega328, Atmega2560).
 - Mikrokontrollerlar, dasturlarni yuklab olish va bajarish uchun ishlatiladi, analog va raqamli signallarni qayta ishlaydi.
3. **Quvvat Manbai (Power Supply):**
- Arduino platalari 5V yoki 3.3V quvvat manbai bilan ishlaydi.
 - Quvvat manbai sifatida USB port yoki tashqi quvvat adapteridan foydalanish mumkin.
4. **USB Interfeysi:**
- Arduino platalari kompyuter bilan bog'lanish uchun USB interfeysidan foydalanadi.
 - USB orqali dasturlarni yuklash va quvvat bilan ta'minlash mumkin.
5. **Analog va Raqamli Pinlar:**
- **Raqamli Pinlar (Digital Pins):** Signalni ON/OFF holatida qabul qiladi yoki uzatadi. Misol: LED yoritish, tugmalar.
 - **Analog Pinlar (Analog Pins):** Analog signallarni qabul qiladi va raqamli signalga aylantiradi. Misol: Harorat sensori, potensiometr.
6. **PWM (Pulse Width Modulation) Pinlar:**
- Raqamli signallarni PWM texnologiyasi orqali analog signalga o'xshash qilib o'zgartiradi.
 - LED yorqinligini boshqarish, motorlarni tezligini boshqarish uchun ishlatiladi.
7. **Seriya interfeysi (Serial Communication):**
- Arduino va boshqa qurilmalar yoki kompyuter o'rtasida ma'lumot almashish uchun seriya interfeysi ishlatiladi (UART, I2C, SPI).

Arduino Tizimining Imkoniyatlari

1. Prototiplash:

- Arduino platformasi turli xil loyihalarni tezkor prototiplash uchun juda qulay.
- Sensorlar, motorlar, displeylar va boshqa komponentlarni tez va oson ulash mumkin.

2. Dasturlash Interfeysi:

- Arduino IDE (Integrated Development Environment) yordamida dasturlarni yozish va yuklash oson.
- C/C++ asosida yozilgan oddiy va intuitiv kodlar.

3. Ochiq Kodli Kutubxonalar:

- Arduino uchun ko'plab ochiq kodli kutubxonalar mavjud.
- Har xil komponentlar va sensorlar bilan ishlashni osonlashtiradi.

4. Keng Jamoa va Resurslar:

- Arduino platformasi katta foydalanuvchilar jamoasiga ega.
- Forumlar, darsliklar, loyihalar va boshqa resurslar orqali yordam olish mumkin.

5. Tarmoq va IoT (Internet of Things):

- Arduino WiFi yoki Ethernet modullari yordamida internetga ulanishi mumkin.
- IoT loyihalarini amalga oshirishda foydalaniladi.

6. Keng Qo'llanilish Soqalari:

- Ta'lim, robototexnika, avtomatlashtirish, san'at va texnologiya, uy avtomatizatsiyasi va ko'plab boshqa sohalarda qo'llaniladi.

Arduino, oson dasturlash va turli xil loyihalarni amalga oshirish imkoniyatlari bilan elektronika va dasturlashni o'rganish uchun juda qulay platforma. Keng kutubxonalar va katta jamoa yordami bilan, Arduino har qanday loyiha uchun mukammal tanlovdir.

S86.Yorug'lik diod ishlash prinsipi va turlari. Yorug'lik diodlarning ishlash prinsipi va turlari

Yorug'lik diodning ishlash prinsipi

Yorug'lik diodi (LED - Light Emitting Diode) - yarim o'tkazgichli elektron qurilma bo'lib, unga ta'sir etgan elektr tok natijasida nur chiqaradi. LED ishlashining asosiy prinsipi quyidagicha:

1. Yarimotkaz materiallarda elektron-yuk teshigini tutish.
2. Elektronlar va yuk teshiklari qayta birlashganda, energiya ajralib chiqadi.
3. Bu ajralib chiqqan energiya to'lqin uzunligiga mos bo'lgan nur shaklida yutiladi.
4. Nur qanchalik yuqori energetik bo'lsa, u shunchalik ko'zga ko'rinuvchi qismga (ko'k, yashil, qizil) mos keladi.

Yorug'lik diodlarning turlari

1. **Rangli LED-lar:** Qizil, yashil, ko'k, sariq, oq va boshqa ranglar mavjud.
2. **Infraqizil LED-lar:** Ko'zga ko'rinmas infraqizil nurlarga ega.
3. **Ultrabinafsha LED-lar:** Ko'zga ko'rinmas ultrabinafsha nurlarga ega.
4. **OLED (Organic LED):** Organik moddalardan tayyorlangan chiroyli, nozik va energiyani kam sarflaydigan LED-lar.
5. **Matrisali (piksel) LED-lar:** Ko'p LED-lardan tashkil topgan matrisa ko'rinishida.
6. **Maxsus LED-lar:** Ko'p rangli, tezkor, kuchli nur beruvchi, lazerli LED-lar va boshqalar.

LED-larning qo'llanilish sohalari

LED-lar keng qo'llaniladi:

- Yoritish: Lampa, chiroqlar, displeylar
- Ko'rsatkichlar: Signal, belgili chiroqlar
- Dekorativ yoritish: Bezaklar, yorug'lik effektlari
- Sensor: Optik, infra-qizil, ultrabinafsha sensorlar
- Optik aloqa: LED-lar asosidagi kommunikatsiya tizimlar

LED-lar nozik, kichik o'lchamli, arzon, uzun xizmat muddatli, energiyani kam sarflaydigan va turli ranglar bilan mavjud bo'lgani uchun ko'plab sohalarda uchun mos keladigan yorug'lik manbai hisoblanadi.

Yorug'lik Diodi (LED) Ishlash Prinsipi

Yorug'lik diodi (Light Emitting Diode, LED) - bu elektron komponent bo'lib, elektr energiyasini yorug'likka aylantiradi. Yorug'lik diodining asosiy ishlash prinsipi p-n o'tish qatlami orqali elektronlar va teshiklarning rekombinatsiyasi natijasida yorug'lik (fotonlar) chiqarishdan iborat.

Ishlash prinsipi:

1. **P-N o'tish qatlam:** LED ikkita asosiy qatlamdan iborat: p-type (pozitiv) va n-type (negativ) yarim o'tkazgichlar. P-type yarim o'tkazgich ko'p miqdorda teshiklarga (pozitiv zaryad tashuvchilar) ega, n-type esa ko'p miqdorda elektronlarga (negativ zaryad tashuvchilar) ega.
2. **Elektr toki ta'siri:** P-N o'tish qatlamiga elektr toki o'tganda, elektronlar n-type qismidan p-type qismiga o'tadi. Elektronlar va teshiklar rekombinatsiyalanadi (qo'shiladi) va bu jarayon natijasida energiya fotonlar shaklida chiqariladi.
3. **Fotonlar chiqarilishi:** Rekombinatsiya jarayonida chiqarilgan energiya fotonlar shaklida bo'lib, bu fotonlar yorug'lik ko'rinishida chiqadi.

Yorug'lik Diodi Turlari

LEDlar turli xil turlarga ega bo'lib, ularning har biri ma'lum bir ilova uchun mo'ljallangan. Quyida eng keng tarqalgan LED turlari keltirilgan:

1. **Standart LED:**
 - Odatda qizil, yashil, sariq yoki ko'k rangda bo'ladi.
 - Past kuchlanish va oqimda ishlaydi.
 - Asosan indikatorlar, displeylar va oddiy yoritish tizimlarida ishlatiladi.
2. **Yuqori Yorqinlikdagi LED (High-Brightness LED):**
 - Oddiy LEDlarga nisbatan yorqinroq.
 - Odatda ko'proq quvvat sarflaydi.
 - Chiroqlar, avtomobil yoritish tizimlari va boshqa yuqori yorqinlik talab qiladigan ilovalarda ishlatiladi.
3. **RGB LED:**
 - Qizil, yashil va ko'k LEDlarni bir korpusda birlashtiradi.
 - Har xil ranglarni aralashtirish va turli ranglar hosil qilish imkonini beradi.
 - Asosan dekorativ yoritish va rangli displeylar uchun ishlatiladi.
4. **Infraqizil LED (IR LED):**
 - Ko'zga ko'rinmas infraqizil yorug'lik chiqaradi.
 - Asosan masofaviy boshqaruv pultlari va tunda ishlovchi kameralar uchun ishlatiladi.
5. **Ultrabinafsha LED (UV LED):**
 - Ultrabinafsha yorug'lik chiqaradi.
 - Tibbiy davolash, sterilizatsiya va maxsus qoplamalarni faollashtirish uchun ishlatiladi.
6. **OLED (Organic LED):**
 - Organik materiallardan yasalgan LED.
 - Ekranlarda yuqori sifatli tasvir va past energiya sarfi bilan mashhur.
 - Mobil telefonlar, televizorlar va boshqa ko'rinishdagi displeylar uchun ishlatiladi.

7. COB LED (Chip on Board):

- Bir nechta LED chiplarini bir platformaga joylashtiradi.
- Yagona LEDga qaraganda ko'proq yorqinlik va energiya samaradorligini ta'minlaydi.
- Asosan yoritish ilovalarida ishlatiladi.

Yorug'lik diodlari (LEDlar) yuqori samaradorlik, uzoq muddatli xizmat va kichik hajmi tufayli ko'plab ilovalarda keng qo'llaniladi. Ularning turli xil turlari va ranglari har xil maqsadlarda ishlatilishi mumkin, bu esa ularni elektronika va yoritish texnologiyalarida muhim komponentlarga aylantiradi.

S87.Fotorezistor nima. Fotorezistor ta'sirida yorug'ligini qiymatini aniqlash.

Fotorezistor - yorug'lik shuasiga sezgir bo'lgan yarimo'tkazgichli rezistor. Unda yorug'lik kuchiga bog'liq ravishda elektr qarshiligi o'zgaradi. Boshqacha qilib aytganda, fotorezistor yorug'lik ta'sirida qarshiligini o'zgartiradigan elektron qurilmasi.

Fotorezistorning ishlash prinsipi

Fotorezistordagi yarimo'tkazgich materiallarda (odatda, kadmiy sulfid yoki kadmiy selenid) yorug'lik ta'sirida erkin elektronlar va bo'shliqlar hosil bo'ladi. Bu erkin zaryad tashuvchilar o'tkazuvchanlikni oshiradi va qarshilikni kamayishiga olib keladi. Yanada qorong'i sharoitlarda esa, qarshilik oshadi.

Fotorezistorning qiymatlari va yorug'lik

Fotorezistorning elektr qarshiligi (R) quyidagicha o'zgaradi:

- Qorong'i sharoitda (past yorug'lik): R yuqori bo'ladi (masalan, $1\text{ M}\Omega$)
- Yorug'lik ta'sirida: R kamayadi (masalan, $1\text{ k}\Omega$)

Fotorezistorni yorug'lik shiddat (E) bilan qiymati orasidagi bog'lanish quyidagicha:

$$R = k / E^n$$

Bunda k - doimiy, n - fotorezistorning turdagi qiymati (0.5 - 1 oralig'ida).

Demak, fotorezistorning qarshiligi yorug'lik kuchiga teskari proporsional bo'ladi.

Yorug'lik kuchini o'lchash uchun fotorezistorlardan keng foydalaniladi.

Fotorezistor (yoki yorug'likka sezgir rezistor, LDR - Light Dependent Resistor) - bu elektron komponent bo'lib, yorug'lik ta'sirida uning elektr qarshiligi o'zgaradi. Yorug'likning intensivligi oshishi bilan fotorezistorning qarshiligi kamayadi, va aksincha, yorug'likning intensivligi kamayganda uning qarshiligi oshadi.

Fotorezistorning Ishlash Prinsipi

Fotorezistor yarim o'tkazgich materiallardan yasaladi. Yorug'lik fotonlari yarim o'tkazgich materialga tushganda, ular energiya qo'shib, elektronlar va teshiklarni yarim o'tkazgich ichida harakatga keltiradi. Bu jarayon elektr o'tkazuvchanligini oshiradi va fotorezistorning qarshiligi kamayadi. Yorug'lik yo'qligida esa, elektr o'tkazuvchanlik past bo'ladi va qarshilik oshadi.

Fotorezistor Ta'sirida Yorug'likni Qiymatini Aniqlash

Fotorezistor yordamida yorug'lik intensivligini o'lchash uchun uning qarshiligini o'lchash kerak. Bu jarayon odatda mikrokontroller yoki boshqa elektron qurilmalar yordamida amalga oshiriladi. Quyidagi amallar yorug'likni qiymatini aniqlash uchun qo'llaniladi:

1. Fotorezistorni Ulash:

- Fotorezistorni kuchlanish bo'luvchi sxemada foydalanish odatiy hisoblanadi. Bu sxema yordamida fotorezistor qarshiligini o'lchash oson bo'ladi.
- 2. **Analog Signalni O'lchash:**
 - Fotorezistorning qarshiligi o'zgarganda, kuchlanish bo'luvchi sxemaning chiqish kuchlanishi ham o'zgaradi.
 - Bu chiqish kuchlanishi mikrokontrollerning analog kirishiga ulanadi va ADC (Analog to Digital Converter) orqali raqamli qiymatga aylantiriladi.
- 3. **ADC Qiymatini O'qish:**
 - Mikrokontroller ADC yordamida fotorezistorning chiqish kuchlanishini o'qiydi va raqamli qiymatga aylantiradi.
 - Bu raqamli qiymat yorug'lik intensivligiga mos keladi.
- 4. **Yorug'likni Qiymatga Aylantirish:**
 - ADC qiymatini yorug'lik intensivligiga aylantirish uchun kalibrlash kerak bo'ladi. Kalibrlash jarayoni yorug'lik intensivligi va ADC qiymatlari o'rtasida bog'lanishni topishga yordam beradi.

Fotorezistor Qo'llanilish Soqalari

1. **Yoritish Nazorati:**
 - Uydagi avtomatik yoritish tizimlari, ko'cha chiroqlari va boshqa yoritish tizimlari.
2. **Kameralarda Ekspozitsiya Nazorati:**
 - Kameralar va boshqa optik qurilmalarda yorug'lik miqdorini o'lchash va mos ravishda sozlash.
3. **Tugma va Sensorlar:**
 - Yorug'lik sezgir sensorlar va tugmalar.
4. **Xavfsizlik Tizimlari:**
 - Harakatni aniqlash va xavfsizlik tizimlarida.

Fotorezistorlar oddiy, arzon va foydali komponentlar bo'lib, ular ko'plab ilovalarda yorug'lik intensivligini o'lchash va nazorat qilish uchun ishlatiladi.

S88. LCD Ekran, Rangli 128x160LCD ekran bilan ishlash.

LCD (Liquid Crystal Display) - suyuq kristall ekran. LCD ekran elektrik signal yordamida yorug'likning ma'lum qismini o'tkazishi yoki to'sishi orqali ma'lumotlarni ko'rsatadi. LCD-lar kam energiya sarflaydi, kichik o'lchamli va arzon bo'lganligi uchun keng tarqalgan.

128x160 rangli LCD ekranning xususiyatlari

- **O'lchami:** 128 piksel eni, 160 piksel bo'yi (2.4 dyuymli)
- **Rang:** 65,536 rangli (16-bitli)
- **Aloqa:** Serial interfeys (I2C, SPI)
- **Boshqarish:** Korpusda yoki tashqi mikroprotsessor yordamida
- **Qo'llanish:** Mobil qurilmalar, elektronika, interaktiv panellar

128x160 rangli LCD ekran bilan ishlash

1. **Ekranni ulash:** LCD ekranni mikroprotsessor yoki Arduino, Raspberry Pi kabi platalarga mos interfeys orqali ulash (I2C, SPI).
2. **Konfiguratsiya:** LCD drayver/kutubxona yordamida ekran parametrlarini (rang, yorug'lik, kontrastlik) sozlash.
3. **Matn/grafika chiqarish:** LCD drayver/kutubxona funksiyalari yordamida ekranga matn, piksel, chiziq, shakl, rasm chiqarish.
4. **Animatsiya:** Kichik animatsiyalar, ko'rsatkichlar, indikatorlar yaratish.
5. **Boshqaruv:** Ekran ustida sensorli yoki tugmali boshqaruv imkoniyatlari.

Bu rang LCD ekranlarda rangli tasvirlar, grafik interfeyslar, odam-mashina muloqotini ta'minlovchi interfeys yaratish mumkin. Ular kompakt qurilmalar uchun juda mos keladi.

LCD Ekran

LCD (Liquid Crystal Display) ekranlari suyuq kristallar yordamida tasvirlarni yaratadigan yassi panelli displeylar hisoblanadi. LCD ekranlar ko'p turlarda mavjud bo'lib, oddiy ikki rangli (monoxrom) displeylardan tortib, rangli (RGB) displeylargacha keng ko'lamda ishlatiladi.

128x160 Rangli LCD Ekran

128x160 o'lchamli rangli LCD ekranlar kichik va o'rta loyihalar uchun juda mashhur. Bu ekranlarda 128 piksel kenglik va 160 piksel balandlikda tasvir ko'rsatiladi. Rangli LCD ekranlar odatda RGB (Red, Green, Blue) modelida ishlaydi, ya'ni har bir piksel uchta rangning kombinatsiyasi orqali rang hosil qiladi.

128x160 Rangli LCD Ekran bilan ishlash

128x160 rangli LCD ekran bilan ishlash uchun odatda mikrokontroller (masalan, Arduino) va tegishli kutubxonalar qo'llaniladi. Quyidagi qadamlar ekran bilan ishlash jarayonini ko'rsatadi:

1. Ekranni ulash

Ekranni mikrokontrollerga to'g'ri ulash kerak. Ekran interfeysiga qarab (SPI, I2C yoki Parallel) ulanish sxemasi o'zgarishi mumkin. Ko'p hollarda SPI interfeysi ishlatiladi. Ulash uchun ekran va mikrokontroller o'rtasidagi mos pinlarni ulash kerak:

- VCC: Quvvat manbai (+3.3V yoki +5V)
- GND: Zamin
- SCL (Serial Clock): Soat signali (mikrokontrollerning SCK piniga)
- SDA (Serial Data): Ma'lumot signali (mikrokontrollerning MOSI piniga)
- RES (Reset): Reset pini
- DC (Data/Command): Ma'lumot yoki buyruq tanlash pini
- CS (Chip Select): Chipni tanlash pini

2. Arduino IDE va Kutubxonalar

Arduino platformasida ishlash uchun tegishli kutubxonalarni o'rnatish kerak. Eng mashhur kutubxonalardan biri bu Adafruit tomonidan taqdim etilgan Adafruit ST7735 va Adafruit GFX kutubxonalaridir.

3. Kutubxonalarni o'rnatish

Arduino IDE oching va quyidagi kutubxonalarni o'rnatish:

1. Adafruit GFX Library:

- Arduino IDE menyusida "Sketch" -> "Include Library" -> "Manage Libraries..." ni tanlang.
- Qidiruv maydoniga "Adafruit GFX" deb yozing va Adafruit GFX Library kutubxonasini o'rnatish.

2. Adafruit ST7735 and ST7789 Library:

- Qidiruv maydoniga "Adafruit ST7735" deb yozing va Adafruit ST7735 and ST7789 Library kutubxonasini o'rnatish

3. 128x160 rangli LCD ekranlar mikrokontroller loyihalarida foydalanish uchun juda qulay. Ushbu ekranlar ko'plab sensorli va vizual interfeys ilovalarida ishlatiladi. Ekranni to'g'ri ulash, tegishli kutubxonalarni o'rnatish va dasturlash orqali siz ushbu ekranlarni o'z loyihalaringizda muvaffaqiyatli ishlatishingiz mumkin.

4. 4o

S89. Harf va raqamlarni displeyga chiqarish. Ekran orqali Geometrik shakllarni harakatlantirish.

Harf va raqamlarni displeyga chiqarish uchun LCD, OLED yoki LED displeylar keng qo'llaniladi. Harf va raqamlar matnlarini boshqarish uchun mikroprotsessor yoki mikrokontrollerdan foydalaniladi. Bu qurilmalar boshqarish orqali matn va raqamlarni displeyga yuborish, joylashtirish va ko'rsatish imkonini beradi.

Geometrik shakllarni harakatlantirish uchun esa grafik interfeys va animatsiya imkoniyatlari kerak bo'ladi. Bu maqsadda ko'p piksellik LCD, OLED yoki LED displeylar yoki grafik displeylar, masalan, TFT (Thin Film Transistor) displeylar ishlatiladi. Mikroprotsessor yoki mikrokontroller orqali boshqarish yordamida geometrik shakllar, chiziqchalar, animatsiyalar yaratish va harakatlantirish mumkin. Bunday boshqaruv tizimlari va dasturlash tillari, masalan, Arduino, Raspberry Pi, Python, C++, C# kabi dasturlash tillari va kutubxonalar yordamida ishlatiladi. Ularning imkoniyatlari orqali matn, raqam, geometrik shakllar va harakatlantirishlar ko'rsatilishi, o'zgartirilishi va boshqarilishi mumkin. Misol uchun, grafik displeyda ko'rsatilgan shaklni to'g'ri, yuqori, pastga, chapga, o'ngga harakatlanishi kiritilishi mumkin.

Shuningdek, sensorli displeylar (masalan, naqshi ortida sensorlar bo'lgan TFT ekranlar) orqali foydalanuvchining kirishlarini (to'qima, chizish, chertish) qabul qilib, shu asosda geometrik shakllar ustida interaktiv harakatlar amalga oshirish ham mumkin.

Harf va Raqamlarni Displeyga Chiqarish

128x160 rangli LCD ekranga harf va raqamlarni chiqarish uchun mikrokontroller (masalan, Arduino) va Adafruit ST7735 hamda Adafruit GFX kutubxonalaridan foydalanish mumkin. Quyidagi kod misolida harf va raqamlarni displeyga chiqarish ko'rsatiladi:

Ekran Orqali Geometrik Shakllarni Harakatlantirish

Geometrik shakllarni harakatlantirish uchun Adafruit GFX kutubxonasidagi grafik funksiyalaridan foydalanish mumkin. Quyidagi kod misolida to'rtburchak shaklini ekranda harakatlantirish ko'rsatiladi:

Geometrik Shakllarni Chizish uchun GFX Kutubxonasi Funksiyalari

Adafruit GFX kutubxonasi yordamida ekranda turli geometrik shakllarni chizish mumkin. Quyida ba'zi asosiy funksiyalar ko'rsatilgan:

- `tft.drawPixel(x, y, color)`: Bitta piksel chizish.

- `tft.drawLine(x0, y0, x1, y1, color)`: Chiziq chizish.
- `tft.drawRect(x, y, w, h, color)`: To'rtburchak chizish.
- `tft.fillRect(x, y, w, h, color)`: To'rtburchakni to'ldirish.
- `tft.drawCircle(x, y, r, color)`: Doira chizish.
- `tft.fillCircle(x, y, r, color)`: Doirani to'ldirish.
- `tft.drawTriangle(x0, y0, x1, y1, x2, y2, color)`: Uchburchak chizish.
- `tft.fillTriangle(x0, y0, x1, y1, x2, y2, color)`: Uchburchakni to'ldirish.

128x160 rangli LCD ekran bilan harf va raqamlarni chiqarish hamda geometrik shakllarni harakatlantirish uchun Adafruit GFX va ST7735 kutubxonalaridan foydalanish mumkin. Ushbu kutubxonalar turli grafik funksiyalarni taqdim etadi, bu esa ekranda turli xil shakllar, matn va animatsiyalarni osongina yaratishga imkon beradi.

S90.RFID reader RC522 Moduli haqida tushuncha va ishlash jarayoni?

RFID (Radio Frequency Identification) reader RC522 moduli, bezaklar orqali ma'lumotlarni o'qish va yozish uchun ishlatiladigan bir tur avtomatik identifikatsiya tizimi modulidir. U mikroprotsessorlar, Arduino va boshqa kontrollerlar bilan ishlab chiqilgan tizimlar uchun mo'ljallangan.

RC522 modulining ishlash jarayoni quyidagicha bo'ladi:

1. **Komunikatsiya interfeysi:** RC522 modulini mikrokontroller bilan bog'lash uchun komunikatsiya interfeysi kerak bo'ladi. Modulning eng ko'p ishlatiladigan interfeysi SPI (Serial Peripheral Interface) bo'ladi. Bu interfeys orqali datalar yuboriladi va qabul qilinadi.
2. **Antenna ulash:** Modulda anten tizimi mavjud bo'ladi. Bu anten arqali bezaklardagi ma'lumotlar bilan radiofrekans (RF) kommunikatsiya o'rnatiladi. Bezaklar modulning yonida o'tkazilganda, RF signal orqali ma'lumotlar o'qiladi.
3. **SPI kommunikatsiyasi:** Mikrokontrollerdan modulga SPI interfeysi orqali komandalar va ma'lumotlar yuboriladi. Modul SPI interfeysining mo'ljallangan pinlariga ulanadi (masalan, SDA, SCK, MOSI, MISO).
4. **Protokol va komandalar:** RC522 moduli MIFARE bezaklar va boshqa bezak turlari bilan ishlash uchun mo'ljallangan protokollarni qo'llaydi. Kommunikatsiya jarayonida mikrokontrollerdan modulga beriladigan protokol va komandalar orqali bezak ma'lumotlari o'qiladi va yoziladi.
5. **Ma'lumotlar o'qish/yozish:** Modul bezaklar bilan kommunikatsiya o'rnatgandan so'ng, mikrokontroller modulga buyruqlar yuboradi va bezakdan kelgan ma'lumotlarni o'qib oladi. O'qilgan ma'lumotlar mikrokontrollerga uzatiladi va ulardan foydalanuvchi tomonidan istalgan ko'rinishda ishlatiladi.

RC522 moduli bezaklar bilan ishlashda ishlatiladi. Bezaklar, modulning qulayliklarini o'rganish uchun dastur yaratish va modul bilan bog'liq komandalar yuborish lozim. Modulning dokumentatsiyasida komandalar va ularga mos javoblar haqida ko'rsatmalar mavjud bo'ladi.

Shu tarz RFID modullar, bezaklarni identifikatsiya etish, otomatik kartalar, qulayliklar, vaqt hisoblagichi tizimlari, kirish nazorat tizimlari va boshqa maqsadlar uchun foydalaniladi.

RFID (Radio-Frequency Identification) texnologiyasi obyektlarni identifikatsiya qilish va kuzatish uchun radio to'lqinlar yordamida ma'lumotlarni o'qish va yozish imkonini beradi. **RC522** RFID moduli bu texnologiya asosida ishlovchi mashhur modullardan biridir. U past quvvat iste'mol qiladi, arzon va oson ulanish xususiyatiga ega.

RC522 Moduli Xususiyatlari:

1. **Chastotasi:** 13.56 MHz.
2. **Kommunikatsiya interfeysi:** SPI (Serial Peripheral Interface), I2C, UART.
3. **Qiroat masofasi:** Taxminan 5 sm.
4. **Qayta o'qish va yozish:** MIFARE1 standartiga mos keladigan kartalar va o'qiladigan taglar bilan ishlaydi.
5. **Quvvat manbai:** 3.3V.

RC522 RFID Moduli Ishlash Jarayoni

1. Modulli Mikrokontrollerga Ulash

RC522 RFID moduli ko'pincha SPI interfeysi orqali mikrokontrollerga ulanadi. Arduino bilan ulash uchun quyidagi pin konfiguratsiyasi ishlatiladi:

- **SDA (SS):** Pin 10
- **SCK:** Pin 13
- **MOSI:** Pin 11
- **MISO:** Pin 12
- **IRQ:** Ulash shart emas
- **GND:** GND
- **RST:** Pin 9
- **3.3V:** 3.3V

2. Arduino IDE uchun Kutubxonani O'rnatish

RFID moduli bilan ishlash uchun Arduino IDE uchun MFRC522 kutubxonasini o'rnatish kerak. Bu kutubxona RC522 moduli bilan ishlash uchun kerakli funksiyalarni taqdim etadi.

Arduino IDE-ni oching, Sketch -> Include Library -> Manage Libraries menyusiga o'ting va qidiruv maydoniga "MFRC522" yozing. "MFRC522 by GithubCommunity" kutubxonasini tanlang va o'rnatish.

4. Kod Izohi

1. **Kutubxonalarini Qo'shish:**
 - SPI.h: SPI interfeysini boshqarish uchun.
 - MFRC522.h: RC522 modulini boshqarish uchun.
2. **Pinlarni Belgilash:**
 - SS_PIN va RST_PIN pinlari RC522 moduliga ulash uchun ishlatiladi.
3. **Setup Funksiyasi:**

- Serial.begin(9600): Serial aloqa boshlanishi.
- SPI.begin(): SPI interfeysini boshlash.
- rfid.PCD_Init(): RC522 modulini boshlash.

4. **Loop Funksiyasi:**

- rfid.PICC_IsNewCardPresent(): Yangi kartani tekshirish.
- rfid.PICC_ReadCardSerial(): Kartani o'qish.
- rfid.uid.uidByte[i]: Kartaning UID (Unique Identifier) raqamini chiqarish.

RC522 RFID Moduli Qo'llanilishi

RC522 moduli ko'plab qo'llanilish sohasiga ega:

- **Kirishni nazorat qilish tizimlari:** Ofis, uy va boshqa joylarga kirishni boshqarish.
- **Kutubxonalar:** Kitoblarni identifikatsiya qilish va kuzatish.
- **Logistika:** Tovarlarini kuzatish va identifikatsiya qilish.
- **Avtomatlashtirilgan to'lov tizimlari:** Jamoat transporti, avtoturargohlar va boshqa to'lov tizimlari.

RFID texnologiyasi ko'plab sohalarda qo'llaniladi va RC522 moduli uning qulay va arzon echimlaridan biridir. Ushbu modul yordamida siz ko'plab foydali va qiziqarli loyihalarni amalga oshirishingiz mumkin.

S91 Arduinodagi buzzerlar, faol va passiv signallari va ulardan foydalanish.

Arduino dagi buzzerlar aktivligi (aktiv yoki passiv) va ulardan foydalanish quyidagicha:

1. Aktiv buzzer:

- Aktiv buzzerlar o'z kuchlanish manbaiga ega bo'lib, to'g'ridan-to'g'ri Arduino platasi bilan bog'lanadi.
- Ularni Arduino platasi yoki boshqa mikroprotsessorlarning chiqish pinlari orqali boshqarish mumkin.
- Aktiv buzzerlar tegishli kuchlanish (masalan, 5V) berilishi bilan o'ziyoq tovush chiqaradi.
- Aktiv buzzerlar Arduino yordamida oson boshqariladi, dasturlash orqali turli xil tovushlar va melodiylar chaldirishi mumkin.
- Masalan, tone() funksiyasi yordamida turli chastotalarda tovushlar chiqarish mumkin.

2. Passiv buzzer:

- Passiv buzzerlar o'z kuchlanish manbaiga ega emas, Arduino platasi orqali qo'shimcha kuchlanish olishi kerak.
- Ularni Arduino platasi chiqish pinlariga ulanadi va chiqish pinlaridan kuchlanish berib boshqariladi.
- Passiv buzzerlar aktiv buzzerga qaraganda oddiy tuzilishga ega, lekin ularni boshqarish uchun qo'shimcha dasturlash talab qilinadi.
- Passiv buzzerlarni analogWrite() funksiyasi yordamida boshqarish mumkin, bu orqali turli xil tovushlarni chiqarish imkoniyati paydo bo'ladi.

Buzzerlardan foydalanish uchun quyidagi qadamlar bajariladi:

1. Arduino platasi va buzzer (aktiv yoki passiv) orasida elektr ulash.
2. tone() yoki analogWrite() funksiyalarini buzzerning tovush chiqarishi uchun qo'llash.
3. Kerakli melodiya, effekt yoki signallarni yaratish uchun dasturlash.

Buzzerlar signallar, ogohlantirish, melodiya chaldirish, feedback berish kabi maqsadlarda qo'llaniladi. Masalan, "beep" signali, musiqiy effektlar, burgular, kirish-chiqish holatlarini bildiruvchi tovushlar va boshqalar.

Arduinodagi Buzzerlar: Faol va Passiv Signallar

Faol va Passiv Buzzerlar

• Faol Buzzer:

- O'z ichiga osilatorni oladi, ya'ni oddiy yuqori yoki past kuchlanish bilan boshqariladi.
- Arduino'dan to'g'ridan-to'g'ri signal kerak emas.

- Ovoz chiqarish uchun faqat kuchlanish o'zgarishini talab qiladi.
- **Passiv Buzzer:**
 - Osilatorga ega emas, ya'ni o'z kuchlanishini osilator orqali boshqarish kerak.
 - Arduino'dan signalni boshqarish talab etiladi (masalan, PWM).

Buzzerlarni Ulash

Faol Buzzer Ulash

Faol buzzerning ijobiy (+) pinini Arduino'ninng 5V piniga va manfiy (-) pinini Arduino'ninng GND piniga ulash kerak. Agar siz buzzerning ovozini kod orqali boshqarish uchun ishlatmoqchi bo'lsangiz, ijobiy (+) pinini Arduino'dagi raqamli chiqish piniga ulashingiz mumkin.

Passiv Buzzer Ulash

Passiv buzzerning ijobiy (+) pinini Arduino'dagi raqamli chiqish piniga va manfiy (-) pinini Arduino'dagi GND piniga ulash kerak. Passiv buzzerni boshqarish uchun Arduino'da PWM signalini yaratish kerak.

Buzzerlardan Foydalanish

- **Signalizatsiya va Ogohlantirish:** Oddiy ovozli signalizatsiya tizimlari, masalan, eshik yoki deraza ochilganda ogohlantirish uchun.
- **Melodiyalarni ijro etish:** Passiv buzzer yordamida turli chastotalarda ovoz chiqarish orqali oddiy melodiyalarni ijro etish mumkin.
- **Reaksiya tekshirish:** Foydalanuvchi tugmalariga bosganda yoki sensorlar ma'lumot berganda tovushli tasdiqlash signallari.

S92. HC-SR04 ultra tovushli masofa moduli.

HC-SR04, ultrasonik (ultra tovushli) masofa modulidir. Uzun masofalar o'lchash uchun ishlatiladi va Arduino, Raspberry Pi, ESP8266 kabi mikrokontrollerlar bilan birlashtiriladi.

HC-SR04 modulining ishlash jarayoni quyidagicha bo'ladi:

1. **Ultrasonik tovush sifatida ishlash:** HC-SR04 modulida ikkita qism mavjud bo'ladi - tovush izlantiruvchi (transmitter) va qabul qiluvchi (receiver). Transmitter tovush izlantiradi va uni moduldan yuboradi. Qabul qiluvchi esa qaytuvchi tovushni qabul qiladi.
2. **Masofa o'lchash:** Tovush izlantiruvchi tovushni yuborganidan so'ng, uning ovoz tezligi orqali masofa hisoblanadi. Tovush tezligi atmosferda ma'lum bir tezligi (masalan, 340 m/s) bilan tarqaladi. Qabul qiluvchi tovushni qabul qildiktan so'ng, modul bu vaqtni o'lchab, masofani hisoblab chiqaradi.
3. **Ultrasonik dalalar:** HC-SR04 modul ultrasonik dalalar ishlatadi. Bu dalalar ovoz tezligida chiqadi va yorug'likning sesh talqiniga o'xshaydi. Ular yorug'likdan qaytarib kelganida, modul qabul qiladi va masofani hisoblab chiqaradi.
4. **Pulssiz modda ishlash:** HC-SR04 moduli pulssiz (pulse) rejimda ishlaydi. Bu modda, modulga biror buyruq berilganida, u ultrasonik tovushni yuboradi va keyin qabul qiluvchidan kelgan javobni kuzatib boradi.
5. **Mikrokontroller bilan bog'lanish:** HC-SR04 moduli, mikrokontroller (masalan, Arduino) bilan bog'lanadi. Uch ta pinidan iborat bo'lib, ulardan ikkisi tovush izlantiruvchiga bog'lanadi, va bitta pin esa qabul qiluvchiga bog'lanadi. Tovush izlantiruvchi va qabul qiluvchi pinlarining ulanishi uchun mikrokontrollerdagi mos pinlarni tanlash lozim.

HC-SR04 ultrasonik masofa moduli, robotika, masofa o'lchash tizimlari, hinderni aniqlash, yo'lni topish, ishlab chiqarish sohasi va boshqa muhim ilovalarda ishlatiladi. Uning o'rnatilishi va ishlatilishi oson va uni Arduino yoki boshqa mikrokontroller bilan birlashtirish juda qulay.

HC-SR04 Ultrasonik Masofa Moduli

HC-SR04 ultrasonik masofa moduli masofani aniqlash uchun ishlatiladi. Bu modul odatda Arduino bilan ishlatiladi va juda aniq masofani o'lchash imkoniyatiga ega.

HC-SR04 Moduli Texnik Ko'rsatkichlari

- **Ishchi kuchlanish:** 5V DC
- **Quvvat iste'moli:** 15mA
- **Ultrasonik chastota:** 40kHz
- **Maksimal masofa:** 4 metr

- **Minimal masofa:** 2 sm
- **Aniqlik:** 3 mm

Pin Konfiguratsiyasi

1. **VCC:** 5V DC (Arduino'dan 5V piniga ulanadi)
2. **Trig:** Trigger signal (Arduino raqamli chiqish piniga ulanadi)
3. **Echo:** Echo signal (Arduino raqamli kirish piniga ulanadi)
4. **GND:** Zamin (Arduino'dan GND piniga ulanadi)

HC-SR04 Modulini Arduino'ga Ulash

- **VCC:** Arduino 5V
- **GND:** Arduino GND
- **Trig:** Arduino raqamli chiqish pin, masalan, pin 9
- **Echo:** Arduino raqamli kirish pin, masalan, pin 10

Kod Tushuntirish

1. **Pinlarni belgilash:**
 - trigPin va echoPin pinlari Arduino raqamli pinlari sifatida aniqlanadi.
2. **Setup funksiyasi:**
 - pinMode(trigPin, OUTPUT): Trig pinini chiqish rejimida sozlaydi.
 - pinMode(echoPin, INPUT): Echo pinini kirish rejimida sozlaydi.
 - Serial.begin(9600): Serial aloqani 9600 baud tezlikda boshlaydi.
3. **Loop funksiyasi:**
 - **Trigger signali:** Trig piniga qisqa muddatli yuqori signal uzatiladi (10 mikrosoniya).
 - **Echo signali:** Echo pinidan signalni olish va uning davomiyligini o'lchash (mikrosoniyalarda).
 - **Masofani hisoblash:** Olingan davomiylikdan foydalanib masofani hisoblash (tezlik = 343 m/s).
 - **Natijani chiqarish:** Masofa serial monitorga chiqariladi.

Qo'llanilish Soqhalari

HC-SR04 ultrasonik moduli ko'plab qo'llanilish sohalariga ega, jumladan:

- **Robototexnika:** Ob'yektlarni aniqlash va masofani o'lchash.
- **Avtomatik to'xtash tizimlari:** Avtomobillarni park qilishda yordam.
- **Suyuqlik darajasi o'lchash:** Suyuqlik saqlovchi idishlarda darajani o'lchash.
- **Xavfsizlik tizimlari:** Ob'yektlar yaqinlashganda signalizatsiya qilish.

HC-SR04 moduli masofani aniq va ishonchli o'lchash uchun juda foydali va oson foydalaniladigan modul hisoblanadi. Bu modul yordamida ko'plab foydali va qiziqarli loyihalarni amalga oshirishingiz mumkin.

S93.HC-SR04 ultra tovushli masofa modulini loyihalarda qo'llanilishi?

HC-SR04 ultrasonik masofa moduli quyidagi loyihalarda qo'llanilishi mumkin:

1. Robotika va avtonomli transport vositalari:

- Robotlarning masofani o'lchab, to'siqlarni aniqlab, harakat yo'nalishini belgilashda.
- Avtomobil, dron va boshqa transport vositalarida to'g'ri yo'nalishni aniqlash va to'siqlardan xavfsiz harakatlanish.

2. Ogohlantirish va xavfsizlik tizimlarida:

- Xonalardagi insonlarni aniqlab, xavfsizlik tizimlariga signal berish.
- Eshik, derazalar, to'siqlar yoki boshqa narsalarni aniqlab, ogohlantirish signallari chiqarish.

3. Uy-joy va bino avtomatlashtirishda:

- Uskunalarini avtomatik ravishda boshqarish uchun harakatni aniqlab, funksiyalarni ishga tushirish.
- Uy-joy ichidagi narsalarning joylashishini aniqlash va boshqarish.

4. Smart qishloq xo'jaligi va yer ishlari:

- Devarlar yoki to'siqlarni aniqlab, qishloq xo'jaligi texnikasining harakatini boshqarish.
- Namelash, yorilar, zararkunandalardan himoya kabi maqsadlarda qo'llash.

5. Industrual avtomatlashtirishda:

- Ishlab chiqarish konveyerlari, uskuna va mexanizmlarni nazorat qilish.
- Mahsulotlarni avtomatik ravishda saralash, joylashtirish yoki etkazib berish.

6. Inson-mashinalar interfaysida:

- Insonlarning harakatini aniqlab, mashinalarni boshqarish.
- Interfaollik yoki ixcham boshqaruv tizimlarida foydalanish.

HC-SR04 moduli nisbatan arzon va juda keng qo'llaniladi. U o'zining soddaligi, kam energiya sarfi va yuqori aniqlik darajasi bilan ajralib turadi. Shuning uchun u yuqoridagi ko'plab loyihalarda keng qo'llaniladi.

HC-SR04 Ultrasonik Masofa Modulining Loyihalarda Qo'llanilishi

HC-SR04 ultrasonik masofa moduli turli xil loyihalarda keng qo'llaniladi. Uning arzonligi, aniqligi va foydalanish osonligi sababli, bu modul robototexnika, avtomatlashtirish va xavfsizlik tizimlarida muhim rol o'ynaydi. Quyida HC-SR04 moduli yordamida amalga oshirilishi mumkin bo'lgan ba'zi loyihalar keltirilgan.

1. Robototexnika

Ob'yekt Aniqlash va To'siqlardan Qochish

- **Loyihaning Tavsifi:** HC-SR04 moduli robotning atrofidagi ob'yektlarni aniqlash va ularning oldida to'xtash yoki yo'nalishni o'zgartirish imkoniyatini beradi.
- **Foydalanish:** Modulli robotlarning masofani aniqlash va harakatlanishini ta'minlash.

2. Avtomatik Park Tizimi

Avtomobil Park Tizimi

- **Loyihaning Tavsifi:** HC-SR04 moduli yordamida avtomobilni avtomatik park qilish tizimi.
- **Foydalanish:** Avtomobillarni aniq va xavfsiz park qilishga yordam beradi.

cpp

3. Suyuqlik Darajasini O'lchash

Suyuqlik Darajasi Monitoring

- **Loyihaning Tavsifi:** HC-SR04 moduli yordamida suyuqlik saqlovchi idishlarning darajasini aniqlash.
- **Foydalanish:** Suv idishlari yoki yonilg'i tanklarida suyuqlik darajasini aniqlash va monitoring qilish.

4. Xavfsizlik Tizimlari

Xavfsizlik va Harakat Aniqlash

- **Loyihaning Tavsifi:** HC-SR04 moduli yordamida harakat aniqlash va xavfsizlik tizimlarini yaratish.
- **Foydalanish:** Harakat aniqlanganda ogohlantirish berish yoki signalizatsiya tizimini yoqish.

Loyihalarning Qo'shimcha E'tiborlari

- **Aniqlik va Ishonchlilik:** Loyihalarda HC-SR04 moduli foydalanishdan oldin, uning aniqligini va ishonchliligini tekshirish kerak.
- **Atrof-muhit Sharoitlari:** Moddaning ishlashiga ta'sir qilishi mumkin bo'lgan atrof-muhit sharoitlarini hisobga olish.
- **Xavfsizlik:** Elektr xavfsizligini ta'minlash uchun HC-SR04 modulini to'g'ri ulash va foydalanish.

HC-SR04 ultrasonik masofa moduli yordamida loyihalar yaratish juda qiziqarli va foydali bo'lishi mumkin. Bu moduli turli sohalarda, jumladan robototexnika, avtomatlashtirish va xavfsizlik tizimlarida keng qo'llanilishi mumkin.

S94. DC Motor, motorni o'ngga chapga aylantrish .Motor tezligini bosharish?

DC motorlarni o'ngga va chapga aylantrish uchun motorning kuchlanishini moslashtirish kerak. Bu esa motorning tezligini bosharish bilan amalga oshiriladi. DC motorning aylanish yo'nalishi, elektr to'qimasi yoki polari orqali o'zgaradi. O'ngga aylanish uchun motorning polari o'ngga, chapga aylanish uchun esa polari chapga bog'lanadi. Bu bilan motorning aylanish yo'nalishi o'zgaradi.

Motorning tezligini bosharish uchun quyidagi usullardan foydalanish mumkin:

1. **Transistorlar orqali bosharish:** Darlington tranzistorlar yoki MOSFETlar kabi guvohnomali tranzistorlar aracılığıyla motorning boshqa poli bilan bog'langan terminalini kuchlanishini bosharish mumkin. Transistorlar o'ngga yoki chapga bog'langan voltajni o'zgartirish orqali motorning tezligini o'zgartirish imkoniyatini beradi.
2. **H-kesishgan moslamalar:** H-kesishgan moslamalar (H-bridge circuits) motorlarni boshqarish uchun keng qo'llaniladi. Ular motorning aylanish yo'nalishini o'zgartirish imkonini beradi. H-kesishgan moslamalar to'g'ridan-to'g'ri arduino yoki boshqa mikrokontroller bilan birlashtiriladi va moslashuvchilar yordamida motorlarni o'ngga va chapga aylanishini bosharish imkoniyatini beradi.
3. **Puls modulyatsiya (PWM):** DC motorlarni bosharishda PWM ishlatiladi. PWM, daraja bilan uzluksiz voltaj ta'minlash orqali motorning boshlanish vaqti va boshqarishini o'zgartirish imkonini beradi. PWM dasturlash orqali mikrokontrollerdan chiqadigan darajali signal orqali motorlarni bosharish mumkin.

DC Motorni Boshqarish

DC motorlarini Arduino bilan boshqarish uchun odatda H-Bridge motor haydovchisi yoki motor haydovchi moduli (masalan, L298N) ishlatiladi. Ushbu qurilmalar yordamida motorni o'ngga yoki chapga aylantirish va tezligini boshqarish mumkin.

Komponentlar

- Arduino (Uno, Mega, yoki boshqa model)
- DC motor
- H-Bridge motor haydovchi moduli (L298N)
- 9V yoki 12V batareya (motor uchun quvvat manbai)
- Jumper kabellar
- Breadboard (agar kerak bo'lsa)

DC Motorni O'ngga va Chapga Aylantirish

DC motorini o'ngga yoki chapga aylantirish uchun H-Bridge motor haydovchisi ishlatiladi. L298N motor haydovchisi odatda motorni boshqarish uchun qulay vosita hisoblanadi.

H-Bridge Motor Haydovchi Moduliga Ulash

L298N Pin Ulanishlari:

- **IN1 va IN2:** Motor 1 uchun boshqarish signallari (Arduino raqamli chiqish pinlariga ulanadi)
- **ENA:** Motor 1 uchun tezlikni boshqarish (Arduino PWM chiqish piniga ulanadi)
- **VCC:** Motor uchun quvvat (9V yoki 12V batareya)
- **GND:** Zamin (batareya va Arduino GND pinlariga ulanadi)
- **5V:** L298N ichki elektronikasini quvvatlantirish (Arduino 5V piniga ulanadi)

Kod Tushuntirish

1. Pinlarni belgilash:

- enA PWM chiqish pini motor tezligini boshqarish uchun ishlatiladi.
- in1 va in2 raqamli chiqish pinlari motor yo'nalishini boshqarish uchun ishlatiladi.

2. Setup funksiyasi:

- pinMode funksiyasi yordamida enA, in1 va in2 pinlari chiqish rejimida sozlanadi.

3. Loop funksiyasi:

- **Motorni o'ngga aylantirish:** in1 piniga yuqori (HIGH) signal va in2 piniga past (LOW) signal berish orqali amalga oshiriladi. analogWrite funksiyasi yordamida enA piniga 255 qiymat berib motor maksimal tezlikda aylantiriladi.
- **Motorni chapga aylantirish:** in1 piniga past (LOW) signal va in2 piniga yuqori (HIGH) signal berish orqali amalga oshiriladi.
- **Motor tezligini o'zgartirish:** for sikli yordamida motor tezligi bosqichma-bosqich oshiriladi. analogWrite funksiyasi orqali enA piniga turli xil qiymatlar beriladi.

Tezlikni Boshqarish

Motor tezligini boshqarish uchun PWM (Pulse Width Modulation) signallari ishlatiladi. Arduino'da analogWrite funksiyasi yordamida PWM signal yaratilib, motor tezligi boshqarilishi mumkin. PWM signal qiymati 0 dan 255 gacha bo'lishi mumkin, bu esa motor tezligini 0% dan 100% gacha o'zgartirish imkonini beradi.

DC motorlarini Arduino bilan boshqarish orqali ko'plab qiziqarli va foydali loyihalarni amalga oshirishingiz mumkin. H-Bridge motor haydovchi moduli yordamida motor yo'nalishini va tezligini boshqarish oson va samarali usuldir. Ushbu misollar va tushuntirishlar yordamida siz o'zingizning loyihalaringizda DC motorlarini muvaffaqiyatli boshqarishingiz mumkin.

S95. Stepper motor bilan ishlash. Stepper motorni qadamini hisoblash tizimini ishlab chiqish

Stepper motorlar diskret qadamli aylanishni bajaradigan elektronik qurilmalardir. Ularning har bir qadami bilan motor belgilangan burchakka aylanadi. Stepper motorlarni boshqarish uchun qadamini hisoblash tizimini quyidagicha ishlab chiqishingiz mumkin:

1. **Motorning qadamini aniqlash:** Stepper motorlarning bir qadami belgilangan daraja miqdorida aylanishni ifodalaydi. Motorning xususiyatlari (maksimal qadam soni, o'zgaruvchanlik miqdori, qadamlar soni) bo'lgan ma'lumotlarini olishingiz kerak.
2. **Mikrokontroller yoki PLC tanlash:** Stepper motorlarni boshqarish uchun mikrokontroller (masalan, Arduino) yoki PLC (Programmable Logic Controller) dan foydalanishingiz mumkin. Bu apparatlar motorning qadamini hisoblash, boshqarish va boshqa kerakli funksiyalarni o'zida o'zlashtirgan dasturlarni yuklash imkonini beradi.
3. **Motorning boshqarish interfeysini tuzish:** Stepper motorlarni boshqarish uchun moslashuvchilar (driver) va moslashuvchilar bilan bog'lanish interfeysini tuzishingiz kerak. Bu interfeys arduino, PLC yoki boshqa mikrokontroller bilan bog'lanadi. Moslashuvchilar motorning boshqa polari bilan bog'lanadi va ularga buyruqlar berish orqali motorning aylanish yo'nalishini va tezligini boshqarish imkoniyatini beradi.
4. **Qadam hisoblash algoritmasini yaratish:** Stepper motorning qadamini hisoblash uchun avvalgi qadamni eslab qolish va yangi qadamlarga o'tish jarayonini aniqlashning algoritmasini yaratishingiz kerak. Bu algoritma mikrokontroller tomonidan bajariladi va moslashuvchilarga buyruqlar yuboradi.
5. **Kontrol dasturini yaratish:** Qadam hisoblash algoritmasini ishga tushirish uchun bir kontrol dasturini yaratishingiz kerak. Bu dastur motorning qadamini hisoblaydi, yangi qadamlarga o'tish buyruqlarini generatsiya qiladi va moslashuvchilarga yuboradi.

Stepper motorlarni boshqarishda mikrokontroller va moslashuvchilar keng qo'llaniladi. Dasturlash tilini (masalan, Arduino dasturlash tilini) bilish kerak va motorning xususiyatlari va talablari bo'yicha kerakli elektronik qurilmalarni va interfeysni tanlash lozim. Tizimni ishlab chiqishda elektronika, dasturlash va elektronik qurilmalar bilan mashg'ul bo'lishingiz mumkin.

Stepper motorlar, belgilangan miqdorda harakatlar qilish uchun juda mos maqsadga mo'ljallangan. Ular nomi bo'yicha, har bir "qadam" (step) deb nomlangan belgilangan miqdorda harakatni bajarganida, bu qadamlar soni bilan birlashtiriladi. Stepper motorlar bilan ishlash uchun qadam hisoblash tizimi quyidagi komponentlardan iborat bo'ladi:

1. **Stepper Motor:** Bu motorlar, boshqa motorlarga nisbatan o'ziga xos bo'lgan dizayn va ishlab chiqarilgan, qadamli harakatni bajargan.
2. **Driver:** Stepper motorlarni boshqarish uchun zarur bo'lgan elektronika modul. Bu modul arduino yoki boshqa mikrokontroller bilan bog'lanadi va motorning xarakterlarini boshqarish uchun elektr energiya ni ushlab turadi.
3. **Mikrokontroller:** Arduino, Raspberry Pi yoki boshqa mikrokontroller, stepper motor va driver moduliga ulanish va uning harakatlarini boshqarish uchun ishlatiladi.
4. **Kod yozish dasturi:** Kodni yozish va arduino yoki mikrokontrollerga yuklash uchun kompyuterda dastur yoki IDE.

Qadam Hisoblash Tizimi Uchun Tushuntirish

Stepper motorlar, xususan, belgilangan qadam hisoblanib, to'xtaydi. Har bir qadamning kengligi, motorning o'lchamiga va uning qadamli harakatini aniqlash uchun kerakli o'lchamlar soniga bog'liq. Qadam hisoblash tizimi quyidagi tartibda ishlaydi:

1. **Stepper Motor Va Driver Ulanishi:** Stepper motor driver moduli arduino yoki boshqa mikrokontrollerga bog'lanadi. Ular orasidagi bog'lanish schema (wiring diagram) asosiy xaritalar bilan yordamida bajariladi.
2. **Qadam Uchun Kodni Yozish:** Stepper motorlarni boshqarish uchun kod yozish kerak bo'ladi. Ushbu kod, motorning har bir qadam uchun kerak bo'lgan signalni jo'natadi.
3. **Stepper Motorni Boshqarish:** Kod, stepper motorning har bir qadam uchun belgilangan miqdorda harakatlanishini ta'minlaydi. Misol uchun, agar 200 qadamli stepper motorni aylanish uchun boshqarishni aniqlaysiz, kod 200 marta "forward" (oldinga) yoki "backward" (orqaga) komandalarni jo'natadi.
4. **Qadam Hisoblash:** Har bir belgilangan qadamda stepper motor xaritalarini o'rganadi va qadam sonini hisoblaydi.
5. **Natija:** Stepper motor, belgilangan qadam hisoblanganida, kerakli harakatni bajargan. Natijada, ushbu harakat bajarilganligi hisoblanadi.

S96.Servo motorni ishlash jarayoni. Servo motor orqali robor qo'llarini boshqarish.

Servo motorlar ro'yxatdagi burchakni aniqlab, belgilangan burchakga aylanishni bajaradigan elektronik qurilmalardir. Ular burchakni belgilangan joyga olib kelish uchun qulaylik va to'g'ri hisoblanish imkoniyatini beradi. Servo motorlar ro'yxatdagi qo'llardan biri bo'lib, ular orqali robot qo'llarini boshqarish quyidagi jarayonlardan iborat bo'ladi:

1. **Servo motorlarni tanlash:** Ba'zi servo motorlar haqiqiy burchakni topish uchun potensiyometr (burchakni o'qish uchun datchik) bilan birlashtirilgan bo'lishi mumkin. Bunday servo motorlarni tanlashda, burchak miqdorini ko'rsatuvchi o'qning oynasini aylantirib, shunchaki robot qo'llaridagi burchakni aniqlashingiz mumkin.
2. **Servo motorlarni boshqarish interfeysini tuzish:** Servo motorlarni boshqarish uchun moslashuvchilar (servo motor boshqaruvchi) bilan bog'lanish interfeysini tuzishingiz kerak. Bu interfeys arduino, mikrokontroller yoki boshqa boshqaruvchilar bilan bog'lanadi. Moslashuvchilar servo motorning burchak polini boshqarib, belgilangan burchak miqdoriga olib keladi.
3. **Burchak belgilash algoritmasini yaratish:** Servo motorlarni burchakni belgilash uchun algoritma yaratishingiz kerak. Bu algoritma servo motorning burchak polini o'zgartirish va belgilangan burchakga olib kelish jarayonini aniqlashning buyruqlarini o'z ichiga oladi. Algoritma servo motorning burchak polini to'g'ri o'zgartirish, yashirish va belgilangan burchakga olib kelishni amalga oshiradi.
4. **Kontrol dasturini yaratish:** Burchak belgilash algoritmasini ishga tushirish uchun bir kontrol dasturini yaratishingiz kerak. Bu dastur servo motorning burchak polini belgilaydi, belgilangan burchakka olib kelish buyruqlarini generatsiya qiladi va moslashuvchilarga yuboradi.

Servo motorlar burchakni o'zgartirish imkonini beradi. Ular robot qo'llarini boshqarishda, burchakni belgilash va o'zgartirish uchun qulay va to'g'ri vositalardir. Servo motorlarni boshqarishda Arduino, Raspberry Pi yoki boshqa mikrokontrollerlar keng qo'llaniladi. Dasturlash tilini bilish, servo motorning xususiyatlari va talablari bo'yicha moslashuvchini tanlash va moslashuvchini boshqarish interfeysini tuzish lozim.

Servo motorlar, boshqa motorlarga nisbatan maxsus bo'lib, belgilangan burch va yo'nalishda turli vazifalarni bajarish uchun ishlab chiqarilgan. Ular robotik qo'llarni boshqarish, dronlarni yo'nalish, avtomobil modelini qo'ldan boshqarish va boshqalar kabi ommaviy vazifalarni bajarishda juda yaxshi foydalaniladi. Servo motorlarni boshqarish uchun quyidagi komponentlar kerak bo'ladi:

1. **Servo Motor:** Bu, burch va yo'nalishda maxsus harakatni amalga oshiradigan motor. Uning xususiyati, belgilangan burch va pozitsiyalarda turli vazifalarni bajarishi.
2. **Arduino yoki boshqa mikrokontroller:** Servo motor boshqarish uchun elektronik komponentlar.
3. **Servo motor driver:** Servo motorlarni boshqarish uchun moslashtirilgan driver.
4. **Breadboard va jumper kabellari:** Komponentlar ulanish uchun.
5. **Energiyani ta'minlash:** Arduino va servo motor uchun ta'minlanadigan energiya manbai.

Servo Motorlarni Boshqarish Tizimi

Servo motorlarni boshqarish tizimi quyidagi tartibda ishlaydi:

1. **Servo Motor Ulanishi:** Servo motor driveri orqali Arduino'ga bog'lanadi.
2. **Arduino'ga Servo Motor Ulanishi:** Servo motor bog'lanish joyiga moslashtiriladi.
3. **Servo Motor Uchun Kod Yozish:** Arduino'ga servo motor boshqarish uchun kerakli kodlar yoziladi.
4. **Arduino'ga Yozilgan Kodni Yuklash:** Kodlar Arduino'ga yuklanadi va servoga boshqarish amalga oshiriladi.
5. **Servo Motorni Boshqarish:** Servo motor ko'rsatkichlari moslashtiriladi va kerakli burch va pozitsiyaga yo'naltiriladi.

S97. Temperature DHT11 va DHT22 sensor moduli bilan ishlash va ularni imkoniyatlari.

Temperature va namlik aniqlash uchun DHT11 va DHT22 sensor modullaridan foydalanish mumkin. Ular temperaturani Celsius yoki Fahrengeyt graduslarida, namlikni esa foiz (%) shaklida o'lchay oladi. Bu modullarning quyidagi imkoniyatlari mavjud:

1. DHT11 sensor moduli:

- Temperaturani 0°C dan 50°C gacha o'lchaydi, 2°C aniqlikda.
- Namlikni 20% dan 90% gacha o'lchaydi, 5% aniqlikda.
- Qulay va arzon sensor moduli, odatda 3-4 dollar narxida.
- Kamroq murakkab, datchik va mikrokontroller o'rtasida faqat 3 ta signal o'tish yo'llariga ega.
- Sinxron ma'lumot uzatish protokoliga asoslangan.

2. DHT22 sensor moduli:

- Temperaturani -40°C dan +80°C gacha o'lchaydi, 0.1°C aniqlikda.
- Namlikni 0% dan 100% gacha o'lchaydi, 0.1% aniqlikda.
- Murakkabroq sensor moduli, odatda 5-10 dollar narxida.
- Datchik va mikrokontroller o'rtasida 4 ta signal yo'llariga ega.
- Asinkron ma'lumot uzatish protokoliga asoslangan.
- DHT11 ga qaraganda aniqroq o'lchovlarga ega.

Ushbu sensrlar arduino, raspberry pi yoki boshqa mikrokontroller platformalari bilan oson integratsiya qilinishi mumkin. Mikrokontroller maxsus kutubxanalar orqali sensorlar bilan ishlab, ulardagi ma'lumotlarni o'qib, temperaturani va namlikni aniqlashi, hamda boshqa funktsiyalarni bajarishi mumkin.

DHT11 va DHT22 sensorlarni tanlashda, o'lchov oralig'i, aniqlik, narx va mikrokontroller bilan integratsiya qilish osonligi kabi parametrlarni e'tiborga olish kerak. Masalan, yuqori aniqlik talab qilinadigan loyihalar uchun DHT22 yaxshiroq, agar aniqlik kam muhim bo'lsa, arzonroq DHT11 yetarli bo'ladi.

DHT11 va DHT22 temperatur va haroratni (temperature and humidity) o'lchashda ishlatiladigan keng qo'llaniladigan sensor modullardir. Ular burch, harorat va namiqni o'lchab beradi. Ular odatda mikrokontrollerlar (masalan, Arduino) bilan bog'lanadi va oddiy dasturlash tilida boshqariladi.

DHT11 va DHT22 Sensorni Ishlatish

DHT11 va DHT22 sensorlarini ishlatish uchun quyidagi komponentlar kerak:

1. **Arduino (yoki boshqa mikrokontroller):** Sensor modullarni boshqarish uchun kerakli elektronika.

2. **DHT11 yoki DHT22 sensor moduli:** Temperatur va haroratni o'lchash uchun sensor.
3. **Breadboard va jumper kabellari:** Sensor modulini mikrokontrollerga ulash uchun.
4. **USB kabellari:** Arduino'ni kompyuterga ulash uchun.

Kodni Tushuntirish

1. **DHT Kütüphanesi:** DHT sensor modulini ishlatish uchun DHT kütüphanesi yuklandi.
2. **Sensor Ulanishi:** DHT sensor modulining bog'lanishi Arduino piniga belgilandi (DHTPIN).
3. **Sensor Modulini Boshlash:** dht.begin() funksiyasi yordamida DHT sensor moduli boshlanadi.
4. **Loop Funksiyasi:** loop funksiyasida har 2 sekunda bir harorat va namlik o'lchandi va konsolga chiqariladi (Serial.print orqali).

DHT11 va DHT22 Sensorlarining Imkoniyatlari

- **DHT11:**
 - Temperatur o'lchash uchun: 0 °C dan 50 °C gacha ($\pm 2^{\circ}\text{C}$)
 - Namlik o'lchash uchun: 20% dan 80% gacha ($\pm 5\%$)
 - Iqtisodiy va soddalik sensori, ammo kam ta'minoti uchun moslashtirilgan
 - Uy va oddiy mahallarda amalga oshirilish uchun moslashtirilgan
- **DHT22:**
 - Temperatur o'lchash uchun: -40 °C dan 80 °C gacha ($\pm 0.5^{\circ}\text{C}$)
 - Namlik o'lchash uchun: 0% dan 100% gacha ($\pm 2-5\%$)
 - Yaxshi aniqligi va moslashtirilishi

DHT11 va DHT22 sensor modullari, oddiy sensorlardir, ular temperatur va namlikni o'lchab beradi. Ular Arduino yoki boshqa mikrokontrollerlar bilan moslashtiriladi va harorat va namlik ma'lumotlarini to'plab olish uchun o'zgartirilgan dastur yozish uchun qulaydir. Ular oddiy loyihalarda va hamma turlar bilan qo'llaniladi, masalan, iklim nazoratida, uy otomatsionida va boshqa mahallarda.

S98. HCSR501 Xarakat sensor moduli?

HC-SR501, harakat (xarakat) sensori yoki PIR (Passive Infrared) sensori deb nomlanadi. Ushbu sensor moduli harakatni aniqlash uchun infratuzilishdan foydalanadi. HC-SR501 sensor moduli quyidagi imkoniyatlarga ega:

1. **Harakatni aniqlash:** HC-SR501, odamlar yoki ob'ektlarning infratuzilishini qayta tiklash orqali harakatni aniqlaydi. Uning tizimi PIR sensori va infratuzilish qayta tiklash elementlaridan iborat bo'lib, harakat aniqlandikda elektr signalni chiqaradi.
2. **O'zgaruvchanlik:** Sensor modulida harakatni aniqlash uchun o'zgaruvchanlik sozlamalari mavjud. Ular orqali sensorning tepasidagi infratuzilish tizimining aniqini, harakatga javob berish va vaqtni sozlash mumkin.
3. **Moslashuvchanlik:** HC-SR501 sensori, arduino, Raspberry Pi, ESP8266 va boshqa mikrokontroller platformalari bilan moslashuvchan. U sensorning signal chiqarish pinini mikrokontrollerga bog'lab, harakatni aniqlash va uni boshqarish imkoniyatini yaratadi.
4. **Ishlab chiqarish qulayligi:** HC-SR501 sensor moduli o'zgarishsiz harakatni aniqlab beradi va ishlab chiqarish jarayonida oson va qulaylik bilan qo'llaniladi. Uning o'rnatish va boshqarish jarayonlari oson va moslashuvchanlik bilan amalga oshiriladi.
5. **Energiya samaradorligi:** HC-SR501 sensori juda kam energiya sarflaydi va ishlab chiqarishda kamroq quvvat talab qiladi. Bu sababli, batareya yoki boshqa portabellik istemolarda ishlatilishi uchun mos keladi.

HC-SR501 sensori harakatni aniqlash uchun juda keng qo'llaniladi. U xususiyatlari va narxi uning osonlik va moslashuvchanlik bilan bog'liqdir. Harakatni aniqlash, xavfsizlik, avtomatik yorug'lik tizimlari va boshqa turdagi loyihalarda foydalanish uchun mosdir.

HC-SR501 xarakat (bewegung) sensor moduli, harakatni aniqlash uchun ishlatiladi va Arduino, Raspberry Pi, ESP8266 va boshqa mikrokontrollerlar bilan integratsiya qilinadi. Ushbu sensor moduli oddiydan ko'p ishlatiladigan qurilma, uyotmasinigga yoki ish joyiga yaqinroq joylashgan maydonlarni nazorat qilish uchun ishlatiladi. Harakat aniqlaganda, sensor chiqishini o'zgartiradi, shuningdek, harakatli ob'ektni aniqlashda juda ishonchli bo'ladi.

HC-SR501 Xarakat Sensorining Tarkibi:

1. **Pyroelektrik sensor:** Bu sensor, harakatni aniqlash uchun infrachiziqliq (IR) nurlarini qabul qiladi va undagi harakatni aniqlaydi.

2. **Xarakat kengligi va doimiylikni o'zgartirish uchun potensial sensor:** Sensor harakatga javob berish kengligini va vaqtni moslash uchun imkoniyat beradi.
3. **Mikrokontroller va elektronika:** Sensor modulining ichida, harakatni aniqlash va sensor tavsifi uchun kerakli elektronika va mikrokontroller joylashgan.

HC-SR501 Xarakat Sensorining Ishlatilishi:

1. **Sensor Modulining Bog'lanishi:** HC-SR501 sensor moduli, Arduino, Raspberry Pi yoki boshqa mikrokontrollerga bog'lanadi. Ular sensor modulining signal (OUT), quvvat (VCC) va yagona uzatish (GND) pinlariga ulanadi.
2. **Programmi Yozish:** Arduino IDE yoki boshqa dasturlash muhiti yordamida, sensor modulini boshqarish uchun kerakli kodni yozing.
3. **Harakatni Aniqlash:** Sensor moduli harakatni aniqlagan paytda signalni chiqaradi. Harakat yuzaga kelsa, sensorning chiqishi o'zgaradi.
4. **Sensor Modulining O'rniga Yozilgan Kodni Ishtirok etishi:** Yozilgan kod, sensor modulini boshqarish uchun kerakli komandalarni o'z ichiga oladi. Misol uchun, harakat aniqlangan paytda qurilma yoki elektronik jihozlarni ishga tushirish.

Xulosa:

HC-SR501 xarakat sensori, harakatni aniqlash uchun moslashtirilgan sensor moduli, juda ko'p joyda qo'llaniladi. Uning o'ziga xos chiziqlari va arduino, Raspberry Pi, ESP8266 va boshqa mikrokontrollerlar bilan moslashtirilishi juda sodda. Bu sensor moduli, ish joyini yoki uyning xavfsizlik nazoratini yaxshilash, avtomatik yorug'likni yo'naltirish, va boshqa amaliyotlarda ishlatiladi.

S99. Bir razryadli 7ta segmentli indikator haqida tushuncha va ishlash jarayoni?

Bir razryadli 7ta segmentli indikator - bu yorituvchi displey qurilmasi bo'lib, faqat bitta raqam (0 dan 9 gacha) ko'rsatish uchun ishlatiladi. Uning ishlash jarayoni quyidagicha:

1. **Tuzilishi:** 7ta segmentli indikator asosan 7 ta LED (yorug'lik diodi) dan iborat bo'ladi. Ular a, b, c, d, e, f, g - deb nomlangan yettitta segmentga bo'lingan. Indikator yana dot va decimal point segmentiga ega bo'lishi mumkin.
2. **Raqam ko'rsatish:** Hamma segmentlar o'chirilgan holatda displey hech qanday raqam ko'rsatmaydi. Raqamlarni ko'rsatish uchun tegishli segmentlar yoqiladi. Masalan, "8" raqamini ko'rsatish uchun barcha 7 ta segmentni yoqish kerak.
3. **Ishlash prinsipi:** Har bir segment alohida LED bo'lganligi sababli ularni boshqarish uchun mikrokontroller(masalan arduino)dan foydalaniladi. Mikrokontroller avvaldan o'rnatilgan dastur asosida tegishli segmentlarni yoqib, kerakli raqamni ko'rsatadi.
4. **Raqamlar ko'rsatish:** 7ta segmentli indikator faqat 0 dan 9 gacha raqamlarni ko'rsatadi. Bunda mikrokontroller tomonidan segmentlarni to'g'ri yoqish va o'chirish amalga oshiriladi.
5. **Samaradorlik:** 7ta segmentli indikator juda oddiy va samarali displeyli qurilma hisoblanadi. U kam energiya sarflaydi, oson o'rnatiladi va bir raqamni ko'rsatish uchun yetarli. Bu kabi belgi displeylari keng qo'llaniladi: soatlar, termometrlar, o'lchovchilar va h.k.

Bir razryadli 7ta segmentli indikatorlarni mikrokontroller (arduino, raspberry pi va boshqalar) bilan oson bog'lash va ularni avtomatik boshqarish mumkin. Ular sodda va arzon bo'lgan, lekin faqat bir raqam ko'rsatish imkoniyatiga ega.

Bir razryadli yoki oddiy 7 segmentli indikator, sonlar, harflar va boshqa belgilar bilan sonni ko'rsatish uchun ishlatiladi. U bu qisqartirilgan to'g'ri dengiz koordinatlarini va segmentlarni ishlatadi, har bir sonni yoki harfni ko'rsatish uchun. Har bir segment (a dan g ga) uning koordinatalarini ko'rsatadi. Agar segment ko'rsatilgan bo'lsa, energiya uzatilganida yorqinlik chiqadi.

7 Segmentli Indikatorning Ishlashi:

Oddiy bir razryadli 7 segmentli indikator bilan ishlash uchun, moslashtirilgan mikrokontroller (masalan, Arduino) kerak. Quyidagi komponentlar kerak:

1. **Arduino:** Ishlash uchun mikrokontroller.
2. **7 segmentli indikator:** Burch yoki harfni ko'rsatish uchun.
3. **220 Ohm rezistorlar:** Segmentlarga quvvat berish uchun.
4. **Breadboard va jumper kabellari:** Komponentlarni ulash uchun.
5. **Energiya manbai:** Arduino va indikator uchun energiya ta'minlash uchun.

7 Segmentli Indikatorning Bog'lanishi:

1. 7 segmentli indikatorning birinchi tomoniga to'g'ri birlikning (A-G) ildizi bog'lanadi.
2. Boshqa tomon Arduino pinlariga bog'lanadi (kommonly anod yoki kommonly katod).

7 segmentli indikatorlar sonni, harfni yoki boshqa belgani ko'rsatish uchun ishlatiladi. Har bir segment energiya uzatilganda yorqinlik chiqaradi va kerakli belgi yoki raqam ko'rsatiladi. Bu qurilma ko'pincha arduino yoki boshqa mikrokontrollerlar bilan birlashtiriladi. Yukoridagi kodda, oddiy sonni ko'rsatish uchun 7 segmentli indikator bog'landi. Yangi segmentlar boshqa sonni yoki harfni ko'rsatish uchun moslashtiriladi.

S100. Rele. Ikki va undan ortiq segmentli relelar haqida tushuncha va ishlash jarayoni.

Relelar elektron energiyani mexanik energiyaga aylantirish uchun ishlatiladigan qurilmalar hisoblanadi. Ikki va undan ortiq segmentli relelar, kuchli elektronik devicelarni boshqarish uchun keng qo'llaniladi. Ularning tushunchasi va ishlash jarayoni quyidagicha:

1. **Tuzilishi:** Segmentli relelar, elektromagnitlik bir shtangani (solenoid) va bir yoki undan ortiq elektr kalit (kontakt)dan iborat bo'ladi. Solenoid tomonidan yaratilgan elektromagnit kuchasi, elektr kalitni boshqarish uchun ishlatiladi.
2. **Ikki segmentli rele:** Ikki segmentli relelarda yagona elektromagnit va ikkita elektr kalit (kontakt) mavjud bo'ladi. Bu kalitlar o'zaro aloqada bo'lib, elektr to'plamlarini yo'naltirish va ularga energiya yetkazish uchun ishlatiladi.
3. **Undan ortiq segmentli rele:** Undan ortiq segmentli relelarda ikki yoki undan ortiq elektromagnit va kalitlar mavjud bo'ladi. Bu turdagi relelarda bir nechta elektr kalitlar o'zaro bog'lanishi orqali kompleks elektr loyihalarni boshqarish mumkin.
4. **Ishlash prinsipi:** Relelar elektr to'plamlariga kuch yetkazish uchun ishlatiladi. Elektr to'plamni boshqarish uchun rele solenoidining to'g'ridan-to'g'ri ishlaydigan kuchini qo'llash kerak. Solenoidga elektr darajasi yetkazilib, undan olingan mexanik kuch kalitlarga ta'sir etadi va ularni yo'naltiradi yoki ochadi.
5. **Ishlatish:** Relelar, bir nechta maqsadlarda ishlatilishi mumkin. Ularni elektr energiya tarmog'i, avtomobillar, klimatik vositalar, qurilmalar, termostatlar, fotoapparatlar va boshqa elektron qurilmalar boshqarish uchun foydalanish mumkin.

Relelar katta kuch va energiya ko'rsatkichlarini boshqarish uchun keng qo'llaniladi. Ularning segmentlar soni o'zgartiklariga qarab o'zgaradi, ammo tushunchalari o'zaro bog'liq bo'ladi. Segmentli relelar elektron va elektr texnologiyalarda muhim qurilmalar hisoblanadi.

Relelar, elektr energiyasini boshqa qurilmalar uchun yo'naltirish va boshqarish uchun ishlatiladi. Ular elektromagnit kuchini ishlatib, qavatlar (contacts)ni yo'naltiradilar. Ikki va undan ortiq segmentli relelar esa bir nechta qavatlar (contacts) va ikki yoki undan ortiq elektromagnitlarni ishlatadilar.

Ikki Segmentli Rele:

Ikki segmentli rele, faqat ikkita kontak qavat (contact) mavjud bo'ladi: otkritish (NO - Normally Open) va yopilish (NC - Normally Closed). Elektromagnitning band bo'lishi payti, qavatlar boshlang'ich holatda bo'ladi (NO - boshlang'ich ochiq va NC - boshlang'ich yopiq). Elektromagnit energiyaga loyiqlashganida, NO qavat yopiladi va NC qavat ochiladi.

Undan Ortiq Segmentli Rele:

Undan ortiq segmentli relelarda bir nechta qavatlar (contacts) mavjud bo'ladi. Ular katta miqdorda ko'rsatkichlarni yo'naltirish uchun ishlatiladi. Ko'rsatkichning holati ham bir vaqtning o'zida ko'rsatilishi mumkin.

Ikki Segmentli Relening Tuzilishi:

Ikki segmentli rele, elektr qavatlarini (contacts) yo'naltirish uchun elektromagnitni ishlatadi. Uning tuzilishi quyidagicha:

1. **Elektromagnit:** Elektromagnit stator va rotor bilan ishlaydi. Elektr energiya uzatilganida, elektromagnitni qiynaydigan elektro o'rnatish yaratiladi.
2. **Rotor:** Elektromagnitning band bo'lishi payti, rotor, kontaktlar bilan bog'langan mexanik uzatishlar (arms) yordamida qiynaydi.
3. **Qavatlar (Contacts):** Ikki segmentli releniki faqat ikkita qavat (contact) bo'ladi: NO (o'tkirish) va NC (yopilish). Elektromagnit band bo'lishi paytida, NO qavati ochiq va NC qavati yopiq bo'ladi.

Undan Ortiq Segmentli Relening Tuzilishi:

Undan ortiq segmentli relelarda ikkita yoki undan ko'proq kontak qavat (contacts) mavjud bo'ladi. Ular segmentning kengligiga bog'liq ravishda qavatlar yoki to'g'ridan to'g'ri o'zaro bog'lanadi.

Relening Ishlash Jarayoni:

Relelar, elektr qavatlarini yo'naltirish uchun moslashtiriladi. Quyidagi komponentlar kerak bo'ladi:

1. **Rele:** Rele moduli (ikkita segmentli yoki undan ortiq segmentli).
2. **Arduino (yoki boshqa mikrokontroller):** Relelarni boshqarish uchun.
3. **Breadboard va jumper kabellari:** Komponentlarni bog'lash uchun.
4. **Xulosa:**
5. Relelar, elektr qavatlarini yo'naltirish va boshqa qurilmalar bilan aloqani boshqarish uchun ishlatiladi. Ikki segmentli relelarda, faqat ikkita qavat (kontakt) mavjud bo'ladi: otkirish (NO) va yopilish (NC). Undan ortiq segmentli relelarda esa, bir nechta qavatlar

S101 Mobil ilovalarning rivojlanish tarixi. Dastlabki mobil qurilmalar va mobil ilovalar. Zamonaviy mobil OT lar.

Mobil ilovalarning rivojlanish tarixi quyidagicha:

Dastlabki mobil qurilmalar va mobil ilovalar:

1980-yillarda dastlabki analog mobil telefonlar (NMT, AMPS, TACS) paydo bo'lgan. Ular og'ir, qimmat va cheklangan xususiyatlarga ega bo'lgan.

1990-yillarning boshlarida digital mobil telefonlar (GSM, CDMA, TDMA) rivojlana boshladi. Ular yangi ovoz va SMS xizmatlarini taqdim etdi.

Dastlabki smartfonlar (Palm, Blackberry) va ular uchun ilovalar 1990-yillarning oxirida paydo bo'ldi. Bular ish vaqtini boshqarish, e-pochta, instant messenger kabi asosiy vazifalarni bajargan.

Zamonaviy mobil OT lar:

2007-yilda iPhone va iOS tizimi chiqdi. Bu yangi ko'p funktsiyali smartfonlar davriga asos soldi.

Android platformasi 2008-yilda paydo bo'ldi va tezda smartfonlar bozoriga yetakchi operatsion tizim sifatida kirib keldi.

Keyingi yillarda mobil ilovalar juda tez rivojlandi. Ular turli xizmatlarni taqdim etadigan, ixtisoslashgan, interaktiv va multimediali bo'lib bordi. Zamonaviy mobil OT lari sifatida iOS, Android va Windows Phone/Mobile kabi yetakchi platformalar mavjud. Ular har xil xususiyatlar, dasturlash vositalari va app-bozorlariga ega. Umumiy qilib aytganda, mobil texnologiyalar juda tez rivojlanmoqda va hayotimizning bir qismiga aylanib bormoqda. Mobil ilovalar rivojlanish tarixi va mobil operatsion tizimlar haqidagi ma'lumotlar quyidagicha taqdim etilgan:

Dastlabki mobil qurilmalar va ilovalar

1. Dastlabki mobil qurilmalar:

- Motorola DynaTAC (1983) : Ilk tijoriy mobil telefonlardan biri bo'lib, juda katta va og'ir edi. Ushbu telefon asosan oddiy qo'ng'iroqlar uchun ishlatilgan.

- Nokia 1011 (1992) : Bu GSM tarmog'ida ishlaydigan birinchi tijoriy mobil telefon edi. U juda oddiy funktsiyalarni, jumladan, qo'ng'iroqlar va SMS jo'natishni qo'llab-quvvatladi.

2. Dastlabki mobil ilovalar:

- O'yinlar : 1990-yillarda Nokia telefonlarida mavjud bo'lgan "Snake" (Ilon) o'yini mobil ilovalar tarixida muhim rol o'ynadi.
- Kontaktlar va Kalendari : Ilk mobil qurilmalar oddiy telefon kitoblari va taqvim ilovalari bilan jihozlangan edi.

Zamonaviy mobil operatsion tizimlar

1. Symbian OS : Bu operatsion tizim 2000-yillarda Nokia qurilmalarida juda mashhur edi. Symbian dasturlash tili C++ da bo'lib, multitasking va ko'plab ilovalarni qo'llab-quvvatladi.
2. BlackBerry OS : BlackBerry qurilmalari uchun maxsus yaratilgan. Ular o'zining xavfsizligi va biznes uchun mosligi bilan tanilgan edi.
3. iOS : 2007-yilda Apple tomonidan ishlab chiqilgan. Bu operatsion tizim iPhone va iPad qurilmalarida ishlatiladi. iOS App Store orqali foydalanuvchilarga millionlab ilovalarni taklif qiladi.
4. Android : 2008-yilda Google tomonidan ishlab chiqilgan va hozirda dunyoda eng keng tarqalgan mobil operatsion tizim hisoblanadi. Android Google Play Store orqali millionlab ilovalarga kirish imkonini beradi.
5. Windows Phone : Microsoft tomonidan ishlab chiqilgan bo'lib, zamonaviy va interaktiv interfeys bilan ajralib turadi. Keyinchalik Windows 10 Mobile ga aylantirilgan, ammo hozirgi kunda foydalanishdan to'xtatilgan.
6. Tizen : Samsung tomonidan yaratilgan va asosan aqlli soatlar va boshqa IoT qurilmalarida ishlatiladi. Shuningdek, ba'zi Samsung smartfonlarida ham mavjud.

Mobil ilovalar rivojlanishi

1. Ilk App Store'lar va Marketlar:

- Apple App Store (2008) : iOS ilovalari uchun rasmiy do'kon. Dasturchilar uchun katta platforma bo'lib, mobil ilovalar iqtisodiyotini rivojlantirishda muhim rol o'ynadi.
- Google Play Store (2008) : Android ilovalari uchun rasmiy do'kon. Ushbu platforma ham tezkor rivojlandi va millionlab ilovalarni taklif qilmoqda.

2. Ilovalar turlari:

- Ijtimoiy tarmoqlar : Facebook, Instagram, Twitter va TikTok kabi ilovalar mobil ilovalarning keng tarqalishiga katta hissa qo'shdi.

- Produktivlik ilovalari : Microsoft Office, Google Docs va Evernote kabi ilovalar mobil qurilmalardan ish yuritishni osonlashtirdi.

- O'yinlar : Mobil o'yinlar sanoati katta daromad keltiruvchi sektorga aylandi, Candy Crush, PUBG Mobile va Genshin Impact kabi o'yinlar buning misoli.

3. Moliyaviy texnologiyalar : Mobil to'lov ilovalari, masalan, PayPal, Venmo va Apple Pay, shuningdek, banking ilovalari moliyaviy xizmatlardan foydalanishni osonlashtirdi.

Mobil ilovalar va qurilmalar rivojlanishi texnologiyaning kundalik hayotimizga qanday ta'sir qilayotganligini yaqqol namoyon qiladi. Bu jarayon mobil ilovalar ishlab chiqish sohasida innovatsiyalarni davom ettirish va yangi texnologiyalarni joriy etish bilan davom etadi.

S102. Mobil ilovalarni yaratish tillari va texnologiyalari: iOS va Android uchun dasturlash, emulyatorlar

Mobil ilovalarni yaratish uchun ishlatiladigan asosiy dasturlash tillari va texnologiyalari quyidagicha:

iOS uchun dasturlash

1. Dasturlash tillari:

- Swift : Apple tomonidan ishlab chiqilgan va 2014-yilda taqdim etilgan. Swift xavfsizlik, tezlik va zamonaviy sintaksis bilan ajralib turadi.
- Objective-C : Ilgari iOS ilovalarini yaratish uchun asosiy til bo'lgan. Hozirda Swift bilan birgalikda ishlatiladi va eski loyihalar uchun qo'llaniladi.

2. Rasmiy IDE:

- Xcode : MacOS uchun Apple tomonidan taqdim etilgan rasmiy dasturlash muhiti. U Swift va Objective-C tillarini qo'llab-quvvatlaydi, ilovalarni yaratish, test qilish va joylashtirish imkonini beradi.

3. Frameworklar va kutubxonalar:

- UIKit : iOS ilovalari uchun foydalanuvchi interfeys elementlarini yaratish uchun ishlatiladi.
- SwiftUI : Declarative yondashuv asosida foydalanuvchi interfeyslarini yaratish uchun yangi framework. Xususiyati shundaki, kodni yozish va natijani ko'rish oson.

Android uchun dasturlash

1. Dasturlash tillari:

- Kotlin : 2017-yilda Google tomonidan Android ilovalari uchun rasmiy til sifatida e'lon qilingan. Kotlin zamonaviy, xavfsiz va sintaktik jihatdan qulay til hisoblanadi.
- Java : Uzoq vaqt davomida Android ilovalarini yaratish uchun asosiy til bo'lib kelgan. Hozir ham ko'p loyihalarda foydalaniladi.

2. Rasmiy IDE:

- Android Studio : Google tomonidan taqdim etilgan rasmiy dasturlash muhiti. Kotlin va Java tillarini qo'llab-quvvatlaydi, ilovalarni yaratish, test qilish va joylashtirish imkonini beradi.

3. Frameworklar va kutubxonalar:

- Jetpack : Android ilovalarini tezroq va osonroq yaratish uchun mo'ljallangan komponentlar to'plami. UI, LiveData, Navigation va boshqa ko'plab komponentlarni o'z ichiga oladi.
- AndroidX : Ilova komponentlari va kutubxonalarining yangi versiyalari va kengaytirilgan imkoniyatlarini taqdim etadi.

Emulyatorlar

1. iOS emulyatorlari:

- Xcode Simulator : Xcode ichida mavjud bo'lgan emulyator bo'lib, turli xil iPhone va iPad modellarini sinab ko'rish imkonini beradi. Bu rasmiy vosita va iOS ilovalarini test qilish uchun juda qulay.

2. Android emulyatorlari:

- Android Emulator : Android Studio ichida mavjud bo'lgan emulyator bo'lib, turli xil Android qurilmalarini sinab ko'rish imkonini beradi. Juda ko'p turli xil qurilmalar va Android versiyalarini qo'llab-quvvatlaydi.
- Genymotion : Bu uchinchi tomon emulyatori bo'lib, turli xil Android qurilmalarini sinab ko'rish uchun keng imkoniyatlar taqdim etadi. Tezligi va moslashuvchanligi bilan mashhur.

Cross-platform dasturlash:

1. Dasturlash tillari va frameworklar:

- Flutter : Google tomonidan ishlab chiqilgan va Dart tilida yozilgan. Bitta kod bazasidan iOS va Android uchun ilovalarni yaratish imkonini beradi.
- React Native : Facebook tomonidan ishlab chiqilgan va JavaScript bilan yozilgan. Bitta kod bazasidan iOS va Android uchun ilovalarni yaratish imkoniyatini taqdim etadi.
- Xamarin : Microsoft tomonidan ishlab chiqilgan va C# tilida yozilgan. Bu platforma ham iOS, ham Android uchun bitta kod bazasidan ilovalarni yaratish imkonini beradi.

2. Foydasi:

- Tezroq rivojlantirish : Bitta kod bazasidan foydalangan holda ikkala platforma uchun ham ilovalarni yaratish va yangilash tezroq va samaraliroq bo'ladi.
- Kam xarajatlar : Rivojlantirish jarayonida resurslarni tejash imkonini beradi, chunki bitta jamoa ikkala platforma uchun ham ilovalarni ishlab chiqishi mumkin.

Mobil ilovalar yaratishda ishlatiladigan tillar va texnologiyalar zamonaviy ehtiyojlarga moslashgan va tezkor rivojlanishni ta'minlaydi. Hozirgi kunda iOS va Android uchun ilovalar yaratish jarayoni yanada qulay va samarali bo'lib bormoqda.

S103. Mobil ilovalarni joylash uchun platformalar: Play Market, Apple Store internet do'konlari

Mobil ilovalarni joylash uchun asosiy platformalar quyidagilar:

Play Market (Google Play Store):

Android qurilmalar uchun mo'ljallangan ilovalarni joylashtirish uchun asosiy do'kon. Google tomonidan boshqarilib, Android operatsion tizimining rasmiy do'koni hisoblanadi. Ilovalarni joylashtirish va tarqatish uchun Google tomonidan qo'yilgan talablar va standartlarga rioya qilish talab etiladi. Ijtimoiy, moliyaviy va boshqa turli xizmatlar ko'rsatadi.

Apple App Store (Apple Store):

iOS, iPadOS va macOS qurilmalari uchun mo'ljallangan ilovalarni joylashtirish uchun asosiy platforma. Apple tomonidan boshqarilib, Apple ekosistemi uchun rasmiy do'kon hisoblanadi. Ilovalarni joylashtirish va tarqatish uchun Apple tomonidan qo'yilgan qattiq talablar va sifat standartlariga rioya qilish talab etiladi. Ijtimoiy, moliyaviy va boshqa xizmatlar ko'rsatadi.

Boshqa internet do'konlari:

Amazon Appstore, Samsung Galaxy Store, Huawei AppGallery kabi boshqa platformalar ham mavjud. Bu do'konlar ma'lum bir brendga yoki ekosistemaga mo'ljallangan bo'ladi. Ular ham o'z talablari va sifat standartlariga ega. Umumiy qilib aytganda, mobil ilovalarni keng auditoriyaga tarqatish uchun Google Play va Apple App Store yetakchi platformalar hisoblanadi. Lekin boshqa do'konlar ham mavjud bo'lib, ularga ham e'tibor berish kerak.

Mobil ilovalarni joylash va tarqatish uchun eng mashhur platformalar Google Play Store va Apple App Store hisoblanadi. Bu platformalar dasturchilar uchun keng imkoniyatlar va foydalanuvchilarga katta tanlovni taqdim etadi. Shuningdek, boshqa internet do'konlari ham mavjud bo'lib, ular ham mobil ilovalarni tarqatishda muhim rol o'ynaydi.

S104. Mobil ilova ishlab chiqish tamoyillari, talablarni o'rganish va texnik topshiriqni shakllantirish, platformani tanlash

Mobil ilova ishlab chiqish jarayonida e'tibor berilishi kerak bo'lgan asosiy tamoyillar va bosqichlar quyidagilar:

Talablarni o'rganish va texnik topshiriqni shakllantirish:

- Mijoz yoki foydalanuvchilar ehtiyojlarini chuqur o'rganish.
- Ilovaning maqsadini, asosiy funksiyalarini, foydalanuvchi guruhlarini va foydalanish senariylari aniqlab olish.
- Texnik topshiriqni shakllantirish: tex. talablar, dizayn, funksionallik, interfeys, UX, xavfsizlik integratsiya va boshqalar.

Platformani tanlash:

- Ilovaning maqsad va vazifalaridan kelib chiqqan holda, qaysi mobile OS (iOS, Android, Windows) uchun ilova yaratish kerakligini belgilash.
- Har bir platformaning afzalliklari, cheklovlari, dasturlash tillari, framework-lari, bozorlar va boshqa jihatlarni tahlil qilish.
- Agar imkon bo'lsa, cross-platform yechimni qo'llash va ikki platform uchun ham ilova yaratish.

Dizayn va UX:

- Foydalanuvchi interfeysi (UI) va foydalanuvchi tajribasi (UX) dizaynini ishlash.
- Mobil qurilmalar xususiyatlarini hisobga olgan holda, intuitivligi va tezkorligiga e'tibor berish.
- Brend, rasm, rang, shrift va boshqa elementlarni uyg'unlashtirish.

Loyiha boshqaruvi va dasturlash:

- Agile/Scrum yondashuvlardan foydalanib, muvaffaqiyatli loyiha boshqarishni tashkil qilish.
- Platforma uchun optimal dasturlash tilini (Swift, Kotlin, React Native, Flutter) va framework-larni tanlash.
- Mobil qurilmalarda optimal ishlash, xavfsizlik, ma'lumotlar xususiyatlari va boshqa texnik talablarni bajarish.

Sinovlar, deployment va qo'llab-quvvatlash:

- Ilova ishlashini sinab ko'rish, bug'larni tuzatish va optimizatsiya qilish.
- Ilovani do'konlarga (Google Play, App Store) joylashtirish va deploy qilish.
- Foydalanuvchilar uchun qo'llab-quvvatlash va yangilanishlarni taqdim etish.

Umumiy qilib aytganda, mobil ilova yaratish jarayonida foydalanuvchi ehtiyojlari, platforma xususiyatlari, dizayn va dasturlash tamoyillari kabi ko'plab jihatlarni hisobga olish kerak.

Mobil ilova ishlab chiqish jarayoni samarali va muvaffaqiyatli bo'lishi uchun muhim tamoyillar, talablarni o'rganish va texnik topshiriqni shakllantirish, hamda platformani tanlash bosqichlarini o'z ichiga oladi. Bu bosqichlar dasturchilarga loyihani to'g'ri rejalashtirish va amalga oshirish imkoniyatini beradi.

Mobil ilova ishlab chiqish tamoyillari

1. Foydalanuvchi tajribasi (UX) :

- Soddalik va intuitivlik : Ilovaning interfeysi oddiy va tushunarli bo'lishi kerak.
- Responsive dizayn : Ilova har xil ekran o'lchamlarida to'g'ri ishlashi kerak.
- Foydalanuvchi oqimi : Foydalanuvchilar ilovadan foydalanish jarayonida qiyinchiliklarga duch kelmasliklari kerak.

2. Natijadorlik va tezlik :

- Yuklanish vaqti : Ilova tez yuklanishi kerak.
- Optimal kod yozish : Dastur kodini optimal va samarali yozish lozim.

3. Xavfsizlik :

- Ma'lumotlarni himoya qilish : Foydalanuvchilarning shaxsiy ma'lumotlarini himoya qilish kerak.
- Xavfsizlik protokollari : Muayyan xavfsizlik protokollariga rioya qilish, masalan, SSL/TLS.

4. Skalabilite va kengayish imkoniyati :

- Kelajakdagi yangilanishlar : Ilovaga yangi funksiyalar qo'shish imkoniyatini ta'minlash.
- Foydalanuvchilar sonining oshishi : Ko'proq foydalanuvchilar uchun ilovaning barqaror ishlashi.

Talablarni o'rganish va texnik topshiriqni shakllantirish

1. Talablarni o'rganish :

- Foydalanuvchilar ehtiyojlari : Ilovani kimlar ishlatadi va ularning ehtiyojlari qanday?
- Bozor tahlili : Bozorda mavjud ilovalarni o'rganish, ularning afzallik va kamchiliklarini aniqlash.
- Raqobatchilarni tahlil qilish : Raqobatchilar ilovalari va ular qanday funksiyalarni taqdim etishini o'rganish.

2. Texnik topshiriqni shakllantirish :

- Loyihaning maqsadi : Ilovaning asosiy maqsadi va vazifalari.
- Funksional talablar : Ilova qanday funksiyalarni bajarishi kerak, masalan, foydalanuvchi ro'yxatdan o'tishi, ma'lumotlarni ko'rish, bildirishnomalar.
- Nofunksional talablar : Natijadorlik, xavfsizlik, skalabilite, interfeys dizayni talablariga mos kelish.
- Texnik cheklovlar : Platforma cheklovlari, texnologik cheklovlar.

Platformani tanlash

1. Maqsadli auditoriya va bozordagi ulush :

- iOS yoki Android : Qaysi platformada maqsadli auditoriya ko'proq? Masalan, AQShda iOS foydalanuvchilari ko'proq bo'lsa, Hindistonda Android foydalanuvchilari ko'proq.
- Platformalarning bozor ulushi : Platformaning global va mahalliy bozordagi ulushi.

2. Ilovaning tabiati va funksionalligi :

- Maxsus funksiyalar : iOS yoki Android uchun maxsus funksiyalar kerakmi? Masalan, ARKit (iOS) yoki ARCore (Android).
- Qo'llab-quvvatlanadigan qurilmalar : Platformaning turli qurilmalarda ishlash imkoniyatlari.

3. Rivojlantirish va qo'llab-quvvatlash :

- Rivojlantirish xarajatlari : iOS va Android ilovalarini rivojlantirish xarajatlarini solishtirish.

- Texnik qo'llab-quvvatlash : Har bir platforma uchun mavjud texnik qo'llab-quvvatlash va hujjatlar.

4. Cross-platform yechimlari :

- Flutter, React Native, Xamarin : Bitta kod bazasidan iOS va Android ilovalarini yaratish imkoniyatini ta'minlaydigan platformalarni ko'rib chiqish.

- Afzallik va kamchiliklari : Cross-platform yondashuvining afzalliklari va kamchiliklarini baholash.

Mobil ilova ishlab chiqish jarayoni talablarni to'g'ri o'rganish va texnik topshiriqni aniq shakllantirishdan boshlanadi. Platformani tanlash esa ilovaning maqsadli auditoriyasi, funksionalligi va rivojlantirish xarajatlariga qarab amalga oshiriladi. Ushbu jarayonlarni to'g'ri rejalashtirish muvaffaqiyatli mobil ilovani yaratishga yordam beradi.

S105. Mobil ilova dizaynini UI/UX dizayn

Mobil ilova dizaynida UI (Foydalanuvchi interfeysi) va UX (Foydalanuvchi tajribasi) dizayn muhim o'rin tutadi. Quyida bu ikki muhim komponent haqida qisqacha ma'lumot berilgan:

UI (Foydalanuvchi interfeysi) dizayn:

- Ilovaning vizual ko'rinishi, interaktiv elementlari va funksionalligini belgilaydi.
- Asosiy komponentlar: navigatsiya, ekran dizayn, tugmalar, rasmlar, shriftlar, ranglar va boshqalar.
- Intuitivligi, osonlik, vizual yaxlitlik va estetikaga e'tibor berish kerak.
- Mobil qurilmalarda optimal ko'rinish, responsivlik va ergonomikka rioya qilish muhim.
- Brend va uslubga moslash, konzistent va tizimli yondashuvni talab qiladi.

UX (Foydalanuvchi tajribasi) dizayn:

- Foydalanuvchining ilovadan foydalanish jarayonini, sezgi va his-tuyg'ularini yaratadi.
- Foydalanish osonligi, sezgirligi, kerakli ma'lumotlarga tezkor kirish, interaktivlik, oqilona navigatsiya kabi jihatlarni o'z ichiga oladi.
- Foydalanuvchi ehtiyojlari, odatlari, maqsadlari va xatti-harakatlarini chuqur tahlil qilish asosida shakllantirish.
- Foydalanuvchi sayyohati, oqimlar, ekran ketma-ketligi, interaktiv elementlar kabi UX komponentlarini loyihalashtirish.
- Funksionallik, estetika va foydalanish osonligining muvozanatini ta'minlash muhim.

UI va UX dizaynini integratsiya qilish:

- UI va UX dizaynini bir-biri bilan uzviy bog'liq holda loyihalash.
- Foydalanuvchi ehtiyojlaridan kelib chiqqan holda, vizual yechimlarni ishlab chiqish.
- UI elementlar UX maqsadlariga xizmat qilishi va foydalanuvchilarga qulay tajribani taqdim etishi kerak.

- Sinov va qayta-qayta tekshirish orqali optimal natijaga erishish.

Natijada, mobil ilova dizaynida UI va UX komponentlarini muvaffaqiyatli integratsiya qilish foydalanuvchilarning ilovadan qoniqishini ta'minlaydi.

Mobil ilova dizayni foydalanuvchi interfeysi (UI) va foydalanuvchi tajribasi (UX) dizaynining asosiy jihatlariga e'tibor qaratadi. Ular ilovaning tashqi ko'rinishi va ishlash qulayligini ta'minlashda muhim rol o'ynaydi.

UI Dizayn

UI dizayni ilovaning vizual ko'rinishi, o'zaro aloqalari va foydalanish qulayligini ta'minlaydi. Quyida UI dizaynining asosiy tamoyillari keltirilgan:

1. Visual hierarxiya :

- Rang va kontrast : Muhim elementlar yorqinroq va katta kontrast bilan ajralib turishi kerak.
- Tipografiya : Matn o'qilishi oson bo'lishi uchun shriftlar va ularning o'lchamlari to'g'ri tanlanishi kerak.

2. Izchillik :

- UI komponentlari : Tugmalar, ikonalar, matn maydonlari va boshqa elementlar izchil bo'lishi kerak.
- Navigatsiya : Navigatsiya elementlari (menyular, tugmalar) bir xil uslubda bo'lishi va doimiy ravishda bir xil joyda joylashishi kerak.

3. Minimalizm :

- Soddalik : Keraksiz elementlarni olib tashlash va asosiy funksiyalarga e'tibor qaratish.
- Bo'sh joylar : Foydalanuvchi ko'zi uchun dam olish joylarini ta'minlash va dizaynni toza saqlash.

4. Responsive dizayn :

- Moslashuvchan interfeyslar : Ilova turli ekran o'lchamlarida to'g'ri ko'rinishi kerak.
- Dynamic layout : Ekran o'lchamlari o'zgarganda UI elementlari moslashishi kerak.

UX Dizayn

UX dizayni foydalanuvchi tajribasini yaxshilashga qaratilgan bo'lib, foydalanuvchilarning ilovadan foydalanish qulayligi va mamnunligini ta'minlaydi. Quyida UX dizaynining asosiy tamoyillari keltirilgan:

1. Foydalanuvchi tadqiqotlari :

- Persona yaratish : Maqsadli auditoriyani aniqlash va ular uchun personajlar yaratish.
- Foydalanuvchi senariylari : Foydalanuvchilarning ilovadan qanday foydalanishini o'rganish va senariylar tuzish.

2. Axborot arxitekturasini :

- Kontent tashkiloti : Kontentni tartibga solish va tuzilmaga kiritish.
- Navigatsiya dizayni : Foydalanuvchilarni ilova ichida osongina harakatlanishlarini ta'minlash.

3. Prototiplash va testlash :

- Wireframing : Ilovaning dastlabki maketlarini yaratish.
- Prototiplash : Ilovaning interaktiv prototiplarini yaratish va foydalanuvchilarda sinab ko'rish.

4. Foydalanuvchi fikr-mulohazalari :

- Testlash : Ilovani beta test qilish va foydalanuvchilar fikr-mulohazalarini yig'ish.
- Iterativ rivojlantirish : Foydalanuvchilar fikr-mulohazalariga asoslanib ilovani doimiy ravishda takomillashtirish.

UI/UX Dizayn Jarayoni

1. Tadqiqot va Tahlil :

- Bozor tadqiqoti : Bozordagi raqobatchilar va ularning ilovalarini o'rganish.

- Foydalanuvchi tadqiqoti : Maqsadli foydalanuvchilarni va ularning ehtiyojlarini aniqlash.

2. Loyihalash va Prototiplash :

- Wireframe yaratish : Ilovaning asosiy tuzilishini qog'oz yoki raqamli vositalar yordamida chizish.

- UI komponentlarini loyihalash : Tugmalar, ikonalar, menyular va boshqa UI elementlarini yaratish.

3. Interaktiv Prototiplash :

- Interaktiv prototiplar yaratish : Foydalanuvchilar ilova bilan qanday aloqada bo'lishlarini simulyatsiya qilish.

- Testlash : Foydalanuvchilarda ilovani sinab ko'rish va fikr-mulohazalarni yig'ish.

4. Rivojlantirish va Ishga tushirish :

- Dizaynni dasturchilarga topshirish : Dizaynni kodga aylantirish uchun dasturchilar bilan hamkorlik qilish.

- Ilovani sinash : Ilovani barcha funksional va nofunksional talablar asosida sinash.

5. Iterativ yaxshilash :

- Fikr-mulohazalar yig'ish : Foydalanuvchilar va tahlil vositalaridan olingan ma'lumotlarni yig'ish.

- Takomillashtirish : Fikr-mulohazalar asosida ilovani yaxshilash va yangilash.

Dizayn Vositalari

1. Wireframing va prototiplash vositalari :

- Sketch : MacOS uchun kuchli dizayn vositasi.
- Figma : Bulutga asoslangan dizayn vositasi bo'lib, hamkorlik uchun ideal.
- Adobe XD : Interaktiv dizayn va prototiplash uchun Adobe vositasi.

2. Grafik dizayn vositalari :

- Adobe Photoshop : Rasmlarni tahrirlash va grafik elementlar yaratish uchun keng qo'llaniladigan vosita.

- Adobe Illustrator : Vektor grafikalar yaratish uchun qulay vosita.

3. Animatsiya va interaktivlik :

- Principle : Mobil ilovalar uchun animatsiya va interaktiv prototiplar yaratish.
- InVision : Dizaynlarni taqdim etish va prototiplarni sinash uchun vosita.

Mobil ilova dizaynida UI va UX dizayn tamoyillariga amal qilish foydalanuvchilar uchun qulay va jozibador ilovalarni yaratishga yordam beradi. Foydalanuvchilarning ehtiyojlarini qondirish va ularning ilovadan foydalanish tajribasini yaxshilash dizayn jarayonining asosiy maqsadidir.

S106. Kross pltaformali dasturlash texnologiyalari: DART => Flutter texnologiyasi, Android Studioni

Kross-platform mobil dasturlash texnologiyalari orasida Flutter texnologiyasi yetakchi o'rinlardan birini egallaydi. Flutter Dart dasturlash tilidan foydalanadi.

DART dasturlash tili

- Google tomonidan yaratilgan, ob'ektga yo'naltirilgan, statik tipli dasturlash tili.
- JavaScript va C# kabi tillarga o'xshash, lekin tezkor ishlashiga moslangan.
- Mobil, web va server-side dasturlar uchun mo'ljallangan.
- Tez va samarali natijalar beradi, statik tipli bo'lgani uchun xatolarni tez aniqlash imkonini beradi.
- Flutter uchun asosiy dasturlash tili sifatida ishlatiladi.

Flutter texnologiyasi

- Google tomonidan yaratilgan kross-platform mobil ilova yaratish uchun framework.
- Dart dasturlash tilidan foydalanadi.
- Ilovalarni original dizayn va unumdorlik bilan yaratishga imkon beradi.
- Bir xil kod bazasi asosida iOS va Android uchun ilova yaratish imkonini beradi.
- Integral dizayn tizimiga ega bo'lib, tez va samarali ilova yaratishga yordam beradi.
- Keng qamrovli widget-lar kutubxonasi, davom etuvchi integrasiya va deploymi qo'llab-quvvatlaydi.
- Mashinali o'rganish, xaritalar, videolar, harakatlar kabi keng funktsionallikka ega.

Android Studio

- Google tomonidan ishlab chiqilgan Android ilovalarini yaratish uchun yetakchi IDE.
- Kotlin va Java dasturlash tillarini qo'llab-quvvatlaydi.
- Android SDK, emulatorlar, debug vositalar, UI designer va boshqa qulayliklar bilan ta'minlangan.
- Loyihalar boshqaruvi, versiyalar, build qilish, deploy va boshqa funktsiyalar bilan qulay ishlash imkonini beradi.
- Flutter ilovalarini ham yaratish va deploy qilish uchun qo'llanilishi mumkin.

Umumiy qilib aytganda, kross-platform mobil ilova yaratishda Flutter va Dart texnologiyalari samarali yechimlarni taqdim etadi, Android Studio esa Android uchun ilovalar yaratish uchun qulay muhit bo'lib xizmat qiladi.

Cross-platform dasturlash texnologiyalari, xususan Dart tilida ishlaydigan Flutter texnologiyasi, mobil ilovalarni yaratishda katta imkoniyatlar va qulayliklar taqdim etadi. Quyida Flutter texnologiyasi, uning asoslari, Android Studio bilan qanday ishlashi haqida batafsil ma'lumot beraman.

Dart va Flutter Texnologiyasi

Dart Tili

- Dart : Google tomonidan ishlab chiqilgan dasturlash tili bo'lib, asosan frontend rivojlantirish uchun mo'ljallangan. Dart sintaksisi JavaScript, Java va C# tillariga o'xshash.
- Asosiy xususiyatlari :
 - Statik tipizatsiya : Kodni yozish jarayonida tiplar aniqlanadi, bu esa kompilyatsiya vaqtida ko'p xatolarni oldini olishga yordam beradi.
 - Kompilyatsiya va interpreterlash : Dart kodini tezkor ishlatish uchun JIT (Just-In-Time) va ishlab chiqarish uchun AOT (Ahead-Of-Time) kompilyatsiya qiladi.

Flutter

- Flutter : Google tomonidan ishlab chiqilgan va 2017-yilda taqdim etilgan open-source UI framework bo'lib, bitta kod bazasidan iOS va Android uchun ilovalarni yaratish imkonini beradi.
- Asosiy xususiyatlari :
 - Bir xil kod bazasi : Bitta kod bazasidan turli platformalar uchun ilovalar yaratish.
 - High-performance : Native kompilyatsiya qilinganligi sababli ilovalar yuqori tezlikda ishlaydi.
 - Rich set of widgets : Flutterda mavjud bo'lgan boy widgetlar to'plami yordamida ilovalarni tez va oson yaratish mumkin.

- Hot Reload : Kodingizdagi o'zgarishlarni tezda ko'rish imkonini beradi, bu esa ishlab chiqish jarayonini sezilarli darajada tezlashtiradi.

Flutter va Android Studio

Android Studio haqida

- Android Studio : Google tomonidan ishlab chiqilgan va Android ilovalarini yaratish uchun asosiy IDE hisoblanadi. Flutter ilovalarini yaratishda ham keng qo'llaniladi.
- Xususiyatlari :
 - IntelliJ-based IDE : IntelliJ Platformasiga asoslangan va kuchli rivojlantirish vositalarini taqdim etadi.
 - Dart va Flutter plaginlari : Flutter ilovalarini yaratish uchun Android Studio ichida Dart va Flutter plaginlarini o'rnatish mumkin.

Flutter ilovalarini Android Studio yordamida yaratish

1. Android Studio o'rnatish :

- Android Studio'ni rasmiy veb-saytdan yuklab oling va o'rnatish.

2. Dart va Flutter plaginlarini o'rnatish :

- Android Studio ichida "Plugins" bo'limiga o'ting va "Flutter" va "Dart" plaginlarini qidirib topib, o'rnatish.
- Plaginlarni o'rnatgandan so'ng, IDE ni qayta yuklash talab qilinishi mumkin.

3. Yangi Flutter loyihasini yaratish :

- Android Studio'ni ishga tushiring va "Start a new Flutter project" tugmasini bosish.
- Loyihaning nomi, saqlash yo'li va boshqa parametrlari bilan formani to'ldiring.
- Dart SDK yo'lini ko'rsating (agar kerak bo'lsa).
- Loyihani yaratish tugagach, IDE sizga Flutter loyihasi uchun tayyor ish muhitini taqdim etadi.

4. Kod yozish va sinash :

- Main.dart faylida asosiy kod yoziladi. Bu fayl ilovaning boshlang'ich nuqtasi hisoblanadi.

- Hot Reload : Koddagi o'zgarishlarni tezda ko'rish uchun "Hot Reload" funksiyasidan foydalaning. Bu jarayon ilovani qayta yuklamasdan o'zgarishlarni ko'rishga imkon beradi.

- Emulator yoki real qurilma : Android Studio ichida emulator yaratib, unda ilovani sinab ko'rishingiz yoki real qurilmani ulab, ilovani sinashingiz mumkin.

5. Ilovani ishga tushirish :

- IDE ichida "Run" tugmasini bosib, ilovani emulator yoki ulangan qurilmada ishga tushiring.

Flutterning afzalliklari va kamchiliklari

Afzalliklari:

- Tez rivojlantirish : Hot Reload va boy widgetlar to'plami rivojlantirish jarayonini tezlashtiradi.

- Platformalararo uyg'unlik : Bitta kod bazasi bilan iOS va Android uchun ilovalar yaratish imkoniyati.

- Natijalardagi uyg'unlik : Flutter ilovalari har ikki platformada ham bir xil ko'rinish va his-tuyg'uga ega bo'ladi.

- Katta hamjamiyat va qo'llab-quvvatlash : Flutterning katta hamjamiyati va keng qo'llab-quvvatlash materiallari mavjud.

Kamchiliklari:

- Katta hajmli ilovalar : Flutter ilovalari boshqa native ilovalarga nisbatan kattaroq hajmda bo'lishi mumkin.

- Platformaga xos API lar : Platformaga xos ba'zi API larni ishlatish qiyinroq bo'lishi mumkin va qo'shimcha pluginlar kerak bo'ladi.

- Nisbatan yangi texnologiya : Flutter hali ham rivojlanishda bo'lib, ba'zi barqarorlik va moslashuvchanlik masalalari mavjud bo'lishi mumkin.

Xulosa

Flutter va Dart ilovalarni yaratish uchun kuchli va moslashuvchan vositalarni taqdim etadi. Android Studio esa bu jarayonda qulay va kuchli rivojlantirish muhitini taqdim etadi. Flutterning cross-platform yondashuvi bilan ilovalarni tez va samarali yaratish mumkin bo'lib, ishlab chiqish jarayonini sezilarli darajada soddalashtiradi.

S107. Flutter: Android Studioda asosiy komponentalar bilan ishlash: Layout, Table, ListView, Grid, List va boshqalar

Flutter'da Android Studioda asosiy komponentalar bilan ishlash quyidagicha amalga oshiriladi:

1. Layout (Joylashtirish):

- Flutter'da layout'lar Widgets orqali yaratiladi.
- Asosiy layout widgetlari: ``Row``, ``Column``, ``Stack``, ``Expanded``, ``Flexible`` va boshqalar.
- Layoutlarni ixtiyoriy darajada ichma-ich joylashtirib, murakkab UI tuzilmalarini yaratish mumkin.
- Layoutlar ``padding``, ``margin``, ``alignment``, ``crossAxisAlignment`` kabi parametrlar orqali moslashtiriladi.

2. Table (Jadval):

- Flutter'da ``Table`` widgeti jadvallar yaratish uchun ishlatiladi.
- ``TableRow`` widgeti orqali qatorlar, ``TableCell`` orqali esa hujjatlar yaratiladi.
- Jadval hujjatlarini kerakli joylashtirish va formatlash uchun ``TableColumnWidth``, ``TableDefaultVerticalAlignment`` kabi parametrlar mavjud.

3. ListView (Ro'yxat):

- ``ListView`` widgeti ro'yxatlarni yaratish uchun ishlatiladi.
- Qisqartirilgan, siljuvchan va dinamik ro'yxatlar yaratish mumkin.
- ``ListView.builder()``, ``ListView.separated()``, ``ListView.custom()`` kabi konstruktorlar bilan turli xil ro'yxatlar yaratish mumkin.
- Ro'yxatlarni yana ``itemBuilder``, ``separatorBuilder``, ``itemCount`` kabi parametrlar orqali moslashtiriladi.

4. Grid (Setka):

- ``GridView`` widgeti setka tuzilmalarini yaratish uchun ishlatiladi.
- ``GridView.count()``, ``GridView.extent()``, ``GridView.builder()`` konstruktorlari orqali setka yaratish mumkin.

- ``crossAxisCount``, ``mainAxisSpacing``, ``crossAxisSpacing`` kabi parametrlar orqali setka ko'rinishini moslashtiriladi.

5. List (Ro'yxat):

- ``List`` class'i Dart'da ro'yxatlar bilan ishlash uchun ishlatiladi.
- Ro'yxatlar turli xil ma'lumotlar turlari bilan to'ldirilishi mumkin.
- ``add()``, ``remove()``, ``insert()``, ``sort()``, ``reversed`` kabi metodi ro'yxatlar ustida amallar bajarish uchun ishlatiladi.

Flutter'da bu kabi asosiy komponentalar Android Studioda mobil ilovalarni yaratishda keng qo'llaniladi. Ularni to'g'ri tanlash va moslash ilovaning samarali va sifatli bo'lishini ta'minlaydi.

Flutterda Android Studio orqali asosiy komponentalar bilan ishlash juda qulay va samarali. Flutter komponentalari (widgets) yordamida mobil ilovalarning interfeyslarini yaratish oson. Quyida Layout, Table, ListView, GridView va boshqa muhim komponentalar bilan qanday ishlash haqida batafsil ma'lumot beriladi.

Android Studioda Flutter loyihasini yaratish

Yangi Flutter loyihasini yaratish

Android Studio-ni ishga tushiring.

"Start a new Flutter project" ni tanlang.

"Flutter Application" ni tanlang va Next tugmasini bosing.

Loyihaning nomi, saqlash yo'li va boshqa parametrlarni kiriting.

Finish tugmasini bosing.

Dart va Flutter pluginlarini o'rnatish

File > Settings > Plugins ga o'ting.

Flutter va Dart pluginlarini qidiring va o'rnatish.

Android Studio-ni qayta yuklang.

Hot Reload: Kodingizdagi o'zgarishlarni darhol ko'rish uchun "Hot Reload" funksiyasidan foydalaning. Bu jarayon ilovani qayta yuklamasdan o'zgarishlarni ko'rishga imkon beradi.

Emulator yoki real qurilmada sinash

Android Studio ichida AVD Manager yordamida emulator yaratib, ishga tushiring.

Real qurilmani USB orqali ulab, "USB debugging" ni yoqing.

Run tugmasini bosib, ilovani ishga tushiring.

Flutterda Android Studio orqali mobil ilovalar yaratish jarayoni juda intuitiv va qulay. Layout, Table, ListView, GridView va boshqa komponentalar bilan ishlash yordamida interfeysni tez va samarali yaratish mumkin. Flutterning "Hot Reload" funksiyasi va Android Studio bilan integratsiyasi rivojlantirish jarayonini sezilarli darajada soddalashtiradi.

S108. Flutter: Android Studioda ma'lumotlar bazasi bilan ishlash, SQLite, MySQL MBBT lar bilan ulanish

Flutter'da Android Studioda ma'lumotlar bazasi bilan ishlash quyidagi yo'nalishlar orqali amalga oshiriladi:

1. SQLite:

- Flutter'da SQLite ma'lumotlar bazasi bilan ishlash uchun ``sqflite`` kutubxonasi tavsiya etiladi.
- ``sqflite`` kutubxonasi orqali SQLite ma'lumotlar bazasini yaratish, ma'lumotlarni saqlash, o'qish, yangilash va o'chirish amallarini amalga oshirish mumkin.
- Ushbu kutubxona ``sqflite`` paketi orqali Flutter loyihalariga integratsiya qilinadi.
- Oddiy va o'rta darajadagi ma'lumotlar bilan ishlash uchun SQLite yaxshi yechim bo'ladi.

2. MySQL MBBT:

- Flutter'da MySQL ma'lumotlar bazasi bilan ishlash uchun ``mysql1`` yoki ``mysql_driver`` kabi kutubxonalardan foydalanish tavsiya etiladi.
- Bu kutubxonalar orqali MySQL serverga ulanish, so'rovlar yuborish, ma'lumotlarni o'qish, yangilash va o'chirish amallarini amalga oshirish mumkin.
- MySQL ma'lumotlar bazasi bilan ishlashda serverdagi ma'lumotlar bilan sinxronlash, masofadan ulanish va murakkab so'rovlar yechimi muhim.

3. Boshqa MBBT:

- Flutter'da boshqa ma'lumotlar bazasi boshqaruv tizimlaridan (MongoDB, PostgreSQL, Firebase Firestore) foydalanish ham mumkin.
- Ular uchun tegishli kutubxonalar (`mongo_dart`, `postgres`, `cloud_firestore`) mavjud.
- Ma'lumotlar hajmi va tuzilma murakkabligi, sinxronlash, masofadan ulanish kabi talablar asosida eng maqbul ma'lumotlar bazasini tanlash mumkin.

Umumiy qilib aytganda, Flutter'da ma'lumotlar bazasi bilan ishlash uchun tegishli kutubxonalar mavjud bo'lib, ular orqali SQLite, MySQL, MongoDB, PostgreSQL kabi MBBT'lar bilan ishlash imkoni mavjud. Ma'lumotlar hajmi, tuzilma murakkabligi va talablar asosida eng maqbul MBBT'ni tanlash muhim.

Flutter ilovalarida ma'lumotlar bazasi bilan ishlash uchun bir nechta vosita va texnologiyalardan foydalanish mumkin. Flutter ilovalari uchun eng keng tarqalgan ma'lumotlar bazalari SQLite va MySQL hisoblanadi. Quyida Flutter ilovalarida Android Studio orqali bu ma'lumotlar bazalari bilan ishlash haqida ma'lumot beriladi.

S109. Mobil ilovalarni ishlab chiqish va ulardan foydalanishda xavfsizlik parametrlari, ruxsatlar, ma'lumotlarni shifrlash

Mobil ilovalarni ishlab chiqish va ulardan foydalanishda xavfsizlik parametrlari, ruxsatlar va ma'lumotlarni shifrlash juda muhim. Quyida bu boradagi asosiy chora-tadbirlar keltirilgan:

1. Xavfsizlik parametrlari:

- SSL/TLS sertifikatlari: Ilovaning serverga ulanishida SSL/TLS sertifikatlari yordamida ma'lumotlarni xavfsiz uzatish.
- App Shielding: Ilovaning kodni va ma'lumotlarini zararli dasturlardan himoya qilish.
- Biometrik autentifikatsiya: Foydalanuvchi identifikatsiyasi uchun TouchID, FaceID, skanerlash kabi usullarni qo'llash.
- Oyin ichida sotib olishlarni himoya qilish: In-app sotib olish amallarini xavfsizligini ta'minlash.

2. Ruxsatlar:

- Foydalanuvchi ruxsatlari: Ilovaning ishlashi uchun kerakli ruxsatlarni (kamera, kontaktlar, GPS va boshqalar) so'rash va ulardan foydalanish.
- Minimal ruxsatlar: Ilova uchun zarur bo'lgan minimal ruxsatlarni so'rash.
- Dinamik ruxsatlar: Foydalanuvchi tomonidan ruxsatlarni yoqish/o'chirish imkoniyatini taqdim etish.

3. Ma'lumotlarni shifrlash:

- Mahalliy ma'lumotlarni shifrlash: Ilovada saqlanadigan ma'lumotlarni, masalan, profil, konfiguratsiya, kesh va hokazo, shifrlash.
- Tarmoq orqali uzatiladigan ma'lumotlarni shifrlash: Serverga yuborilayotgan va serverdan olinadigan ma'lumotlarni SSL/TLS yordamida shifrlash.
- Shaxsiy ma'lumotlarni shifrlash: Foydalanuvchining shaxsiy ma'lumotlarini (kontaktlar, geolokatsiya va boshqalar) himoya qilish.

4. Boshqa xavfsizlik chora-tadbirlari:

- Jailbreak/Rootlangan qurilmalari aniqlash: Agar ilova jailbreak/rootlangan qurilmalarda ishlamasa yaxshi.
- Code Obfuscation: Ilovaning kodini oshkora ko'rinishdan yashirish.
- Xatolarni deb-agginglash: Ilovadagi xatolar haqidagi ma'lumotlarni odamlar ko'rib turolmasligi uchun.
- Yangilanishlar: Ilovaning yangi versiyalarini muntazam ravishda chiqarish va xavfsizlik kamchiliklarini bartaraf etish.

Yuqoridagi chora-tadbirlarni qo'llash mobil ilovalarning xavfsizligini oshiradi, foydalanuvchi ma'lumotlarini himoya qiladi va ilovaning zamonaviy talablarga javob berishini ta'minlaydi.

Mobil ilovalarni ishlab chiqish va ulardan foydalanishda xavfsizlik juda muhim omil hisoblanadi. Quyida ilovalar xavfsizligini ta'minlash uchun asosiy xavfsizlik parametrlari, ruxsatlar, va ma'lumotlarni shifrlash haqida batafsil ma'lumot beraman.

Mobil ilovalarni ishlab chiqish va ulardan foydalanishda xavfsizlikka katta e'tibor qaratish lozim. Ma'lumotlarni shifrlash, ruxsatlarni to'g'ri boshqarish, autentifikatsiya va avtentifikatsiya usullarini qo'llash, va boshqa xavfsizlik amaliyotlarini qo'llash orqali ilovangizni foydalanuvchilar uchun xavfsizroq qilishingiz mumkin. Mobil ilovalarni yaratishda xavfsizlik amaliyotlarini doimiy ravishda yangilab turish va kuzatish lozim.

110. Mobil datchik turlari va ular bilan ishlash

Mobil qurilmalarda qo'llaniladigan asosiy datchik turlari va ular bilan ishlash quyidagicha:

1. Akselerometr

- Qurilmaning tezlanishini, burilishini va harakatini aniqlaydi.
- Buning uchun 3 o'lchovli tezlanishni o'lchaydi (X, Y, Z o'qlari bo'yicha).
- Akselerometr ma'lumotlaridan foydalanib, qurilmaning holatini (portret, lanshaft, chayqalish) aniqlash, pedometr, o'yin boshqaruvi va boshqalar uchun foydalaniladi.

2. Girooskop

- Qurilmaning jismoniy aylanishini, burilishini va inersiyasini aniqlaydi.
- Bunda 3 o'lchovli burilish ma'lumotlari (X, Y, Z o'qlari bo'yicha) qayd etiladi.
- Girooskop ma'lumotlari akselerometr ma'lumotlari bilan birgalikda ishlatiladi, masalan, AR, VR, boshqaruv va hokazolar uchun.

3. Magnit sezgir

- Yer magnit maydonining yo'nalishini aniqlaydi.
- Qurilmaning kompas sifatida ishlashi uchun foydalaniladi.
- Magnit sezgir ma'lumotlari GPS, akselerometr va girooskop ma'lumotlari bilan birgalikda ishlatiladi.

4. GPS

- Qurilmaning joylashuv ma'lumotlarini (kenglik, uzunlik, balandlik) aniqlaydi.
- Navigatsiya, GPS kuzatuv, lokatsiya-asosli xizmatlar uchun foydalaniladi.

5. Yaqinlik sezgiri

- Qurilmaning boshqa jismlar yoki yuzalarga yaqinlashishini aniqlaydi.
- Masalan, qulf ekranini yoqish/o'chirish, ovoz kirish/chiqish, ekran yoqish/o'chirish kabi funksiyalar uchun ishlatiladi.

6. Yorug'lik sezgiri

- Qurilma atrofidagi yoritilganlik darajasini o'lchaydi.

- Masalan, ekran yorug'ligini moslash, qulf ekranini yoqish/o'chirish uchun foydalaniladi.

Bu datchiklardagi ma'lumotlar Flutter'da tegishli platformalar uchun yozilgan kutubxonalar (sensors, geolocator, proximity_sensor va boshqalar) orqali olinadi va ishlatiladi.

Mobil ilovalarda turli xil sensorlar (datchiklar) mavjud bo'lib, ular foydalanuvchiga boy tajriba taqdim etishga yordam beradi. Mobil qurilmalardagi sensorlar quyidagi turlarga bo'linadi:

Mobil datchik turlari

1. Akselerometr : Qurilmaning harakatlanishini o'lchaydi (tezlanish, tebranish, orientatsiya o'zgarishi).
2. Gyroskop : Qurilmaning burchak tezligini o'lchaydi, bu esa qurilmaning orientatsiyasini aniqlashda yordam beradi.
3. Magnetometr : Qurilmaning geomagnit maydonini o'lchaydi, bu esa elektron kompas vazifasini bajarishga yordam beradi.
4. GPS : Qurilmaning geografik joylashuvini aniqlaydi.
5. Yaqinlik datchigi : Qurilmaga yaqinlashgan narsalarni aniqlaydi (masalan, telefonni qulog'ingizga olib borganingizda ekran o'chishi).
6. Nur datchigi : Atrof-muhitning yorug'lik darajasini aniqlaydi.
7. Barometr : Atmosfera bosimini o'lchaydi, bu esa balandlikni aniqlashda yordam beradi.
8. Termometr : Qurilmaning ichki haroratini o'lchaydi.
9. Namlik datchigi : Atrof-muhitning namlik darajasini o'lchaydi.
10. Tovush datchigi : Atrof-muhit tovushini o'lchaydi.

Mobil ilovalarda sensorlardan foydalanish ilovalarning funksional imkoniyatlarini kengaytiradi va foydalanuvchi tajribasini yaxshilaydi. Flutterda akselerometr, gyroskop, GPS kabi sensorlardan foydalanish uchun mos paketlardan foydalanish va kerakli ruxsatlarni so'rash orqali qulay va xavfsiz integratsiya qilish mumkin. Sensorlardan foydalanganda har doim foydalanuvchi maxfiyligini himoya qilishga va kerakli ruxsatlarni to'g'ri boshqarishga e'tibor berish muhimdir.

S111. Flutter: Android Studioda ma'lumotlar bazasi bilan ishlash, SQL boshlang'ich buyruqlari SELECT, INSERT, UPDATE, DELETE ni ishlatish.

Flutter'da Android Studioda ma'lumotlar bazasi bilan ishlash va SQL boshlang'ich buyruqlaridan foydalanish uchun quyidagi bosqichlarda amalga oshirish mumkin:

1. SQLite ma'lumotlar bazasini sozlash :

- Android Studio loyihasidagi `pubspec.yaml` faylida `sqflite` paketi qo'shiladi.
- `lib/database/database_helper.dart` faylida `DatabaseHelper` klassi yaratiladi. Bu klass ma'lumotlar bazasi bilan ishlash uchun zarur funksiyalarni o'z ichiga oladi.

2. Ma'lumotlar bazasi jadvallarini yaratish :

- `DatabaseHelper` klassida `initDatabase()` funksiyasi yaratiladi. Bu funksiya ma'lumotlar bazasi va jadvalni yaratadi.

- Masalan, `users` jadvalini yaratish:

```
```dart
Future<void> initDatabase() async {
 await database.execute("
 CREATE TABLE users (
 id INTEGER PRIMARY KEY AUTOINCREMENT,
 name TEXT,
 email TEXT,
 age INTEGER
)
 ");
}
```
```

3. SQL buyruqlarini bajarish :

- SELECT :

```
```dart
```

```
Future<List<Map<String, dynamic>>> getUsers() async {
 return await database.query('users');
}
...
```

- INSERT :

```
```dart  
  
Future<int> insertUser(Map<String, dynamic> user) async {  
  return await database.insert('users', user);  
}  
...
```

- UPDATE :

```
```dart  

Future<int> updateUser(Map<String, dynamic> user) async {
 return await database.update(
 'users',
 user,
 where: 'id = ?',
 whereArgs: [user['id']],
);
}
...
```

- DELETE :

```
```dart  
  
Future<int> deleteUser(int id) async {  
  return await database.delete(  
    'users',  
    where: 'id = ?',  
  );  
}
```

```
whereArgs: [id],  
);  
}  
...
```

4. Flutter UI'da ma'lumotlar bilan ishlash :

- `DatabaseHelper` klassidagi funksiyalardan foydalanib, Flutter UI'da ma'lumotlar bilan ishlash.

- Masalan, `HomePage` klassida `getUsers()` chaqiriladi va olingan ma'lumotlar ekranga chiqariladi.

Ushbu yondashuv Flutter'da Android Studiodagi SQLite ma'lumotlar bazasi bilan oson va samarali ishlashga imkon beradi. Bunda SQL boshlang'ich buyruqlaridan foydalanish orqali ma'lumotlarni yaratish, yangilash, o'qish va o'chirish amalga oshiriladi.

Flutter ilovalarida lokal ma'lumotlar bazasi (SQLite) bilan ishlash uchun `sqflite` paketi keng qo'llaniladi. Bu paket orqali SQL boshlang'ich buyruqlari bilan ma'lumotlarni qo'shish, o'zgartirish, o'chirish va olish mumkin. Quyidagi misollar orqali SQL boshlang'ich buyruqlaridan foydalanishning qanday amalga oshirilishini ko'rib chiqamiz.

S112. Mobil ilovalarni ishlab chiqishda ruxsatlar, ma'lumotlarni shifrlash.

Mobil ilovalarni ishlab chiqishda ruxsatlar (permissions) va ma'lumotlarni shifrlash (data encryption) juda muhimdir, chunki ular foydalanuvchilarning ma'lumiyati va maxfiyliklarini himoya qilishga yordam beradi. Quyidagi qadamlar orqali ruxsatlarni so'ray olasiz va ma'lumotlarni shifrlashingiz mumkin:

Ruxsatlar (Permissions)

Android uchun

Android ilovalarida ruxsatlarni so'ramoq uchun `AndroidManifest.xml` faylida mos qismni qo'shing. Misol uchun, agar kamera ruxsatini talab qilasiz, quyidagi qismni `AndroidManifest.xml` faylidan o'zgartiring:

```
```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.myapp">
 <uses-permission android:name="android.permission.CAMERA" />
 ...
</manifest>
```
```

Ruxsatlarni so'rab olish uchun Android Studio tahrirlovchida `AndroidManifest.xml` faylini oching va kerakli ruxsatlarni qo'shing.

iOS uchun

iOS ilovalarida ruxsatlarni so'ramoq uchun `Info.plist` faylga kerakli ruxsatlarni qo'shing. Misol uchun, agar foydalanuvchining joylashuvini aniqlashga ruxsat so'rash kerak bo'lsa, quyidagi qismni `Info.plist` faylga qo'shing:

```
```xml
<key>NSLocationWhenInUseUsageDescription</key>
<string>May I use your location?</string>
```
```

Ma'lumotlarni shifrlash (Data Encryption)

Mobil ilovalarda foydalanuvchilar ma'lumotlarini shifrlash juda muhimdir. Shifrlashni o'zi foydalanuvchilar ma'lumiyati va maxfiyliklarini himoya qiladi.

Ma'lumotlarni shifrlash uchun kerakli usullar:

1. SharedPreferences : Android va iOS ilovalarida mos foydalanuvchi sozlamalarni saqlash uchun ishlatiladi.
2. sqflite : Lokal ma'lumotlar bazasini ishlatishda qo'llaniladi. Sqflite paketi ma'lumotlar bazasidagi ma'lumotlarni shifrlash uchun imkoniyat beradi.
3. FlutterSecureStorage : Bu paket foydalanuvchining maxfiy ma'lumotlarini shifrlash uchun ishlatiladi.
4. Encryption package : Bu paket bilan ma'lumotlarni shifrlash imkoniyatiga ega bo'lasiz.

Ma'lumotlarni shifrlashni amalga oshirish uchun mos paketni tanlang va foydalaning. Raqamli ma'lumotlarni shifrlash uchun kriptografiya algoritmlaridan, masalan AES (Advanced Encryption Standard) dan foydalanishingiz mumkin.

Bu usullar foydalanuvchilar ma'lumotlarini shifrlash va maxfiylikni ta'minlashda yordam beradi va amaliyotni samarali va ishonchli qilishga yordam beradi.

Mobil ilovalarni ishlab chiqishda ruxsatlar va ma'lumotlarni shifrlash juda muhim masalalardir. Quyida bu boradagi asosiy yondashuvlarni ko'rib chiqamiz:

1. Ruxsatlar (Permissions) :

- Mobil ilova ishlashi uchun zarur bo'lgan ruxsatlarni olish juda muhim.
- Ilova foydalanuvchilarga to'g'ri va aniq tushuntirilishi kerak, ilova qanday ruxsatlar so'rayotgani va ulardan qanday foydalanilishi to'g'risida.
- Android va iOS uchun ruxsatlar turlicha boshqariladi:
 - Android : `AndroidManifest.xml` faylida ruxsatlar ro'yxati ko'rsatiladi. Ilova ishga tushishida ushbu ruxsatlar talab qilinadi.
 - iOS : Ilova ishga tushishida foydalanuvchidan alohida-alohida ruxsatlar so'raladi (masalan, kamera, lokatsiya va boshqalar).

2. Ma'lumotlarni shifrlash (Data Encryption) :

- Mobil ilovalar orqali uzatiladigan va saqlanadigan ma'lumotlar (foydalanuvchi ma'lumotlari, API kalitlari, maxfiy ma'lumotlar va h.k.) maxfiy bo'lishi kerak.
- Buni ta'minlash uchun ma'lumotlarni shifrlash muhim.
- Android : `Android Keystore` tizimidan foydalanish mumkin. Bu tizim ma'lumotlarni shifrlash va raqamli imzo qo'yish uchun ishlatiladi.
- iOS : `Keychain` xizmati ma'lumotlarni shifrlash uchun ishlatiladi. Bu xizmat maxfiy ma'lumotlarni saqlash va ulardan foydalanishni osonlashtiradi.
- Shuningdek, Flutter'da `flutter_secure_storage` paketi orqali ham ma'lumotlarni shifrlash imkoniyati mavjud.

3. Boshqa xavfsizlik chora-tadbirlari :

- Ilova kodini obfuskatsiya qilish (tushunilishini qiyinlashtirish) - bu xatolarni kamaytirishga va ularni qidirish qiyinlashtirishga yordam beradi.
- API-lar va xizmatlar bilan aloqani faqat HTTPS orqali amalga oshirish.
- Mobil qurilmalar uchun maxsus xavfsizlik siyosatlarini joriy qilish (masalan, jismoniy kirish cheklovlari, trusted zone va boshqalar).
- Foydalanuvchi ma'lumotlarini minimal darajada saqlash, zarur bo'lmagan ma'lumotlarni saqlashdan qochish.

Yuqoridagi yondashuvlar mobil ilovalar uchun xavfsizlik va maxfiylikni ta'minlashda muhim ahamiyatga ega. Har bir loyiha uchun xavfsizlik talablari turli bo'lgani sababli, ushbu masalalar doimiy ravishda ko'rib chiqilishi va dolzarb qoldirilishi kerak.

S113. Processlar (Activities)

Mobil ilovalar uchun processlar (Activities) juda muhim tushunchadir. Quyida Flutter'da processlar (Activities) bilan ishlash bo'yicha asosiy yondashuvlarni ko'rib chiqamiz:

1. Processlar tushunchasi :

- Processlar - bu mobil ilovaning asosiy bloki. Har bir screen/sahifa bir process hisoblanadi.
- Processlar foydalanuvchining o'zaro aloqasini boshqaradi, ma'lumotlarni ko'rsatadi va qabul qiladi.
- Har bir process o'zining hayot davri (lifecycle) va o'zaro bog'liq holatlaridan iborat.

2. Processlarni yaratish :

- Flutter'da processlarni ``Widget`` sinfidan meros olgan ``StatefulWidget`` yoki ``StatelessWidget`` sinflaridan yaratish mumkin.

3. Processlar o'rtasida navigatsiya :

- Processlar o'rtasida o'tish uchun ``Navigator`` sinfi ishlatiladi.
- Masalan, ``HomePage`` dan ``DetailsPage`` ga o'tish:

```
```dart
```

```
Navigator.push(
```

```
 context,
```

```
 MaterialPageRoute(builder: (context) => DetailsPage()),
```

```
);
```

```
```
```

- Ortga qaytish uchun ``Navigator.pop(context)`` ishlatiladi.

4. Hayot davri (Lifecycle) boshqaruvi :

- Processlar o'z hayot davri (lifecycle) holatlarini boshqaradi: ``initState()``, ``build()``, ``dispose()`` kabi.
- Bu holatlar orqali process yaratish, yangilash, yo'q qilish kabi operatsiyalarni amalga oshirish mumkin.

- Masalan, ``initState()`` methodida ma'lumotlarni yuklash, ``dispose()`` methodida esa resurslarni to'xtatish mumkin.

5. Parametrlar uzatish :

- Processlarga parametrlar uzatish orqali ma'lumotlarni uzatish mumkin.

- Masalan, ``DetailsPage`` ga ``userId`` parametrini uzatish:

```
```dart
Navigator.push(
 context,
 MaterialPageRoute(builder: (context) => DetailsPage(userId: 123)),
);
```
```

- Keyin ``DetailsPage`` classida ``userId`` ni olish:

```
```dart
class DetailsPage extends StatefulWidget {
 final int userId;

 DetailsPage({required this.userId});

 @override
 _DetailsPageState createState() => _DetailsPageState();
}
```
```

Yuqoridagi yondashuvlar Flutter'da processlar (Activities) bilan ishlashning asosiy prinsiplarini ifoda etadi. Bu tushunchalarni chuqur o'rganish va amalda qo'llash mobil ilovalar ishlab chiqishda muhim ahamiyatga ega.

"Processlar" Android platformasidagi asosiy kontseptlardan biridir. Android operatsion tizimi har bir ilova va ilova komponentlari uchun alohida jarayonlar (processlar) yaratadi. Bu jarayonlar alohida joylashadi va faol holatda bo'ladi. Processlar ilovalarning ishlashini boshqarish uchun qo'llaniladi va ularning hayot doirasi Android tizimi tomonidan avtomatik ravishda boshqariladi. Android

ilovachilari biron bir ilovada bir nechta foydalanuvchilar interfeysi (UI) yaratish uchun qo'llaniladigan odatiy jarayonlardir. Android platformasida har bir ilova alohida ishlovchi bir yoki bir nechta jarayon (process) ichida ishlaydi.

Processlar Android ilovasining quyidagi jarayonlari yoki komponentlarini o'z ichiga oladi:

1. **MainActivity** : Bosh sahifa yoki ilova boshlanishi bilan bog'liq asosiy jarayon.
2. **Service** : Foydalanuvchidan ko'rib chiqilmaydigan, oz ichiga alohida yoki asosiy jarayonni (masalan, musiqa ijro etish, fayl yuklash, internetga ulanish) amalga oshiradigan komponent.
3. **BroadcastReceiver** : Qabul qilinayotgan systema xabarlariga javob beradigan komponent.
4. **ContentProvider** : Ma'lumotlarga yuklash, saqlash va ulashish uchun ma'lumotlar bazasini boshqaruvchi komponent.

Processlar uchun Android tizimi quyidagi tizim manbalarni o'z ichiga oladi:

- **Activity Manager** : Har bir jarayon (process)ni boshqaradi va aktiviti (activity)lar o'rtasidagi o'zaro muloqotni boshqaradi.
- **Package Manager** : Ilovani boshqarish, yuklash, o'chirish va ta'rifi uchun mas'ul.
- **Window Manager** : Interfeysni boshqaradi va aktivitilar (activities) uchun oynalarni yaratadi.
- **Content Providers** : Ma'lumotlarni almashish uchun API ni boshqaradi.
- **Resource Manager** : Ilovaga xizmat qilish uchun ma'lumotlarni boshqaradi.

Har bir Android ilovasi alohida jarayonda (process) ishlaydi, shuning uchun foydalanuvchi bir ilova yopib qo'yinganida, u barcha uchun bo'lgan resurslar bilan birgalikda o'chadi. Yangi aktiviti ochilganda, yangi jarayon yaratiladi va aktiviti uni ishga tushiriladi. Yangi jarayon yaratilishi uchun operatsion tizimi foydalanuvchi mavjud jarayondan foydalanadi. Jarayonlar orqali ilovachilar o'zaro o'zlarining ma'lumotlarini almashishadi.

114. Voqealar (intents)

"Voqea" (Intent) Android platformasidagi muhim kontseptlardan biridir. Bu, ikkita ilova yoki ilova komponenti o'rtasida o'zaro muloqotni boshlash uchun ishlatiladi. Voqea ilova komponentlarining biridan boshqariladi va boshqa komponentni amalga oshirish uchun ma'lumotlarni almashishga yordam beradi.

Androidda voqea quyidagi maqsadlarni bajarish uchun ishlatiladi:

1. Komponentlar orasidagi muloqot : Bir ilova yoki komponent boshqasini chaqirish va uni boshqarish, masalan, bir aktiviti boshqariladigan komponentni boshqarish.
2. Ma'lumot almashish : Ma'lumotlarni boshqa ilova yoki komponentga o'tkazish.
3. Platforma xizmatlariga so'rov yuborish : Masalan, e-pochta yoki kartaga qo'ng'iroq qilish uchun e-pochtani ochish.
4. Broadcasting : Boshqa ilovalarga xabar yuborish va boshqalar tomonidan o'rnatilgan xabarlar qabul qilish.

Voqea Androidda quyidagi ikkita turda bo'ladi:

1. Explicit Voqea : Bunda bir ilova boshqasining nomi va uning paket nomi aniqlanadi, shuningdek, boshqa bir ilova komponenti yoki faqat bir ilova ichidagi biron bir komponent (masalan, aktiviti) chaqiriladi.
2. Implicit Voqea : Bunda ilova, maqsad va qaror birligini aniqlayadi va Android tizimi uchun qanday ilova yoki komponentlar ustida ish bajarishni aniqlab beradi.

Voqea Android ilovasining komponentlarini birlashtirishning muhim vositasi hisoblanadi. Uning yordamida ilova komponentlari o'rtasida muloqot boshqariladi va ma'lumotlar almashiladi. Bu esa foydalanuvchilarga ilovalar orasida muloqot boshlash va komponentlar orasida muloqotni tashkil etishga imkon beradi.

Yuqoridagi yondashuvlar Flutter'da voqealar (intents) bilan ishlashning asosiy prinsiplarini ifodalaydi. Voqealar orqali ilovalar o'rtasida ma'lumotlar almashinishini ta'minlash, ilovalar funktsionalligini kengaytirish va ularning o'zaro integratsiyasini yaxshilash mumkin.

115. Fragmentlardan foydalanish

Flutter'da fragmentlardan foydalanish juda muhim tushunchadir. Quyida fragmentlar bilan ishlash bo'yicha asosiy yondashuvlarni ko'rib chiqamiz:

1. Fragmentlar nima?

- Fragmentlar - bu processlar (Activities) ichidagi alohida UI komponentlari.
- Fragmentlar mustaqil hayot davrini (lifecycle) boshqaradi va mavjud processlar (Activities) ichida qayta ishlatilishi mumkin.
- Fragmentlar ilovaning modulyarligi va qayta ishlatilishini ta'minlaydi.

2. Fragmentlarni yaratish :

- Flutter'da `StatefulWidget` yoki `StatelessWidget` sinflaridan meros olingan `Fragment` sinfi yaratish mumkin.
- Masalan, `ChatFragment` fragmentini quyidagicha yaratish mumkin:

3. Fragmentlarni processga (Activity) kiritish :

- Fragmentlarni processlar (Activities) ichida `FragmentManager` widget'i orqali kiritish mumkin.
- Masalan, `HomePage` process (Activity) ichida `ChatFragment` ni qo'shish:

4. Fragmentlar o'rtasida o'zaro aloqa :

- Fragmentlarni bir-biri bilan aloqalashtirib, ma'lumotlar almashinishini ta'minlash mumkin.
- Masalan, `ChatFragment` va `UserProfileFragment` fragmentlari o'rtasidagi o'zaro aloqani quyidagicha tashkil etish mumkin:

Yuqoridagi yondashuvlar Flutter'da fragmentlar bilan ishlashning asosiy prinsiplarini ifoda etadi. Fragmentlarni to'g'ri qo'llash ilovaning modulyarligi, qayta ishlatilishi va funktsionalligini yaxshilashga yordam beradi.

Fragmentlar Android ilovalarida UI-ni qism-qism yoki modul ko'rinishida tuzishga yordam beruvchi komponentlardir. Fragmentlar biron bir aktiviti ichidagi biron bir UI hissasini ifodalaydi. Fragmentlar boshqa fragmentlar yoki aktivitilar ichidan

chaqirilishi mumkin va o'zlarida o'zaro muloqotlar o'tkazishlari mumkin. Bu, Android ilovalarida tashqi ekranlarga o'tish, foydalanuvchi interfeysini modullar bo'yicha tuzish va kichik komponentlarni boshqa komponentlarga qo'shish uchun keng foydalaniladi.

Fragmentlar quyidagi vazifalarni bajarish uchun ishlatiladi:

1. Aktivitini bozorda qismlarga ajratish : Boshqa bir aktivitida ko'p qismlar bo'lishi kerak bo'lsa, har bir qismni alohida fragment sifatida ta'rifi mumkin.
2. Qo'shimcha foydalanuvchi interfeysi funksiyalarni qo'shish : Fragmentlar biron bir foydalanuvchi interfeysi qismi sifatida ishlaydi, masalan, tizim paneli, dialog oynasi, yoki shu kabi.
3. Boshqa fragmentlarga muloqot yuborish : Bir fragment boshqa fragmentlarga yoki aktivitilarga muloqot yuborishi mumkin, bu esa ilovada modulyarlik va qulaylikni oshiradi.
4. Boshqa fragmentlarga ma'lumotlarni o'tkazish : Fragmentlar o'z ichiga ma'lumotlar almashishi mumkin va bu ma'lumotlarni boshqa fragmentlarga yoki aktivitilarga o'tkazishi mumkin.

Fragmentlar Android ilovasining har bir versiyasida mavjud bo'lib, ularga ishlovchi funksional qo'shimcha kutubxonalar mavjud. Fragmentlar kerakli UI-ni tuzish, unga ishlovchi qismni yaratish, ma'lumotlarni boshqa fragmentlarga o'tkazish va boshqa fragmentlarga muloqot yuborish kabi amallarni bajarish uchun moslashtirilgan.

Fragmentlar, Android ilovachilari uchun ko'p qulayliklar ta'minlashi bilan birga, ko'p tomonlama qurilish va modulyar kodni ta'minlashda ham yordam beradi. Bu esa ilovani boshqarish va to'liq korxona interfeyslarini yaratishda yordam beradi.

116. Kontent etkazib beruvchilardan foydalanish

Flutter'da kontent etkazib beruvchilardan (Content Providers) foydalanish juda muhim tushunchadir. Quyida kontent etkazib beruvchilar bilan ishlash bo'yicha asosiy yondashuvlarni ko'rib chiqamiz:

1. Kontent etkazib beruvchilar nima?

- Kontent etkazib beruvchilar - bu boshqa ilovalar yoki komponentlar bilan ma'lumotlarni ulashishga imkon beruvchi vositadir.
- Kontent etkazib beruvchilar ma'lumotlar ombori vazifasini bajaradi va ulardan foydalanish uchun standart API ta'minlaydi.
- Kontent etkazib beruvchilar ilovalar o'rtasida ma'lumotlar xavfsizligini ta'minlaydi.

2. Kontent etkazib beruvchilarni yaratish :

- Flutter'da kontent etkazib beruvchilarni yaratish uchun ``ContentProvider`` sinfi ishlatiladi.
- Masalan, kontaktlarni boshqaruvchi ``ContactsContentProvider`` quyidagicha yaratilishi mumkin:

3. Kontent etkazib beruvchilardan foydalanish :

- Kontent etkazib beruvchilardan foydalanish uchun ``ContentProviderClient`` sinfi ishlatiladi.
- Masalan, ``ContactsContentProvider`` dan kontaktlarni olish:

4. Kontent etkazib beruvchilarning afzalliklari :

- Kontent etkazib beruvchilar ilovalar o'rtasida ma'lumotlar almashinuvini ta'minlaydi.
- Kontent etkazib beruvchilar ma'lumotlar xavfsizligini ta'minlaydi.
- Kontent etkazib beruvchilar ilovalarning modulyarligi va qayta ishlatilishini yaxshilaydi.

Yuqoridagi yondashuvlar Flutter'da kontent etkazib beruvchilar bilan ishlashning asosiy prinsiplarini ifoda etadi. Kontent etkazib beruvchilarni to'g'ri qo'llash ilovaning modulyarligi, xavfsizligi va ma'lumotlar almashinish qobiliyatini yaxshilashga yordam beradi.

Kontent etkazib beruvchilardan (Content Providers) foydalanish, Android ilovasidagi ma'lumotlar bazasiga yoki ma'lumotlarga o'qish va yozish uchun moslashtirilgan bir interfeysni yaratishga imkon beradi. Ular ko'pincha ma'lumotlar bazasiga, fayllarga, multimediyalarga yoki boshqa ilovalar bilan almashish uchun foydalaniladi. Content Providerlar, Android platformasidagi ma'lumotlar bo'limini ifodalaydi va uning ma'lumotlari boshqa ilovalarga taqdim etish va ularga murojaat qilishga imkon beradi.

Content Providerlardan foydalanish quyidagi muhim vazifalarni bajaradi:

1. Ma'lumotlarni almashish : Content Providerlar orqali boshqa ilovalarning ma'lumotlar bazasidan ma'lumotlarni olish mumkin.
2. Ma'lumotlarni saqlash : Content Providerlar yordamida ilovalar ma'lumotlarni boshqa ilovalarning foydalanishiga taqdim etish uchun o'z ma'lumotlar bazasida ma'lumotlarni saqlayadi.
3. Ma'lumotlarga yozish : Ma'lumotlarni o'zgartirish, yangilash va o'chirish uchun Content Providerlar orqali ma'lumotlarga yozish mumkin.
4. Ma'lumotlar almashini himoyalash : Content Providerlar ma'lumotlarga foydalanish uchun ma'lumotlar bazasini himoyalash va ma'lumotlarga faqat kerakli ruxsatlarga ega bo'lishi mumkin.

Content Providerlar quyidagi imkoniyatlarni taqdim etadi:

- SQL ma'lumotlar bazasiga o'qish va yozish.
- Fayllarga o'qish va yozish.
- Multimediyalarga (rasmlar, musiqalar, videolar) o'qish va yozish.
- Boshqa ilovalarning ma'lumotlar bazasiga o'qish va yozish.

Content Providerlardan foydalanish quyidagi jarayonni o'z ichiga oladi:

1. Content Provider obyekti yaratish.
2. Content Resolver obyekti yaratish.
3. Content Resolver yordamida so'rov yuborish (ma'lumotlarni olish, yozish, o'zgartirish va o'chirish).
4. Content Provider javobini qabul qilish.

Android platformasida uchta eng keng qo'llaniladigan Content Provider mavjud:

1. Contacts Provider : Telefon kontaktlariga o'qish va yozish imkoniyatini taqdim etadi.
2. MediaStore Provider : Multimedia fayllarga o'qish va yozish imkoniyatini taqdim etadi.
3. Calendar Provider : Kalendar ma'lumotlariga o'qish va yozish imkoniyatini taqdim etadi.

Shuningdek, foydalanuvchi o'zining ma'lumotlar bazasiga (masalan, mobil ilovasi ma'lumotlar bazasiga) o'qish va yozish uchun shaxsiy Content Providerlarni yaratish mumkin. Bu, foydalanuvchilarga maxfiylikni ta'minlash uchun kerakli ruxsatlarni so'radi.

116. Kontent etkazib beruvchilardan foydalanish

Flutter'da kontent etkazib beruvchilardan (Content Providers) foydalanish juda muhim tushunchadir. Quyida kontent etkazib beruvchilar bilan ishlash bo'yicha asosiy yondashuvlarni ko'rib chiqamiz:

1. Kontent etkazib beruvchilar nima?

- Kontent etkazib beruvchilar - bu boshqa ilovalar yoki komponentlar bilan ma'lumotlarni ulashishga imkon beruvchi vositadir.
- Kontent etkazib beruvchilar ma'lumotlar ombori vazifasini bajaradi va ulardan foydalanish uchun standart API ta'minlaydi.
- Kontent etkazib beruvchilar ilovalar o'rtasida ma'lumotlar xavfsizligini ta'minlaydi.

2. Kontent etkazib beruvchilarni yaratish :

- Flutter'da kontent etkazib beruvchilarni yaratish uchun `ContentProvider` sinfi ishlatiladi.

3. Kontent etkazib beruvchilardan foydalanish :

- Kontent etkazib beruvchilardan foydalanish uchun `ContentProviderClient` sinfi ishlatiladi.
- Masalan, `ContactsContentProvider` dan kontaktlarni olish:

```
```dart
final contactsProvider = ContentProviderClient(ContactsContentProvider());
final contacts = await contactsProvider.getContacts();
```
```

4. Kontent etkazib beruvchilarning afzalliklari :

- Kontent etkazib beruvchilar ilovalar o'rtasida ma'lumotlar almashinuvini ta'minlaydi.
- Kontent etkazib beruvchilar ma'lumotlar xavfsizligini ta'minlaydi.
- Kontent etkazib beruvchilar ilovalarning modulyarligi va qayta ishlatilishini yaxshilaydi.

Yuqoridagi yondashuvlar Flutter'da kontent etkazib beruvchilar bilan ishlashning asosiy prinsiplarini ifoda etadi. Kontent etkazib beruvchilarni to'g'ri qo'llash ilovaning modulyarligi, xavfsizligi va ma'lumotlar almashinish qobiliyatini yaxshilashga yordam beradi.

Kontent etkazib beruvchilardan (Content Providers) foydalanish, Android ilovasidagi ma'lumotlar bazasiga yoki ma'lumotlarga o'qish va yozish uchun moslashtirilgan bir interfeysni yaratishga imkon beradi. Ular ko'pincha ma'lumotlar bazasiga, fayllarga, multimediyalarga yoki boshqa ilovalar bilan almashish uchun foydalaniladi. Content Providerlar, Android platformasidagi ma'lumotlar bo'limini ifodalaydi va uning ma'lumotlari boshqa ilovalarga taqdim etish va ularga murojaat qilishga imkon beradi.

Content Providerlardan foydalanish quyidagi muhim vazifalarni bajaradi:

1. Ma'lumotlarni almashish : Content Providerlar orqali boshqa ilovalarning ma'lumotlar bazasidan ma'lumotlarni olish mumkin.
2. Ma'lumotlarni saqlash : Content Providerlar yordamida ilovalar ma'lumotlarni boshqa ilovalarning foydalanishiga taqdim etish uchun o'z ma'lumotlar bazasida ma'lumotlarni saqlayadi.
3. Ma'lumotlarga yozish : Ma'lumotlarni o'zgartirish, yangilash va o'chirish uchun Content Providerlar orqali ma'lumotlarga yozish mumkin.
4. Ma'lumotlar almashini himoyalash : Content Providerlar ma'lumotlarga foydalanish uchun ma'lumotlar bazasini himoyalash va ma'lumotlarga faqat kerakli ruxsatlarga ega bo'lishi mumkin.

Content Providerlar quyidagi imkoniyatlarni taqdim etadi:

- SQL ma'lumotlar bazasiga o'qish va yozish.
- Fayllarga o'qish va yozish.
- Multimediyalarga (rasmlar, musiqalar, videolar) o'qish va yozish.
- Boshqa ilovalarning ma'lumotlar bazasiga o'qish va yozish.

Content Providerlardan foydalanish quyidagi jarayonni o'z ichiga oladi:

1. Content Provider obykti yaratish.
2. Content Resolver obykti yaratish.
3. Content Resolver yordamida so'rov yuborish (ma'lumotlarni olish, yozish, o'zgartirish va o'chirish).
4. Content Provider javobini qabul qilish.

Android platformasida uchta eng keng qo'llaniladigan Content Provider mavjud:

1. Contacts Provider : Telefon kontaktlariga o'qish va yozish imkoniyatini taqdim etadi.
2. MediaStore Provider : Multimedia fayllarga o'qish va yozish imkoniyatini taqdim etadi.
3. Calendar Provider : Kalendar ma'lumotlariga o'qish va yozish imkoniyatini taqdim etadi.

Shuningdek, foydalanuvchi o'zining ma'lumotlar bazasiga (masalan, mobil ilovasi ma'lumotlar bazasiga) o'qish va yozish uchun shaxsiy Content Providerlarni yaratish mumkin. Bu, foydalanuvchilarga maxfiylikni ta'minlash uchun kerakli ruxsatlarni so'radi.

117. Interfeys yaratishda qo'shimcha komponentlardan foydalanish

Flutter'da interfeys yaratishda qo'shimcha komponentlardan foydalanish juda muhim tushunchadir. Quyida interfeys yaratishda qo'shimcha komponentlar bilan ishlash bo'yicha asosiy yondashuvlarni ko'rib chiqamiz:

1. Qo'shimcha komponentlar nima?

- Qo'shimcha komponentlar - bu standart Flutter widget'laridan tashqari, ilovaning interfeysi va funkcionalligini kengaytirish uchun ishlatiladi.

- Qo'shimcha komponentlar Flutter ekotizimidagi keng qamrovli kutubxonalar tomonidan taqdim etiladi.

- Qo'shimcha komponentlar ilovalarni yanada interaktiv, estetik va funksional qilishga yordam beradi.

2. Qo'shimcha komponentlarni qo'llash :

- Flutter'da qo'shimcha komponentlarni qo'llash uchun ularni pubspec.yaml faylida bog'lash va keyin kodda import qilish lozim.

- Masalan, `flutter_svg` kutubxonasidan `SvgPicture` widgetini qo'llash:

```
```dart
```

```
import 'package:flutter_svg/flutter_svg.dart';
```

```
Widget build(BuildContext context) {
 return SvgPicture.asset('assets/my_icon.svg');
}
```
```

3. Qo'shimcha komponentlar misollar :

- UI komponentlar : `smooth_star_rating`, `carousel_slider`, `expansionpanel_list`
- Animatsiya komponentlar : `lottie`, `flare_flutter`
- Chartlar va grafika : `charts_flutter`, `syncfusion_flutter_charts`

- Interaktiv komponentlar : `flutter_typeahead`, `flutter_slidable`

4. Qo'shimcha komponentlarning afzalliklari :

- Interfeys yaratish jarayonini tezlashtiradi va yanada mukammal qiladi.
- Ilovani yanada interaktiv, estetik va funksional qiladi.
- Ilovaning modulyarligi va qayta ishlatiluvchanligini yaxshilaydi.
- Kodning qayta ishlatilishini oshiradi va ishlab chiqish jarayonini tezlashtiradi.

Yuqoridagi yondashuvlar Flutter'da interfeys yaratishda qo'shimcha komponentlar bilan ishlashning asosiy prinsiplarini ifoda etadi. Qo'shimcha komponentlardan to'g'ri foydalanish ilovaning interfeysi, funksionalligi va estetikasini yaxshilashga yordam beradi.

Interfeys yaratishda qo'shimcha komponentlardan foydalanish Android ilovalarida foydalanuvchi interfeysini (UI) yaratish va taqdim etishda keng qo'llaniladi. Bu qo'shimcha komponentlar, Android ilovalari ichidagi foydalanuvchi interfeysini bozorda, qulay va foydalanishga mo'ljallangan qilishga yordam beradi. Quyidagi qo'shimcha komponentlardan ba'zilari:

1. AlertDialog : Xabarnoma oynasi, foydalanuvchi bilan muloqot oynasi yaratish uchun ishlatiladi.
2. Toast : Kichik oynani boshqarish orqali foydalanuvchiga qisqa xabar bermoqchi bo'lgan voqealarni bildirish.
3. PopupMenu : Boshqa komponentni bosganda chiqadigan ikkilik yoki ko'prik menyu.
4. DatePickerDialog va TimePickerDialog : Sanani va vaqt tanlash uchun oynalar.
5. ProgressDialog : Amalni kutilmoqda xabarini yuklash uchun oyna.
6. Spinner : Tanlash uchun ro'yxatli yoki raqamlanmagan variantlar roy'xati.
7. Seekbar : Qiymatni sozlash uchun ko'prikli oyna.

Bu qo'shimcha komponentlarni ishlatib, Android ilovalarida ko'p qismlarga bo'linadigan va foydalanuvchilarga yaxshi foydalanish tajribasini ta'minlash mumkin. Shu bilan birga, bu komponentlar sizga ko'p oqimlarda foydalanishni osonlashtiradigan yaxshi dizaynlar taqdim etadi.

Masalan, agar siz foydalanuvchining tanlagan variantlardan birini tanlashini talab qilsangiz, ro'yxatni oynalarga yoki Spinner komponentiga joylashtirishingiz mumkin. Bunday qo'shimcha komponentlar foydalanuvchining interfeys bilan qiziqtirishini oshiradi va ilovaning amalini qulayroq qiladi.

Qo'shimcha komponentlar Androidda muhim joy tutadi, chunki ular foydalanuvchining ilova bilan qiziqtirishini oshiradi va ularga foydalanishni osonlashtiradi. Shuningdek, ular interfeysni to'liq tuzish va interaktivlikni oshirishda yordam beradi.

S118. Firebase ma'lumotlar bazasini ilovaga ulash va sozlash

Firebase ma'lumotlar bazasini Flutter ilovasi bilan integratsiya qilish va sozlash juda muhim masaladir. Quyida Firebase ma'lumotlar bazasini Flutter ilovasi bilan ulash va sozlash bo'yicha asosiy yondashuvlarni ko'rib chiqamiz:

1. Firebase bilan integratsiya qilish :

- Flutter ilovasi uchun Firebase SDK-sini o'rnatish:

```
---
```

```
flutter pub add firebase_core
```

```
---
```

- Firebase konsolida yangi loyiha yaratish va Flutter ilovasi uchun konfiguratsiya fayl (google-services.json) olish.

- Flutter ilovasi ichida Firebase konfiguratsiyasini o'rnatish:

```
```dart
```

```
await Firebase.initializeApp(
 options: DefaultFirebaseOptions.currentPlatform,
);
```

```

```

### 2. Firestore ma'lumotlar bazasi bilan ishlash :

- Firestore ma'lumotlar bazasidan foydalanish uchun `firebase\_firestore` paketini o'rnatish:

```

```

```
flutter pub add firebase_firestore
```

```

```

- Firestore ma'lumotlar bazasiga ma'lumotlarni saqlash:

```
```dart
```

```
final docRef = FirebaseFirestore.instance.collection('users').doc('123');  
await docRef.set({
```

```
'name': 'John Doe',  
'age': 30,  
});  
---
```

- Firestore ma'lumotlar bazasidan ma'lumotlarni o'qish:

```
```dart  

final docRef = FirebaseFirestore.instance.collection('users').doc('123');
final snapshot = await docRef.get();
final data = snapshot.data();

```

### 3. Authentication tizimi bilan integratsiya :

- Authentication tizimidan foydalanish uchun `firebase\_auth` paketini o'rnatish:

```

flutter pub add firebase_auth

```

- Foydalanuvchini autentifikatsiyadan o'tkazish:

```
```dart  
  
final userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(  
  email: 'john@example.com',  
  password: 'supersecretpassword',  
);  
---
```

- Foydalanuvchi ma'lumotlariga kirish:

```
```dart  

final user = FirebaseAuth.instance.currentUser;
final name = user?.displayName;
final email = user?.email;
```

```

4. Xavfsizlik va sozlamalar :

- Firebase Firestore va Authentication uchun xavfsizlik qoidalarini sozlash.
- Firebase konsolida ilovaning platform sozlamalarini konfiguratsiya qilish.
- Firebase Analytics, Crashlytics va boshqa xizmatlarni ilovaga integratsiya qilish.

Yuqoridagi yondashuvlar Firebase ma'lumotlar bazasini Flutter ilovasi bilan integratsiya qilish va sozlash bo'yicha asosiy prinsiplarini ifoda etadi. Firebase bilan to'g'ri integratsiya qilish ilovaning xavfsizligi, funkSIONalligi va foydalanuvchi tajribasi sifatini yaxshilashga yordam beradi.

Firebase ma'lumotlar bazasini Android ilovaga ulash va sozlash uchun quyidagi qadamlar bo'yicha amalga oshirishingiz mumkin:

1. Firebase loyihani ochildirish : Firebase veb saytiga kiring va yangi loyihani qo'shing yoki mavjud loyihani tanlang.
2. Firebase ilovangizni Android ilovangizga qo'shish : Firebase veb saytida "Proyekt qo'shish" tugmasini bosib va Android ilovangizni qo'shing. Bundan keyin, `google-services.json` faylini oling va ilovangiz `app` papkasining ichiga joylashtiring.
3. Gradle faylini o'zgartirish : Ilova projektining `build.gradle` faylini oching va quyidagi qatorni qo'shing:

```
```gradle  
apply plugin: 'com.google.gms.google-services'
```
```

4. Firebase Android SDK-ni qo'shish : Ilova projektining `build.gradle` faylini oching va Firebase Android SDK-ni qo'shib, ilovangizga ma'lumotlar bazasiga ulanish imkoniyatini ta'minlang:

```
```gradle  
implementation 'com.google.firebase:firebase-database-ktx'
```
```

5. Firebase-ni ishga tushirish : Ilovaning boshlang'ich aktivitasi yoki ilova klassida Firebase-ni boshqarish uchun Firebase-ni boshqaruvchi obyekt yarating:

```
```java
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");
```
```

6. Ma'lumotlarga qo'shish : Ma'lumotlar bazasiga ma'lumotlarni qo'shish uchun `setValue()` metodidan foydalaning:

```
```java
myRef.setValue("Hello, World!");
```
```

7. Ma'lumotlarni o'qish : Ma'lumotlar bazasidan ma'lumotlarni o'qish uchun `addValueEventListener()` yoki `addListenerForSingleValueEvent()` metodlaridan foydalaning:

```
```java
myRef.addValueEventListener(new ValueEventListener() {
 @Override
 public void onDataChange(DataSnapshot dataSnapshot) {
 String value = dataSnapshot.getValue(String.class);
 Log.d(TAG, "Value is: " + value);
 }

 @Override
 public void onCancelled(DatabaseError error) {
 Log.w(TAG, "Failed to read value.", error.toException());
 }
});
```
```

```

8. Ma'lumotlarni yangilash : Ma'lumotlar bazasidagi ma'lumotlarni yangilash uchun `updateChildren()` metodidan foydalaning:

```java

```
Map<String, Object> childUpdates = new HashMap<>();
```

```
childUpdates.put("/message", "New message");
```

```
myRef.updateChildren(childUpdates);
```

```

Bu qadamlar Firebase ma'lumotlar bazasiga Android ilovadan ulanishni ta'minlash uchun kerakli asosiy qadamalardir. Ular orqali ilovangiz Firebase ma'lumotlar bazasiga ma'lumotlarni qo'shish, o'qish, yangilash va o'chirish imkoniyatiga ega bo'ladi.

## **S119. Mobil ilovalarda fayllar bilan ishlash**

Mobil ilovalarda fayllar bilan ishlash quyidagi asosiy vazifalarni o'z ichiga oladi:

1. Fayllarni o'qish : Fayllardan ma'lumotlarni olish va ularga murojaat qilish.
2. Fayllarga yozish : Yangi fayllar yaratish va fayllarga ma'lumotlar yozish.
3. Fayllarni o'chirish : Bor fayllarni o'chirish.
4. Fayllarni o'zgartirish : Ma'lumotlarni mavjud fayllarda o'zgartirish.

Bu amallarni bajarish uchun, Android ilovalarida quyidagi usullardan biri yoki bir nechta ishlatiladi:

1. Fayl kutubxonasi (File API) : Androidda standart Java fayl kutubxonasiga murojaat qilish orqali fayllar bilan ishlash mumkin. Bu usul orqali fayllarni o'qish, yozish, o'chirish va o'zgartirish amallarini bajarish mumkin.
2. Fayl ishlab chiqish (File Creation) : Yangi fayllar yaratish uchun fayl ishlab chiqish usuli ishlatiladi. Bu usul orqali ilova tomonidan yangi fayllar yaratiladi va ularga ma'lumotlar yoziladi.
3. ContentProvider orqali fayllar bilan ishlash : Androidda fayllar bilan ishlash uchun `ContentProvider` interfeysi mavjud. U orqali ilova foydalanuvchining fayllariga o'qish, yozish va o'chirish imkoniyatiga ega bo'ladi.
4. External Storage Access Framework : Android 4.4 (API 19) dan boshlab, ilova foydalanuvchining tashqi xotirasidagi fayllariga o'zaro amalga oshirish uchun `Storage Access Framework`ni (SAF) ishlatishi mumkin.
5. Fayluri (FileUri) : Fayllarga murojaat qilish uchun `FileUri` ishlatilishi mumkin. Bu usul orqali foydalanuvchi fayllariga murojaat qilish imkoniyati mavjud.
6. Fayluri (FileUri)larni almashish : Fayllarni o'qish va yozish uchun `FileUri`larni almashish usuli ishlatiladi. Bu usul orqali ilova fayllariga murojaat qilish va ma'lumotlar yozish mumkin.

Har bir usulning o'ziga xos afzalliklari va cheklari mavjud. Qaysi usulni tanlash kerakligiga ilova xususiyatlariga va vazifalarga bog'liq. Ma'lumotlar bazasini saqlash, yuklash va fayllar bilan ishlashning qanday bir qismini o'rganish va uni amalga oshirish kerakligiga qarab, mos usulni tanlash muhimdir.

Mobil ilovalarida fayllar bilan ishlash juda muhim masaladir. Quyida mobil ilovalarida fayllar bilan ishlashning asosiy yondashuvlarini ko'rib chiqamiz:

## 1. Fayllar bilan ishlash uchun ruxsatlarni olish :

- Android uchun `storage` ruxsatini olish:

```
```dart
// Manifest faylida ruxsatni talablar
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />

// Kodda ruxsatni so'rash
var status = await Permission.storage.request();
if (status.isGranted) {
  // Fayllar bilan ishlashni boshlash
}
```
```

- iOS uchun `photo` va `files` ruxsatlarini olish:

```
```dart
// Info.plist faylida ruxsatlar talablari
<key>NSPhotoLibraryUsageDescription</key>
<string>This app requires access to the photo library.</string>
<key>NSDocumentsFolderUsageDescription</key>
<string>This app requires access to the documents folder.</string>

// Kodda ruxsatlarni so'rash
var status = await Permission.photos.request();
if (status.isGranted) {
  // Fayllar bilan ishlashni boshlash
}
```



```
}  
```
```

## 2. Fayllarni saqlash :

- Ichki saqlash: `path\_provider` paketidan foydalanish:

```
```dart  
  
final directory = await getApplicationDocumentsDirectory();  
final file = File('${directory.path}/example.txt');  
await file.writeAsString('Hello, World!');  
```
```

- Tashqi saqlash: `permission` paketi yordamida ruxsatlar olish kerak:

```
```dart  
  
final directory = (await getExternalStorageDirectory())!.path;  
final file = File('$directory/example.txt');  
await file.writeAsString('Hello, World!');  
```
```

## 3. Fayllarni o'qish :

- Ichki saqlangan fayl:

```
```dart  
  
final directory = await getApplicationDocumentsDirectory();  
final file = File('${directory.path}/example.txt');  
final contents = await file.readAsString();  
print(contents); // Hello, World!  
```
```

- Tashqi saqlangan fayl:

```
```dart
```

```
final directory = (await getExternalStorageDirectory())!.path;
final file = File('$directory/example.txt');
final contents = await file.readAsString();
print(contents); // Hello, World!
```
```

#### 4. Rasmlar bilan ishlash :

- Rasm saqlash:

```
```dart
final pickedFile = await ImagePicker().pickImage(source: ImageSource.gallery);
final file = File(pickedFile!.path);
await file.writeAsBytes(await pickedFile.readAsBytes());
```
```

- Rasm o'qish:

```
```dart
final directory = await getApplicationDocumentsDirectory();
final file = File('${directory.path}/example.jpg');
final image = FileImage(file);
```
```

Yuqoridagi yondashuvlar mobil ilovalarida fayllar bilan ishlashning asosiy prinsiplarini ifoda etadi. Fayllar bilan to'g'ri ishlash ilovaning funkcionalligi va foydalanuvchi tajribasi sifatini yaxshilashga yordam beradi.

## 120. Mobil ilovalarda geolokatsiya bilan ishlash

Mobil ilovalarda geolokatsiya bilan ishlash juda maqbul va ko'p qo'llaniladigan amaldir. Bu, foydalanuvchining joriy joylashuvi (koordinatalari)ni aniqlash va ularga asosan aloqador ma'lumotlarni taqdim etish imkonini beradi. Android platformasida geolokatsiya bilan ishlash quyidagi vazifalarni o'z ichiga oladi:

1. Joriy joylashuvni aniqlash : Foydalanuvchining joriy joylashuvi (enlem va boylam)ni aniqlash uchun GPS, Wi-Fi va mobil operatorning tarmoqlari yordamida.
2. Yordamchi xizmatlarga murojaat qilish : Google Maps, yol yo'lni topish, mehmonxonalar, restoranlar va boshqa joylarga yo'l yo'riqnomasi yaratish va foydalanuvchiga yo'l ko'rsatish uchun geolokatsiya ma'lumotlaridan foydalanish.
3. Yakuniy maqsadlarni aniqlash : Mijozlarning mavjudlikni aniqlash uchun do'konlar, restoranlar, yubiley va ko'ngilloklar va boshqa joylarga yonlendirish.
4. Ko'chma ko'rsatkichlarni aniqlash : Ko'chma turli narsalarni qidirish, yaqin joylar, do'konlar, restoranlar va boshqa joylar haqida ma'lumotlar bilan ta'minlash.
5. Harakatlanuvni kuzatish : Mijozlar to'g'risida qandaydir holatni (yurish, uchrashuv, transport vositalari bilan harakatlanish) aniqlash.
6. Foydalanuvchining harakatlarini kuzatish : Spor ilovalarida, masalan, yugurish yo'li, yog'alar, vaqtlarni vaqtni boshqarish uchun foydalanuvchining harakatlarini kuzatish.

Android platformasida geolokatsiya bilan ishlash uchun quyidagi vazifalarni bajarish mumkin:

1. LocationManager : Bu obyekt foydalanuvchining joriy joylashuvi ma'lumotlariga murojaat qilish uchun ishlatiladi.
2. Google Play Services Location API : Google Play xizmatlarining yordamchi geolokatsiya API si. U ishlatilganida, foydalanuvchining joriy joylashuvi ma'lumotlarini to'liq yuklash va yaxshi amalga oshirilishi uchun qo'llaniladi.
3. Google Maps Android API : Bu API orqali foydalanuvchining joylashuvi ma'lumotlarini harita shaklida ko'rsatish va boshqa joylar bilan ulash uchun ishlatiladi.
4. Fused Location Provider API : Bu API GPS, Wi-Fi, va mobil tarmoqlari kabi ko'rsatkichlardan o'qishning eng yaxshi va energiya samarali usulini taqdim etadi.

Geolokatsiya bilan ishlashning tezligi, ishonchli va qat'iylik, Android ilovasining foydalanuvchilarga qulaylik va foydalanishni osonlashtirish uchun muhim bir qismidir.

## S121. Mobil ilovalar yaratish uchun dasturlash muhiti

Mobil ilovalar yaratish uchun bir nechta dasturlash muhitlari mavjud, ammo iOS va Android platformalariga ilova yaratish uchun eng mashhur dasturlash tillari va muhitlari quyidagilardir:

Android:

1. Java/Kotlin : Android ilovalarini yaratish uchun asosiy til. Kotlin hozirgi kunda Android ilovalarini yaratish uchun keng qo'llaniladi.
2. Android Studio : Android ilovalarini yaratish va ishga tushirish uchun ofitsial dasturlash muhiti. Bu IntelliJ IDEA-ning Android versiyasidir va Android ilovalarini yaratish va test qilish uchun keng imkoniyatlar taqdim etadi.
3. Gradle : Android ilovalarining qurilish va boshqaruvini avtomatlashgan dasturlash vositasi.
4. Android SDK : Android platformasining rivojlanishi uchun kerakli barcha qo'shimcha kitobxonalar va resurslar.

iOS:

1. Swift/Objective-C : iOS ilovalarini yaratish uchun asosiy til. Swift hozirgi kunda iOS ilovalarini yaratish uchun eng qo'llaniladi.
2. Xcode : Apple tomonidan taqdim etilgan ofitsial dasturlash muhiti. Bu muhit iOS ilovalarini yaratish, boshqarish va ishga tushirish imkoniyatlarini taqdim etadi.
3. Cocoa Touch : iOS ilovalari uchun API va xizmatlarni taqdim etadi.
4. UIKit : Applening ishlab chiqarilgan grafik foydalanuvchi interfeysi toolkit'i. iOS ilovalarining grafik interfeysini yaratish uchun keng qo'llaniladi.
5. Core Data : Applening ma'lumotlarni o'z ichiga oluvchi muhit. U, ma'lumotlarni saqlash, boshqarish va saqlash imkoniyatlarini taqdim etadi.

Kross-platfomali dasturlash:

1. Flutter : Google tomonidan taqdim etilgan xavfsiz, kross-platfomali dasturlash muhiti. Bu, iOS va Android ilovalarini yaratish uchun yeg'indisi qo'llaniladi va Dart tilida yoziladi.
2. React Native : Facebook tomonidan ishlab chiqilgan, JavaScript asosida iOS va Android ilovalarini yaratish uchun kross-platfomali muhit.

Bu muhitlar yordamida dasturchilar iOS va Android platformalarida ilovalar yaratish, test qilish va ishga tushirish uchun ilova yaratishlari va boshqarishlari mumkin. Har bir muhitning o'ziga xos afzalliklari va cheklari mavjud, shuning uchun loyihangizning talablari va talablari bo'yicha eng muvozanatli muhitni tanlash juda muhimdir.

Mobil ilovalar yaratish uchun eng mashhur dasturlash muhitlari quyidagilar:

1. Android Studio :

- Android ilovalar uchun asosiy dasturlash muhiti
- Kotlin va Java tillarida dasturlash qo'llab-quvvatlaydi
- Integrated Development Environment (IDE) sifatida ishlaydi
- Xatolarni tekshirish, qidirish va tuzatish vositalari bilan ta'minlangan
- Platforma-uchun-optimallashtirilgan emulator va debugging vositalarini o'z ichiga oladi

2. Xcode :

- iOS ilovalar uchun asosiy dasturlash muhiti
- Swift va Objective-C tillarida dasturlash qo'llab-quvvatlaydi
- Integrated Development Environment (IDE) sifatida ishlaydi
- Xatolarni tekshirish, qidirish va tuzatish vositalari bilan ta'minlangan
- Platforma-uchun-optimallashtirilgan emulator va debugging vositalarini o'z ichiga oladi

3. Visual Studio Code :

- Ko'p platforma uchun universal IDE
- Flutter, React Native va Xamarin kabi aloqa qiluvchi tillar uchun qo'llab-quvvatlangan
- Xatolarni tekshirish, qidirish va tuzatish vositalari bilan ta'minlangan
- Keng pluginlar ekotizimi bilan ta'minlangan
- Platforma-uchun-optimallashtirilgan emulator va debugging vositalarini qo'llab-quvvatlaydi

#### 4. IntelliJ IDEA :

- Android, iOS, Flutter va boshqa platforma uchun ilovalar yaratish uchun qo'llaniladi
- Kotlin, Java, Dart va boshqa tillarni qo'llab-quvvatlaydi
- Xatolarni tekshirish, qidirish va tuzatish vositalari bilan ta'minlangan
- Android Studio'ga o'xshash platforma-uchun-optimallashtirilgan emulator va debugging vositalarini o'z ichiga oladi

#### 5. React Native CLI :

- JavaScript/TypeScript asosida mobil ilovalar yaratish uchun qo'llaniladi
- Qator platforma uchun birlashgan kod yozishga imkon beradi
- Xatolarni tekshirish, qidirish va tuzatish vositalari bilan ta'minlangan
- Platforma-uchun-optimallashtirilgan emulator va debugging vositalarini qo'llab-quvvatlaydi

Bu dasturlash muhitlari mobil ilovalar yaratish uchun keng qo'llaniladi. Har biri o'zining afzalliklari va kamchiliklariga ega, shuning uchun loyiha talablariga qarab to'g'ri muhitni tanlash muhim.

## S122. Mobil ilovalarda ma'lumotlar bazasi

Mobil ilovalarida ma'lumotlar bazasi bilan ishlash juda muhim. Mobil ilovalar odatda mahalliy ma'lumotlar bazasidan va/yoki bulutli ma'lumotlar bazalaridan foydalanadi.

Yuqoridagi yondashuvlar mobil ilovalarida mahalliy va bulutli ma'lumotlar bazasi bilan ishlashning asosiy prinsiplarini ifoda etadi. Loyiha talablariga qarab to'g'ri ma'lumotlar bazasi texnologiyasini tanlash va uning samarali ishlashini ta'minlash muhimdir.

Mobil ilovalarda ma'lumotlar bazasi (MDB) bir ilova tomonidan ma'lumotlarni saqlash va ulash uchun ishlatiladi. Ma'lumotlar bazasi, ilova ma'lumotlarini o'z ichiga oladi va ularni saqlash, o'qish, yozish va o'zgartirish imkoniyatlarini ta'minlaydi. Mobil ilovalarda ma'lumotlar bazasini amalga oshirish uchun bir nechta turli tashqi ma'lumotlar bazalari mavjud, bu orqali ilovalar ma'lumotlar bazasiga murojaat qila oladi. Quyidagi keng qo'llaniladigan MDB lar mavjud:

Android:

1. SQLite : Androidning ishlatiladigan boshlang'ich tashqi ma'lumotlar bazasi. Bu o'rnatilgan va qo'llab-quvvatlanadigan relatsion ma'lumotlar bazasi boshlang'ich tashqi ma'lumotlar bazasi sifatida qo'llaniladi.
2. Room Persistence Library : Android ilovalarida ma'lumotlar bazasini boshqarish uchun Android Room vaqti tarkibida qo'shilgan. Bu, SQLite ma'lumotlar bazasini boshqarish uchun amalga oshirilgan shakl va imkoniyatlarga ega ko'plab komponentlar to'plami hisoblanadi.
3. Firebase Realtime Database : Google Firebase tomonidan taqdim etilgan, Android ilovalarida ma'lumotlarni saqlash uchun obid bo'lgan, obunalish ma'lumotlar bazasi. U, obid ma'lumotlar bazasi sifatida ishlaydi va ma'lumotlar obunalish ma'lumotlar bazasi uchun yashilaydi.
4. Firebase Cloud Firestore : Firebase tomonidan taqdim etilgan, obunalish va dokument ma'lumotlar bazasi. Firestore, qulay API va qulay struktura bilan ma'lumotlar bazasiga o'zlashtirilgan va yuqori qatlamli yozuvlar uchun yuqori tezlikda qulaylik taqdim etadi.



iOS:

1. Core Data : iOS ilovalarida ishlatiladigan ma'lumotlar bazasi kerakli ma'lumotlarini saqlash uchun Apple tomonidan taqdim etilgan tashqi ma'lumotlar bazasi. Bu, obid ma'lumotlar bazasi sifatida ishlaydi va obid ma'lumotlar bazasi uchun yashilaydi.
2. Realm : Yengil va tez iOS ilovasi obunalish ma'lumotlar bazasi. Realm, qulay API va ma'lumotlarni tezkor yuklab olish uchun yuqori darajada tanlash mumkin.
3. Firebase Realtime Database : Firebase tomonidan taqdim etilgan, obunalish ma'lumotlar bazasi. U, obid ma'lumotlar bazasi sifatida ishlaydi va ma'lumotlar obunalish ma'lumatlar bazasi uchun yashilaydi.

MDB lar ilovalarda ma'lumotlarni saqlash, o'qish, yozish va o'zgartirish uchun kerakli funksionalni ta'minlaydi. Har bir MDB ning o'ziga xos afzalliklari va cheklari mavjud, shuning uchun loyihangizning talablari va talablari bo'yicha eng muvozanatli MDB ni tanlash juda muhimdir.

## **S123. Mobil ilovalarda xabarlar almashish xizmatlari**

Mobil ilovalarda xabarlar almashish xizmatlari, ilovalar va foydalanuvchilar orasidagi asosiy kommunikatsiya vositalaridan biridir. Bu xizmatlar, foydalanuvchilarga qulaylik va joynamozlik keltirib chiqarish, ilovalar uchun yangiliklarni, bildirishnomalar, xabarlar va boshqa ma'lumotlarni yetkazish imkoniyatlarini ta'minlash uchun ishlatiladi. Quyidagi keng qo'llaniladigan mobil ilovalarda xabarlar almashish xizmatlari mavjud:

1. Push Xabarlar : Bu, mobil ilova ilovasining qurilishiga qo'shimcha funktsiyalar qo'shadi. Ilgari belgilangan voqealar yoki ma'lumotlar uchun foydalanuvchilarga bildirishnomalar yuborish uchun ishlatiladi. iOS va Android platformalarida Firebase Cloud Messaging (FCM) va Apple Push Notification Service (APNs) kabi xizmatlar foydalanish mumkin.
2. SMS Xizmatlari : Bu xizmatlar orqali mobil ilovalar istalgan vaqtda foydalanuvchilarga SMS xabarlar yuborishadi. Bu xizmatlar ko'p darajada qo'llaniladi va xabarlarni to'g'ri mobil qurilmalariga, o'zi ilova orqali yuboriladi.
3. In-App Xabarlar : Bu xizmatlar ilovada amalga oshiriladi va foydalanuvchi ilova yoki o'yin o'ynash paytida ma'lumotlar, takliflar va xabarlar yuborish uchun ishlatiladi. Bu xabarlar ilova interfeysi ichida ko'rsatiladi.
4. E-mail Xizmatlari : Bu, foydalanuvchilarga xabarlar yuborish uchun klassik e-mail xizmatlaridan foydalanishni ta'minlaydi. Bu xizmatlar mobil ilova orqali yoki ilovadan tashqari xizmatlardan foydalanilishi mumkin.
5. Websocket Xizmatlari : Bu xizmatlar orqali real vaqt ma'lumot almashish va xabarlarini olish uchun ishlatiladi. Bu, ilova va server orasidagi dinamik aloqalarni ta'minlash uchun qo'llaniladi.
6. Local Notifications : Bu, ilova tomonidan o'z ichiga yuboriladigan xabarlarni bildirishnomalar orqali foydalanuvchiga taqdim etish uchun ishlatiladi. Bu xizmatlar qurilmada ishlaydi va internet aloqasi talab qilmaydi.

Har bir xizmatning o'ziga xos afzalliklari va cheklari mavjud. Mahsulotlarni oson va samarali yuklab olish, ilovani qiziqtirish, foydalanuvchilarga ma'lumotlar yetkazish va ma'lumotlar bazasidan xabarlar olishning har biri uchun mos xizmatni tanlash juda muhimdir.

Mobil ilovalarida xabarlar almashish xizmatlari juda muhim. Ular ilovalar orasida va foydalanuvchilar orasida interaktiv aloqani ta'minlaydi. Quyida mobil ilovalarida foydalaniladigan asosiy xabarlar almashish xizmatlarini ko'rib chiqamiz:

### 1. Firebase Cloud Messaging (FCM) :

- Google tomonidan taqdim etilgan xabarlar almashish xizmati
- Android va iOS uchun qo'llab-quvvatlangan
- Push notifications, real-time messaging va boshqa xizmatlarni taqdim etadi
- Flutter uchun FCM paketidan foydalanish oson:

```
```dart
// Firebase SDK'ni o'rnatish
flutter pub add firebase_core
flutter pub add firebase_messaging
// FCM serveriga ulanish
FirebaseMessaging messaging = FirebaseMessaging.instance;
await messaging.requestPermission();
// Push notification olish
FirebaseMessaging.onMessage.listen((RemoteMessage message) {
  // Incoming message processing
});
```
```

### 2. Apple Push Notification Service (APNS) :

- iOS va iPadOS uchun Apple tomonidan taqdim etilgan xabarlar almashish xizmati
- Push notifications, real-time messaging va boshqa xizmatlarni taqdim etadi
- Xizmatdan foydalanish uchun Apple Developer accountiga ega bo'lish kerak
- Flutter uchun APNS integratsiyasi Xcode va Firebase Cloud Messaging orqali amalga oshiriladi

### 3. Amazon Pinpoint :

- AWS tomonidan taqdim etilgan xabarlar almashish xizmati
- Android, iOS, web va boshqa platforma uchun qo'llab-quvvatlangan
- Push notifications, in-app messaging, email va SMS yetkazish xizmatlarini taqdim etadi
- Flutter uchun Amazon Pinpoint SDK orqali integratsiya qilish mumkin:

```
```dart
// Amazon Pinpoint SDK'ni o'rnatish
flutter pub add aws_pinpoint
// Pinpoint clientini yaratish
final pinpoint = PinpointClient(
  region: 'us-east-1',
  credentials: AwsClientCredentials(
    accessKey: 'YOUR_ACCESS_KEY',
    secretKey: 'YOUR_SECRET_KEY',
  ),
);
// Push notification yubortish
await pinpoint.sendUserPush(
  ApplicationId: 'YOUR_APP_ID',
  RequestParameters: {
    'title': 'Hello',
    'body': 'This is a push notification',
  },
);
```
```

Bu xabarlar almashish xizmatlaridan biri yoki bir nechasi mobil ilovangiz uchun eng mos keladigan yechim bo'lishi mumkin. Ularning imkoniyatlari, yetkazish ishonchliligi, narxi va mobil platformaga mosligi asosida to'g'ri xizmatni tanlang.

## S124. Server bilan ishlash

Mobil ilovalarda server bilan ishlash, ilovaning serverga murojaat qilishi, ma'lumotlarni yuklash va olish uchun serverdan foydalanishni ta'minlashni anglatadi. Bu, ilova va server o'rtasidagi aloqalarni tuzish, ma'lumot almashish, yuklash va o'zgartirishni amalga oshirish uchun kerakdir. Quyidagi asosiy usullar server bilan mobil ilova o'rtasidagi aloqalarni amalga oshirish uchun qo'llaniladi:

1. RESTful API : REST (Representational State Transfer) API, ilova va server o'rtasidagi o'zaro aloqani ta'minlash uchun keng qo'llaniladigan usuldir. Bu, ma'lumotlar uchun standart protokollarni ishlatadi, masalan, HTTP. Ilova, HTTP so'rovlarini yuboradi (GET, POST, PUT, DELETE) va server javobini qabul qiladi. Ma'lumotlar JSON, XML yoki boshqa formatlarda o'z ichiga oladi.
2. SOAP (Simple Object Access Protocol) : SOAP, ilova va server o'rtasidagi aloqani ta'minlash uchun boshqa tanlangan xizmat protokollari orasida keng qo'llanilmagan. Bu, XML protokolida ma'lumot almashishni ta'minlaydi va bu protokol xavfsizlik, idish kontrol va qo'shimcha xizmatlar kabi afzalliklarga ega bo'lsa-da, yagona ilova uchun kuchli emas.
3. WebSocket : WebSocket, ilova va server orasidagi ikki tomonlama aloqa tizimini ta'minlaydi. Bu, ilova va server orasidagi dinamik vaqt ma'lumot almashish uchun qo'llaniladi. Bu usul, ilovaga avtomatik ravishda yangilashlar yuboradi, bu esa ilovani dinamik va qulay qiladi.
4. GraphQL : GraphQL, foydalanuvchining ma'lumotlarni spetsifikatsiyasi bo'yicha so'rovlar yuborish uchun ishlatiladigan yangi API tanlovidir. Bu, ilovaga kerak bo'lgan ma'lumotlarni aniq vaqt davomida so'rash imkonini beradi va bu yordamda ilovani tezroq va samarali qiladi.

Server bilan ishlashning asosiy vazifalari quyidagilarini o'z ichiga oladi:

- Ma'lumotlarni olish : Ilova, serverga so'rovlar yuborib, ma'lumotlarni yuklaydi.
- Ma'lumotlarni yuklash : Foydalanuvchi tomonidan kiritilgan ma'lumotlarni serverga yuborish.
- Ma'lumotlarni o'zgartirish : Ma'lumotlarni serverda o'zgartirish, yangilash va o'chirish.
- Autentifikatsiya va xavfsizlik : Foydalanuvchilar o'zlarini autentifikatsiya qilish uchun serverga murojaat qilish, ma'lumotlar muhafazasi va xavfsizligi.
- Ma'lumotlar ombori : Ma'lumotlar bazasi yoki server yordamida ilova ma'lumotlarini saqlash va ulash.

Server bilan ilova orasidagi aloqalar, ilovaning funksionalini kengaytiradi va foydalanuvchilarga qulayliklar keltiradi. Raqamli aloqa, ma'lumotlar bazasi, shifrlash va boshqa xavfsizlik muammolari dasturchilar uchun muhimdir va server va ilova o'rtasidagi aloqalar ularni yechish uchun muhimdir.

## **S125. Android tizimining kushimcha xizmatlari. Google maps xizmati**

Android tizimining kushimcha xizmatlari ko'p turli yo'nalishlarda foydalaniladi va Google Maps xizmati bu tizimning biridir. Bu xizmat, mobil ilovalar uchun joriy joylashuvi bilan bog'liq ma'lumotlarni taqdim etish, yo'l yo'riqnomalarni olish va interaktiv kartalar yaratish imkonini beradi. Quyidagi Android tizimining kushimcha xizmatlari mavjud:

1. Google Maps API : Bu API, Android ilovalariga interaktiv xaritalarni qo'shish imkonini beradi. Ilgari belgilangan joylar, boshqarish va yo'l yo'riqnomalar qidirish, yo'l yo'riqnomalar yaratish, manzillarni topish, navigatsiya, geolokatsiya ma'lumotlari olish va boshqalar uchun qo'llaniladi.
2. Google Location Services API : Bu API, foydalanuvchining joriy joylashuvi (geolokatsiyasi) ma'lumotlarini olishni ta'minlaydi. Bu, GPS, Wi-Fi va mobil operator tarmog'iga murojaat qilib, foydalanuvchining to'g'ri joriy joylashuvi ma'lumotlarini olish uchun qo'llaniladi.
3. Google Places API : Bu API, foydalanuvchining ma'lum bir joyda joylashgan restoranlar, do'konlar, mehmonxonalarni qidirish uchun qo'llaniladi. Bu, qidiruv va topish funksiyalarini taqdim etadi va foydalanuvchilarga ma'lum bir joy haqida qo'llanuvchilarning fikrlarini ham olishga imkon beradi.
4. Google Directions API : Bu API, foydalanuvchining ikki nuqta orasidagi yo'l yo'riqnomasini topish va yaratish imkonini beradi. Bu, boshqarishlar uchun yo'l yo'riqnomalar yaratish va foydalanuvchilarga moslashtirilgan yo'l yo'riqnomalar taqdim etishga imkon beradi.
5. Google Street View API : Bu API, foydalanuvchilarga har bir joyda fotografik manzaralarni ko'rishga imkon beradi. Bu, kartalardagi joylarni harakatlab, foydalanuvchilarga interaktiv ko'rish va ma'lumotlarni qidirish uchun qo'llaniladi.

Bu Google xizmatlari, Android ilovalarining foydalanuvchilarga qulayliklar keltirish uchun eng mashhur va kuchli kushimcha xizmatlardan ba'zilari. Bu xizmatlar, ilova funksiyalarini kengaytirish, joylashuvi ma'lumotlarini qidirish va ko'rsatish, yo'l yo'riqnomalar yaratish, navigatsiya va boshqalar uchun keng qo'llaniladi.

Android tizimining qo'shimcha xizmatlari orasida Google Maps xizmati juda keng tarqalgan va kuchli imkoniyatlarga ega. Bu xizmatni mobil ilovalar uchun quyidagi usullar bilan integratsiya qilish mumkin:



## S126. Mobil ilovalarni kross-platformali dasturlash

Kross-platformali dasturlash, bir qatorda, bir ilovani bir nechta platformalar uchun yaratishni anglatadi. Bu, dasturchilarga iOS, Android, va boshqa platformalar uchun bir necha dastur yozish va ularga bo'lgan talablarni qisqa vaqt ichida soddaqayta tushuntirish imkonini beradi. Kross-platformali dasturlashning eng mashhur usullaridan ba'zilar quyidagilar:

1. Flutter : Google tomonidan taqdim etilgan, Dart tilida yozilgan kross-platformali dasturlash muhiti. Flutter, ilovani iOS, Android, web va desktop platformalariga o'rnatingan har qanday operatsion tizimda ishlatish imkonini beradi. Bu, birinchi natijalar uchun tez ish va sodda interfeyslar yaratish uchun yaxshi tanlovidir.
2. React Native : Facebook tomonidan yaratilgan, JavaScript tilida yozilgan, kross-platformali dasturlash muhiti. Bu, React.js kengaytmasiga asoslangan va ilovalarni iOS va Android platformalariga o'rnatingan qulay va samarali yechimlarni taqdim etadi.
3. Xamarin : Microsoft tomonidan ishlab chiqilgan, C# tilida yozilgan, kross-platformali dasturlash muhiti. Bu, .NET platformasi orqali iOS, Android va Windows uchun ilovalarni yaratish imkonini beradi.
4. PhoneGap/Cordova : Adobe tomonidan ishlab chiqilgan, HTML, CSS va JavaScript orqali yozilgan, kross-platformali dasturlash muhiti. Bu, Web ilovalarni mobil ilovalarga o'rnatingan oson usuldir. PhoneGap ilovani barcha platformalar uchun paketlaydi va ishga tushiradi.
5. Ionic : Angular.js kengaytmasiga asoslangan, PhoneGap/Cordova orqali paketlangan, kross-platformali dasturlash muhiti. Bu, mobil ilovani yaratishda HTML, CSS va JavaScript tilidan foydalanishni ta'minlaydi.

Har bir kross-platformali dasturlash muhiti o'ziga xos afzalliklarga ega va qulayliklarga yoltutadi. Tanlangan muhit, loyihangizning talablari, dastur rivojlanish jarayonining davomiyligi va jamoatchilik (community) muhim ko'rsatkichlari bo'lib hisoblanishi kerak.

## S127. Server bilan ishlash

Server bilan ishlash, mobil ilovalarning asosiy qismi bo'lib, ma'lumotlar almashish, ma'lumotlarni saqlash, foydalanuvchilar orasida aloqa o'rnatish, autentifikatsiya, ma'lumotlarni sinxronlashtirish va boshqa qo'llanishga oid funksiyalarni o'z ichiga oladi. Mobil ilova va server o'rtasidagi aloqalarni tuzishning asosiy vazifalari quyidagilarini o'z ichiga oladi:

1. Ma'lumotlar almashish : Mobil ilova, serverga so'rovlar yuborib, kerakli ma'lumotlarni olish uchun murojaat qiladi. Bu, foydalanuvchilarning ma'lumotlarini yuklash, xabarlar olish, yangiliklarni olish, qidiruvlar va boshqa ma'lumotlarni olishda ishlatiladi.
2. Ma'lumotlarni yuklash : Foydalanuvchilar tomonidan kiritilgan ma'lumotlar serverga yuboriladi va ma'lumotlar bazasiga saqlanadi. Bu, ro'yxatdan o'tish ma'lumotlari, foydalanuvchi profili ma'lumotlari, raxmatnomalar va boshqa ma'lumotlar uchun qo'llaniladi.
3. Ma'lumotlarni sinxronlashtirish : Mobil ilova va server o'rtasidagi ma'lumotlar bazalari o'rtasidagi ma'lumotlar uchun to'g'ri, so'nggi xolatlarni ta'minlash uchun sinxronlashtirilishi kerak. Bu, foydalanuvchilarning ilovalarini bir nechta qurilmalarda ishlatish imkonini beradi.
4. Autentifikatsiya va xavfsizlik : Foydalanuvchilar o'zlarini autentifikatsiya qilish uchun serverga murojaat qilishadi, bu esa foydalanuvchilar uchun maxfiylik va xavfsizlikni ta'minlash uchun muhimdir. Bu, foydalanuvchilarning parollarni hashlash, tokenlar va boshqa xavfsizlik protokollari orqali amalga oshiriladi.
5. Ma'lumotlarni saqlash : Mobil ilovalar foydalanuvchilarning ma'lumotlarini saqlash uchun serverga murojaat qiladi. Bu, ma'lumotlar bazasidagi fayllarni yuklash, ma'lumotlar bazasidagi ma'lumotlarni yangilash, o'chirish va boshqalar uchun ishlatiladi.

Server bilan mobil ilova orasidagi aloqalar, ilova funksiyalarini kengaytirish, joylashuvi ma'lumotlarini qidirish va ko'rsatish, foydalanuvchilar orasida aloqa o'rnatish va boshqalar uchun kritik muhim vazifalardan birini o'z ichiga oladi. Raqamli aloqa, ma'lumotlar bazasi, shifrlash va boshqa xavfsizlik muammolari dasturchilar uchun muhimdir va server va ilova o'rtasidagi aloqalar bularni yechish uchun muhimdir.

### S129. Android Native Development Kit (NDK)

NDK, Android ilovalarining performansini yaxshilash, maxsus funksionallikni qo'llab-quvvatlash va past darajadagi chiqishlar bilan ishlash uchun qo'llaniladi. Lekin shu bilan birga, Java'ga qaraganda murakkab bo'lgan C/C++ kodlarini yozish va yangilab turish talab etiladi.

Android Native Development Kit (NDK), Android ilovalarini yaratishda ishlatiladigan va C/C++ kodi ishga tushirish uchun taqdim etilgan qo'shimcha vosita. Bu, odatda, Android ilovachilari uchun Java yoki Kotlin tilida yozilgan ilovalar bilan bir qatorda ishlatiladi, lekin qo'shimcha, samarali va tezroq amalga oshirilishi kerak bo'lgan bazi qismlar C/C++ tilida yoziladi. NDK yordamida C/C++ kodi ishga tushirilishi yoki mavjud C/C++ kutubxonalaridan foydalanilishi mumkin.

Quyidagi vazifalarni amalga oshirish uchun NDK ishlatilishi mumkin:

1. Juda tez ishlovchi operatsiyalar : C/C++ tilida yozilgan kodi Java yoki Kotlin tilida yozilgan koddan tezroq ishga tushiriladi. Bunda, kiritilgan datalar, ma'lumotlar tahlili, matematik operatsiyalar va boshqa operatsiyalar tezroq ishlaydi.
2. Kiritilgan kodning to'liq ishlashi : Agar ma'lum bir algoritma yoki funktsiya C/C++ tilida yozilsa, NDK yordamida uni ishga tushirish mumkin. Bu, Android ilovalarining ish yuzini kengaytirish uchun qulay va samarali bo'lishi mumkin.
3. Xavfsizlik : C/C++ kodi jarayonining qisman yoki to'la qisman kriptografiya, shifrlash, bo'shlig'ini bo'sh qilish, xavfsizlik protokollari va boshqa xavfsizlik funktsiyalarini ishga tushirish uchun yaxshi yechimlar taqdim etadi.
4. Platforma xususiyatlari : Agar Androidning kuchli platforma xususiyatlari yoki platforma koddan foydalanish talablansagina, NDK yordamida bu talablarni amalga oshirish mumkin.
5. O'yinlar : O'yinlarning grafik tomonlari va fizikani juda ko'p ishchi kuchli xususiyatlar bo'lgani uchun, juda tezroq ishlovchi C/C++ kodining ishga tushirilishi talab qilinadi. NDK, o'yinlarni yaratishda muhim bir vosita bo'lib hisoblanadi.

NDK, ilovachilar uchun qo'llanishga mo'ljallangan daromad va tezlikni ta'minlash uchun mos keladigan muhim vosita hisoblanadi. Ammo uni ishlatish jarayonida, diqqat qaror qilinadigan muhim nuanslar mavjud.

### S130. JSON servisi

JSON servislari veb-ilovalarni, mobil ilovalarni va boshqa tizimlarni integratsiya qilish uchun keng qo'llaniladi. Ular tez, oson o'qiladigan va kross-platformatli ma'lumot almashinuvini ta'minlaydi.

JSON (JavaScript Object Notation) servisi, ma'lumotlar almashish uchun odatda ishlatiladigan bir protokol va formatdir. Bu, serverlar va mazmun provayderlar orasidagi ma'lumot almashishni amalga oshirish uchun o'rnatilgan sodda va oson yechimni ta'minlaydi. JSON servislari, dasturchilar va ilovachilar uchun yomon ko'rishadi va ma'lumotlarni almashish va qo'llash jarayonlarini simplifikatsiyalash imkonini beradi.

JSON servislari quyidagi vazifalarni amalga oshirish uchun ishlatilishi mumkin:

1. Ma'lumotlar almashish : JSON servislari, serverlardan ma'lumotlarni olishni oson va samarali qiladi. Bu, qidiruvlar, filtratsiyalar va boshqa parametrlar orqali ma'lumotlar almashishni ta'minlaydi.
2. Ma'lumotlarni yuklash : JSON servislari, mobil ilovalarga ma'lumotlarni yuborish imkonini beradi. Bu, foydalanuvchilar tomonidan kiritilgan ma'lumotlarni serverga yuborish uchun qo'llaniladi.
3. Ma'lumotlarni sinxronlashtirish : JSON servislari, mobil ilova va server o'rtasidagi ma'lumotlarni sinxronlashtirish imkonini beradi. Bu, foydalanuvchilar o'rtasidagi ma'lumotlarni to'g'ri va to'g'ri ishlashini ta'minlaydi.
4. Autentifikatsiya va xavfsizlik : JSON servislari, foydalanuvchilar o'zlarini autentifikatsiya qilish uchun serverga murojaat qilish uchun ishlatiladi. Bu, foydalanuvchilarning maxfiylik va xavfsizligini ta'minlash uchun muhimdir.
5. Ma'lumotlar bazasi muammosi : JSON servislari, ma'lumotlarni serverda saqlash, yangilash, o'chirish va boshqa ma'lumotlar bazasi operatsiyalarini amalga oshirish uchun ishlatiladi.

JSON servislari, mobil ilovalarni serverlar bilan bog'lash, ma'lumotlarni almashish va qo'llashni oson va samarali qilish uchun keng qo'llaniladi. Bu, dasturchilar va ilovachilar uchun ma'lumot almashish va qo'llash jarayonlarini simplifikatsiyalash uchun qulay va qulaydir.

## S131. Mobil ilovalar yaratish uchun dasturlash tillari

Mobil ilovalar yaratish uchun quyidagi asosiy dasturlash tillari va texnologiyalar ishlatiladi:

### 1. Native iOS (Objective-C, Swift):

- Objective-C va Swift - iOS ilovalarini yaratish uchun rasmiy Apple tomonidan qo'llab-quvvatlanadigan dasturlash tillari.
- Bu tillarda yozilgan kod to'g'ridan-to'g'ri iOS platformasida ishlaydi va iOS SDK ni to'liq qo'llab-quvvatlaydi.
- Kuchli natija beradi, lekin faqat iOS uchun ishlaydi.

### 2. Native Android (Java, Kotlin):

- Java va Kotlin - Android ilovalarini yaratish uchun rasmiy Google tomonidan qo'llab-quvvatlanadigan dasturlash tillari.
- Bu tillarda yozilgan kod to'g'ridan-to'g'ri Android platformasida ishlaydi va Android SDK ni to'liq qo'llab-quvvatlaydi.
- Kuchli natija beradi, lekin faqat Android uchun ishlaydi.

### 3. Cross-platform (React Native, Flutter, Xamarin):

- React Native - JavaScript/TypeScript asosida, iOS va Android uchun ilovalar yaratish.
- Flutter - Dart asosida, iOS va Android uchun ilovalar yaratish.
- Xamarin - C# asosida, iOS, Android va Windows uchun ilovalar yaratish.
- Ushbu yondashuvlar bir xil kod bazasini iOS, Android va boshqa platformalarda ishlashi uchun moslashtirishga imkon beradi.
- Platformadan mustaqil natija beradi, lekin native ilovalar qanchalik yaxshi bo'lmaydi.

### 4. Hybrid (Cordova, Ionic):

- Cordova va Ionic - HTML, CSS va JavaScript asosida, iOS, Android va boshqa platformalarda ishlatiladigan ilovalar yaratish.
- Web-texnologiyalar asosida ishlaydi va nativ SDK ni to'liq qo'llab-quvvatlamaydi.
- Tezligi va xususiyatlari past, lekin tez rivojlanadi.

Bundan tashqari, low-code va no-code yondashuvlar ham mavjud, masalan, Appian, OutSystems, Mendix kabi.

Qaysi dasturlash tili yoki texnologiyani tanlash - loyiha maqsadlari, jamoaning ko'nikmasi, bozor talablari va rivojlanish tezligiga bog'liq bo'ladi.

Mobil ilovalar yaratish uchun dasturlash tillari va muhitlar har qanday tilda ilovalar yaratish imkonini beradi, lekin eng mashhur va keng qo'llaniladiganlar quyidagilardir:

1. Java/Kotlin : Android ilovalarining asosiy dasturlash tillari Java va Kotlin hisoblanadi. Java Android platformasi uchun birinchi dasturlash tili sifatida qabul qilingan, ammo Kotlin ham o'zgartirishsiz yuqori darajadagi qabul qilingan dasturlash tili bo'ldi.
2. Swift/Objective-C : iOS ilovalarining dasturlash tillari Objective-C va Swift hisoblanadi. Swift, Objective-C'ga nisbatan yangi va sodda tilda yozilgan va hozirgi kunda iOS ilovalarni yaratishda keng qo'llaniladi.
3. JavaScript : JavaScript, mobil ilovalarni yaratish uchun qo'llaniladigan universal til hisoblanadi. Bu, React Native va Ionic kabi muhitlarda ishlatiladi va Android va iOS ilovalarini bir vaqtning o'zida yaratish imkonini beradi.
4. Dart : Dart tilida yozilgan Flutter muhiti, Android va iOS platformalari uchun kross-platformali ilovalarni yaratish uchun keng qo'llaniladi. Flutter, sodda, tez va qulay interfeyslar yaratish uchun xavfsiz, samarali dasturlash muhiti hisoblanadi.
5. C#/Xamarin : Microsoftning Xamarin platformasi, C# dasturlash tilida mobil ilovalarni yaratish uchun ishlatiladi. Bu, Android va iOS ilovalarini bir vaqtning o'zida yaratish imkonini beradi.
6. Python : Python tilida yozilgan ilovalar, Android uchun PyQT va iOS uchun Kivy kabi muhitlar orqali yaratilishi mumkin. Bu, bazi tilda yaratilgan mobil ilovalarni yaratish uchun foydalanish mumkin, ammo umuman olganda keng qo'llanilmaydi.

Har bir dasturlash tili va muhiti o'zining afzalliklari va cheklovlari bor, shuning uchun tanlashidan oldin, loyihangizning talablari va qobiliyatlaringizni qaror qiling. Odatda, ilgari tildagi tajribangiz, loyihangizning mahsulotlarini yaratish va taqdim etishda juda muhim bo'ladi.

## **S132. Mobil ilovalar uchun ma'lumotlar bazasi.**

Mobil ilovalar uchun ma'lumotlar bazasi sifatida quyidagi yechimlar keng qo'llaniladi:

### **1. Lokal ma'lumotlar bazasi**

- SQLite

- Mobil qurilmalarda keng tarqalgan, serverga ulanaolmaydigan, lokal ma'lumotlar bazasi.

- Oson o'rnatish, kichik hajm, tez ishlash.

- Masalan, iOS uchun Core Data, Android uchun Room.

- Realm

- Mobil ilovalar uchun moslashtirilgan, high-performance ma'lumotlar bazasi.

- iOS, Android, React Native, Flutter uchun qo'llaniladi.

- Tez ishlaydi, SDK ko'rinishida integratsiya qilish oson.

### **2. Bulutli ma'lumotlar bazasi**

- Firebase Firestore

- Google tomonidan taqdim etilgan, serverless bulutli ma'lumotlar bazasi.

- Realtime sinxronizatsiya, o'z-o'zini sozlash, masshtablanish.

- iOS, Android, Web uchun SDK mavjud.

- Amazon DynamoDB

- AWS tomonidan taqdim etilgan, serverless NoSQL ma'lumotlar bazasi.

- Yuksak masshtablanish, operatsion xarajatlari past.

- iOS, Android, Web uchun SDK mavjud.

### **3. Hybrid yondashuvlar**

- Lokal SQLite + Bulutli ma'lumotlar bazasi

- Tez ishlash uchun lokal ma'lumotlarni saqlash, bulutga sinxronizatsiya qilish.

- Masalan, iOS uchun Realm + Firebase, Android uchun Room + Firebase.

- Bulutli ma'lumotlar bazasi + CDN

- Statik mazmunni CDN orqali yetkazish, dinamik ma'lumotlarni bulutli ma'lumotlar bazasida saqlash.

- Masalan, Firestore + Firebase Hosting, DynamoDB + Amazon S3.

Mobil ilovalarning ma'lumotlar bazasi tanlovi - ma'lumotlar hajmi, ishlash tezligi, sinxronizatsiya, masshtablanish, xavfsizlik kabi omillarga bog'liq bo'ladi. Lokal va bulutli ma'lumotlar bazalarining kombinatsiyasi ko'p hollarda maqbul yechim bo'lib xizmat qiladi.

Mobil ilovalar uchun ma'lumotlar bazasi (MDB), ilova ichidagi ma'lumotlarni saqlash va boshqarish uchun muhim bir qisimdir. MDB, ilova foydalanuvchilari, sohalari va xizmatlari haqida ma'lumotlarni saqlash uchun ishlatiladi va ilova uchun boshqa ma'lumotlar bazasi serverlariga o'rnatilishi mumkin. Quyidagi xususiyatlar MDBning muhim qismlari hisoblanadi:

1. Ma'lumotlar bazasining turi : MDB turini tanlashda, ilova uchun kerak bo'lgan ma'lumotlar turi va ma'lumotlarni saqlash uchun qo'llaniladigan tilda yozilgan dasturlash tili kritik muhimlikka ega bo'ladi. Ba'zi mashhur MDB turkumlari SQLite, MySQL, PostgreSQL, MongoDB kabi.
2. Ma'lumotlar saqlash usullari : MDB ichidagi ma'lumotlar, odatda, moslashtirilgan ma'lumotlar bazasi turi orqali saqlanadi. Ba'zi ilovalar boshqaruv tizimlari kerak bo'lishi mumkin, masalan, fayllar, rasmlar va boshqa kontentlarni saqlash uchun Amazon S3, Google Cloud Storage, Firebase Storage kabi.
3. Foydalanuvchilar ma'lumotlari : MDB, foydalanuvchilar ma'lumotlarini saqlash uchun ham ishlatiladi. Bu, foydalanuvchilar profil ma'lumotlari, autentifikatsiya ma'lumotlari, ma'lumotlar tarixi va boshqa shaxsiy ma'lumotlarni saqlash uchun foydalaniladi.
4. Ma'lumotlar sinxronlashtirish : MDB, mobil ilova va server o'rtasidagi ma'lumotlarni sinxronlashtirish uchun ishlatilishi mumkin. Bu, foydalanuvchilar qo'llab-quvvatlashini, o'rtadagi ilovalar orasidagi ma'lumot almashishni va serverga ma'lumotlarni yuklash va saqlashni osonlashtiradi.
5. Maxfiylik va xavfsizlik : MDB ma'lumotlari, ilovaning maxfiylik va xavfsizligini ta'minlash uchun muhimdir. Foydalanuvchilar ma'lumotlarining himoyalanganligi, shifrlanishi va foydalanuvchilar uchun maxfiylik sozlamalari kritik muhimlikka ega bo'ladi.



MDB, mobil ilovalar uchun muhim tarkibiy qismni tashkil etadi va ilova o'rtasidagi aloqa, ma'lumot almashish va ma'lumotlar saqlash jarayonlarini tuzishda katta rol o'ynaydi. Raqamli aloqa, maxfiylik va xavfsizlikning yaxshi aniqlanishi va foydalanuvchilar uchun yaxshi tajribani ta'minlash MDBning muhim vazifalaridan ba'zilaridir.

### 133. SQLite haqida umumiy ma'lumot.

SQLite haqida quyidagi umumiy ma'lumotlarni keltirib o'tamiz:

1. Tavsifi : SQLite - serverga ulanish talabini qo'ymaydigan, kichik hajmli, oson qo'llanadigan, to'liq SQL-ga mos keladigan ma'lumotlar bazasi boshqarish tizimi (DBMS).
2. Xususiyatlari :
  - Serverga muhtoj emas : SQLite ma'lumotlar bazasi fayllar ko'rinishida saqlanadigan, server jarayonini talab qilmaydi.
  - Kichik hajm : SQLite ta'sis etilgan DBMS orasida eng kichik hajmli (2-3 MB).
  - Tez ishlash : Ma'lumotlar bazasini fayllar tizimida saqlanishi tufayli juda tez ishlaydi.
  - Ishonchli : Ko'p yillar davomida ishlab chiqilgan, qo'llab-quvvatlanadi.
  - Bepul va ochiq kodli : SQLite bepul va ochiq kodli loyiha bo'lib, istalgan maqsad uchun foydalanish mumkin.
3. Ishlash prinsipi : SQLite ma'lumotlar bazasi ma'lumotlar va sxemalarni ma'lumotlar bazasi faylida saqlab, mustaqil ravishda ishlaydi. Foydalanuvchi ilovasi SQLite kutubxonasini chaqirish orqali ma'lumotlar bazasi bilan o'zaro aloqada bo'ladi.
4. Qo'llanishi : SQLite DBMS ko'pgina operatsion tizimlar (Windows, Linux, macOS) va mobil qurilmalar (iOS, Android) uchun qo'llab-quvvatlanadi. Asosan, mobil ilovalar, internet-narsalar, o'rnatilgan tizimlar va boshqa kichik hajmli loyihalar uchun ishlatiladigan ma'lumotlar bazasi hisoblanadi.
5. Afzalliklari : Soddaligi, kichik hajmi, tezligi, bepulligidan tashqari, ACID (atomarlik, bir xillik, izolyatsiya, barqarorlik) xususiyatlariga ham ega.

SQLite ma'lumotlar bazasidan foydalanish, ma'lumotlarni saqlash, boshqarish va ishlash uchun yaxshi yechim bo'lib xizmat qiladi, ayniqsa mobil ilovalar uchun maqbul hisoblanadi.

### S133. SQLite haqida umumiy ma'lumot

SQLite — bu eng keng tarqalgan va ommabop ma'lumotlar bazasi (database) menejeri (DBMS) bo'lib, u ichki dasturiy ta'minot sifatida foydalanish uchun mo'ljallangan. U boshqa ko'plab ma'lumotlar bazalaridan farqli o'laroq, serverga ehtiyoj sezmaydi, ya'ni u "serverless" arxitekturaga ega.

#### Asosiy Xususiyatlari

##### 1. Yengilligi va Mustaqilligi :

- SQLite juda yengil bo'lib, o'zida faqat bitta faylni (odatda `.sqlite`` yoki `.db`` kengaytmasi bilan) saqlaydi. Bu fayl butun ma'lumotlar bazasini o'z ichiga oladi.

- Har qanday operatsion tizimda (Windows, macOS, Linux, Android, iOS va boshqalar) ishlaydi.

##### 2. Serverless :

- Ma'lumotlar bazasi boshqarish tizimi (DBMS) sifatida, u serverni ishga tushirishni talab qilmaydi. Barcha operatsiyalar to'g'ridan-to'g'ri fayl tizimida amalga oshiriladi.

##### 3. O'z-o'zini tashkil qiluvchi va Konfiguratsiyasiz :

- SQLite hech qanday maxsus o'rnatishni yoki konfiguratsiyani talab qilmaydi. U dasturiy ta'minot bilan birga kelishi mumkin va tayyor holda foydalanishga kirishadi.

##### 4. Kuchli SQL qo'llab-quvvatlashi :

- SQLite SQL-92 standartiga mos keladi va ko'pchilik SQL buyruqlarini qo'llab-quvvatlaydi, shu jumladan JOIN, subqueries, triggers, views va boshqalar.

##### 5. Yuqori Ishonchlilik :

- SQLite ACID (Atomicity, Consistency, Isolation, Durability) xususiyatlarini qo'llab-quvvatlaydi, bu esa tranzaksiyalarni xavfsiz va ishonchli qiladi.

##### 6. Zaxira va Ko'chirish :

- Ma'lumotlar bazasi bitta fayl sifatida saqlanadi, shuning uchun uni ko'chirish va zaxiralash juda oson. Faylni boshqa joyga ko'chirish orqali ma'lumotlar bazasini ko'chirishingiz mumkin.

#### Qo'llanilishi

SQLite turli xil sohalarda keng qo'llaniladi, jumladan:

- Mobil Dasturlar : Android va iOS uchun mo'ljallangan dasturlarda ma'lumotlar saqlash uchun.
- Veb Brauzerlar : Google Chrome va Mozilla Firefox kabi brauzerlar foydalanuvchi sozlamalari va tarixini saqlashda foydalanadi.
- Veb-illovalar : Yengil va kichik hajmli ma'lumotlar bazasi sifatida.
- IoT Qurilmalar : Kam quvvat sarfi va kichik xotira hajmi talab etiladigan qurilmalarda.

#### Afzalliklari va Kamchiliklari

##### Afzalliklari:

- Juda kam resurs talab qiladi.
- Integratsiya qilish va ishlatish oson.
- Zaxiralash va ma'lumotlarni ko'chirish oson.
- Keng platforma va dasturlash tillari qo'llab-quvvatlovi.

##### Kamchiliklari:

- Juda katta hajmdagi yoki yuqori darajada yuklangan ma'lumotlar bazalari uchun mos emas.
- Ko'p foydalanuvchi muhitlari uchun yaxshi optimallashtirilmagan.
- Klasterlash va taqsimlangan tizimlarni qo'llab-quvvatlamaydi.

SQLite, o'zining yengilligi va qulayligi tufayli, juda ko'p dasturlar va tizimlar uchun ma'lumotlar bazasi sifatida tanlanadi. U keng qamrovli imkoniyatlarga ega bo'lib, ko'p hollarda mukammal yechim bo'lib xizmat qiladi.

SQLite haqida qisqa ma'lumot:

SQLite - bu umumiy qo'llanish rivojlangan, xotiradan foydalanish, SQL-ga asoslangan, yengil ma'lumotlar ombori tizimi. U 2000-yilda yaratilgan va hozirda dunyo bo'ylab juda keng tarqalgan.

Asosiy xususiyatlari:

- Yengil va faqat bir foydalanuvchili: SQLite ma'lumotlar bazalari faqatgina bir vaqtda bir nechta foydalanuvchiga xizmat qilishi mumkin. Bu uni yengil va tezkor qiladi.

- Serverga bog'liq emas: SQLite ma'lumotlar bazasi ma'lumotlarni serverda emas, balki foydalanuvchi qurilmasida saqlaydi. Bu uni tezkor, sodda va mustaqil qiladi.
- Krossplatforma: SQLite turli operatsion tizimlar (Windows, Linux, macOS) uchun mavjud.
- Bepul va ochiq kodli: SQLite bepul va ochiq kodli dasturiy ta'minotdir, bu uni ommabop va keng qo'llanilishiga olib keladi.
- ACID-ga moslashgan: SQLite ma'lumotlar bazasi ACID (Atomcharlik, Konsistentlik, Izolyatsiya, Barqarorlik) printsiplarini qo'llab-quvvatlaydi.

SQLite juda keng tarqalgan va turli dasturiy ta'minotda, masalan, mobil ilovalar, veb-saytlar, elektron jadvallarda, kichik loyihalarda keng qo'llaniladi.

### S134. SQLite so'rovlari: ma'lumotlarni qo'shish va ko'rish

SQLite so'rovlari yordamida ma'lumotlarni qo'shish va ko'rish juda sodda va oson. Quyida ma'lumotlarni qo'shish (`INSERT`) va ko'rish (`SELECT`) uchun asosiy SQL so'rovlari keltirilgan.

#### Ma'lumotlar Bazasini Yaratish

Avval ma'lumotlar bazasini yaratish va jadval (table) yaratish kerak bo'ladi. Masalan, biz `students` nomli jadvalni yaratamiz.

```
```sql
-- Ma'lumotlar bazasini yaratish va ulanish
sqlite3 mydatabase.db

-- Jadval yaratish
CREATE TABLE students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER NOT NULL,
    grade TEXT NOT NULL
);
```
```

#### Ma'lumot Qo'shish

Jadvalga ma'lumot qo'shish uchun `INSERT INTO` so'rovini ishlatamiz.

```
```sql
-- Ma'lumot qo'shish
INSERT INTO students (name, age, grade) VALUES ('Alice', 20, 'A');
INSERT INTO students (name, age, grade) VALUES ('Bob', 22, 'B');
INSERT INTO students (name, age, grade) VALUES ('Charlie', 23, 'C');
```
```

## Ma'lumotni Ko'rish

Jadvaldagi ma'lumotlarni ko'rish uchun `SELECT` so'rovini ishlatamiz.

```
```sql
```

```
-- Barcha ma'lumotlarni ko'rish
```

```
SELECT * FROM students;
```

```
-- Faqat ma'lum ustunlarni ko'rish
```

```
SELECT name, grade FROM students;
```

```
-- Shartli so'rov: yosh 22 dan katta bo'lgan talabalarning ma'lumotlarini ko'rish
```

```
SELECT * FROM students WHERE age > 22;
```

```
-- Tartiblangan so'rov: yosh bo'yicha o'sish tartibida tartiblash
```

```
SELECT * FROM students ORDER BY age ASC;
```

```
```
```

## SQL So'rovlarining Misollari

### 1. Barcha Ma'lumotlarni Ko'rish

```
```sql
```

```
SELECT * FROM students;
```

```
```
```

Bu so'rov barcha talabalarni va ularning barcha ma'lumotlarini ko'rsatadi.

### 2. Ma'lum Ustunlarni Ko'rish

```
```sql
```

```
SELECT name, grade FROM students;
```

```
```
```

Bu so'rov faqat talabalarning ismlari va ularning baholarini ko'rsatadi.

### 3. Shartli So'rov

```
```sql
```

```
SELECT * FROM students WHERE age > 22;
```

```
```
```

Bu so'rov yosh 22 dan katta bo'lgan talabalarning ma'lumotlarini ko'rsatadi.

#### 4. Tartiblangan So'rov

```
```sql
```

```
SELECT * FROM students ORDER BY age ASC;
```

```
```
```

Bu so'rov talabalarning ma'lumotlarini yosh bo'yicha o'sish tartibida tartiblaydi.

#### Ma'lumotlarni Yangilash va O'chirish

Qo'shimcha ravishda, ma'lumotlarni yangilash (`UPDATE`) va o'chirish (`DELETE`) uchun quyidagi so'rovlar ishlatiladi:

##### Ma'lumotni Yangilash

```
```sql
```

```
-- Ma'lumotni yangilash
```

```
UPDATE students SET age = 21 WHERE name = 'Alice';
```

```
```
```

##### Ma'lumotni O'chirish

```
```sql
```

```
-- Ma'lumotni o'chirish
```

```
DELETE FROM students WHERE name = 'Bob';
```

```
```
```

Bu misollar SQLite bilan ishlashda boshlang'ich qadamlarni tushunishga yordam beradi. SQLite so'rovlari oddiy va oson bo'lib, ular yordamida juda ko'p funksiyalarni bajarish mumkin.

### S135. SQLite so'rovlari: ma'lumotlarni tahrirlash va o'chirish

SQLite so'rovlari yordamida ma'lumotlarni tahrirlash (`UPDATE`) va o'chirish (`DELETE`) juda oddiy. Quyida ushbu operatsiyalarni qanday amalga oshirish ko'rsatilgan.

#### Ma'lumotlarni Tahrirlash

Ma'lumotlarni tahrirlash uchun `UPDATE` so'rovini ishlatamiz. Ushbu so'rov yordamida jadvaldagi mavjud ma'lumotlarni yangilash mumkin.

# Sintaksis

```
```sql
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```
```

# Misollar

#### 1. Talabani Yoshini Yangilash

Faraz qilaylik, `students` jadvalida `Alice` ismli talabani topib, uning yoshini 21 ga o'zgartirmoqchisiz:

```
```sql
UPDATE students
SET age = 21
WHERE name = 'Alice';
```
```

#### 2. Talabani Bahosini Yangilash

`Charlie` ismli talabani bahosini `B` ga o'zgartirish:

```
```sql
UPDATE students
```



```
SET grade = 'B'
```

```
WHERE name = 'Charlie';
```

```
---
```

3. Bir Nechta Ustunlarni Yangilash

`Bob` ismli talabaning yoshini 23 ga va bahosini `A` ga o'zgartirish:

```
```sql
```

```
UPDATE students
```

```
SET age = 23, grade = 'A'
```

```
WHERE name = 'Bob';
```

```

```

### Ma'lumotlarni O'chirish

Ma'lumotlarni o'chirish uchun `DELETE` so'rovini ishlatamiz. Ushbu so'rov yordamida jadvaldagi mavjud ma'lumotlarni o'chirish mumkin.

#### # Sintaksis

```
```sql
```

```
DELETE FROM table_name
```

```
WHERE condition;
```

```
---
```

Misollar

1. Talabani O'chirish

Faraz qilaylik, `students` jadvalida `Bob` ismli talabani o'chirmoqchisiz:

```
```sql
```

```
DELETE FROM students
```

```
WHERE name = 'Bob';
```

```

```

##### 2. Ma'lum Yoshdan Katta Bo'lgan Talabalarni O'chirish

Yoshi 22 dan katta bo'lgan barcha talabalarni o'chirish:

```
```sql
```

```
DELETE FROM students
```

```
WHERE age > 22;
```

```
```
```

Shartlarsiz `DELETE` va `UPDATE` dan Ehtiyot Bo'ling

Agar `WHERE` sharti qo'llanilmasa, barcha ma'lumotlar tahrirlanadi yoki o'chiriladi:

# Barcha Jadvalni Tozalash

Jadvaldagi barcha ma'lumotlarni o'chirish uchun (jadval strukturasi saqlanib qoladi):

```
```sql
```

```
DELETE FROM students;
```

```
```
```

# Barcha Ma'lumotlarni Yangilash

Jadvaldagi barcha yozuvlarning bir ustunini yangilash:

```
```sql
```

```
UPDATE students
```

```
SET grade = 'F';
```

```
```
```

Bu so'rov `students` jadvalidagi barcha yozuvlarning `grade` ustunini `F` ga o'zgartiradi.

SQLite'da ma'lumotlarni tahrirlash va o'chirish juda oson va qulay. `UPDATE` so'rovi yordamida jadvaldagi ma'lumotlarni yangilash, `DELETE` so'rovi yordamida esa ma'lumotlarni o'chirish mumkin. Har doim ehtiyot bo'lish lozim, chunki `WHERE` shartsiz yozilgan so'rovlar barcha yozuvlarga ta'sir qilishi mumkin.

## S137. SQLite haqida umumiy ma'lumot.

SQLite — bu eng keng tarqalgan va ommabop ma'lumotlar bazasi menejeri (DBMS) bo'lib, serverga ehtiyoj sezmasdan ishlaydi va asosan ichki dasturiy ta'minot sifatida foydalanish uchun mo'ljallangan. U "serverless" arxitekturaga ega va bir nechta sabablarga ko'ra juda ommalashgan.

### Asosiy Xususiyatlari

#### 1. Yengilligi va Mustaqilligi :

- SQLite juda yengil bo'lib, o'zida faqat bitta faylni (odatda `.sqlite`` yoki `.db`` kengaytmasi bilan) saqlaydi. Bu fayl butun ma'lumotlar bazasini o'z ichiga oladi.

- Har qanday operatsion tizimda (Windows, macOS, Linux, Android, iOS va boshqalar) ishlaydi.

#### 2. Serverless :

- Ma'lumotlar bazasi boshqarish tizimi (DBMS) sifatida, u serverni ishga tushirishni talab qilmaydi. Barcha operatsiyalar to'g'ridan-to'g'ri fayl tizimida amalga oshiriladi.

#### 3. O'z-o'zini tashkil qiluvchi va Konfiguratsiyasiz :

- SQLite hech qanday maxsus o'rnatishni yoki konfiguratsiyani talab qilmaydi. U dasturiy ta'minot bilan birga kelishi mumkin va tayyor holda foydalanishga kirishadi.

#### 4. Kuchli SQL Qo'llab-quvvatlashi :

- SQLite SQL-92 standartiga mos keladi va ko'pchilik SQL buyruqlarini qo'llab-quvvatlaydi, shu jumladan JOIN, subqueries, triggers, views va boshqalar.

#### 5. Yuqori Ishonchlilik :

- SQLite ACID (Atomicity, Consistency, Isolation, Durability) xususiyatlarini qo'llab-quvvatlaydi, bu esa tranzaksiyalarni xavfsiz va ishonchli qiladi.

#### 6. Zaxira va Ko'chirish :

- Ma'lumotlar bazasi bitta fayl sifatida saqlanadi, shuning uchun uni ko'chirish va zaxiralash juda oson. Faylni boshqa joyga ko'chirish orqali ma'lumotlar bazasini ko'chirishingiz mumkin.

### Qo'llanilishi

SQLite turli xil sohalarda keng qo'llaniladi, jumladan:

- Mobil Dasturlar : Android va iOS uchun mo'ljallangan dasturlarda ma'lumotlar saqlash uchun.
- Veb Brauzerlar : Google Chrome va Mozilla Firefox kabi brauzerlar foydalanuvchi sozlamalari va tarixini saqlashda foydalanadi.
- Veb-ilovalar : Yengil va kichik hajmli ma'lumotlar bazasi sifatida.
- IoT Qurilmalar : Kam quvvat sarfi va kichik xotira hajmi talab etiladigan qurilmalarda.

#### Afzalliklari va Kamchiliklari

##### Afzalliklari:

- Juda kam resurs talab qiladi.
- Integratsiya qilish va ishlatish oson.
- Zaxiralash va ma'lumotlarni ko'chirish oson.
- Keng platforma va dasturlash tillari qo'llab-quvvatlovi.

##### Kamchiliklari:

- Juda katta hajmdagi yoki yuqori darajada yuklangan ma'lumotlar bazalari uchun mos emas.
- Ko'p foydalanuvchi muhitlari uchun yaxshi optimallashtirilmagan.
- Klasterlash va taqsimlangan tizimlarni qo'llab-quvvatlamaydi.

#### Ma'lumot Qo'shish va Ko'rish

##### # Ma'lumot Qo'shish

Jadvalga ma'lumot qo'shish uchun `INSERT INTO` so'rovini ishlatamiz.

```
```sql
```

```
INSERT INTO table_name (column1, column2, column3, ...)
```

```
VALUES (value1, value2, value3, ...);
```

```
```
```

Misol:

```
```sql
```

```
INSERT INTO students (name, age, grade) VALUES ('Alice', 20, 'A');
```

```
```
```

# Ma'lumotni Ko'rish

Jadvaldagi ma'lumotlarni ko'rish uchun `SELECT` so'rovini ishlatamiz.

```
```sql
```

```
SELECT column1, column2, ...
```

```
FROM table_name
```

```
WHERE condition
```

```
ORDER BY column1, column2, ...;
```

```
```
```

### S138. SQLite so'rovlari: ma'lumotlarni qo'shish va ko'rish

SQLite ma'lumotlarni qo'shish va ko'rish uchun asosiy so'rovlar:

1. Ma'lumotlarni qo'shish (INSERT):

```
```sql
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```
```

Masalan, "users" jadvaliga yangi foydalanuvchi qo'shish:

```
```sql
INSERT INTO users (name, email, age)
VALUES ('John Doe', 'john@example.com', 35);
```
```

Agar jadvaldagi barcha ustunlarga qiymatlar berilsa, ustunlar ro'yxatini yozmasdan ham qo'yish mumkin:

```
```sql
INSERT INTO users
VALUES (1, 'Jane Smith', 'jane@example.com', 28);
```
```

2. Ma'lumotlarni ko'rish (SELECT):

Asosiy sintaksis:

```
```sql
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```
```

Masalan, "users" jadvalidan barcha foydalanuvchilarni ko'rish:

```
```sql
```

```
SELECT * FROM users;
```

```
```
```

Faqat ayrim ustunlarni ko'rish:

```
```sql
```

```
SELECT name, email FROM users;
```

```
```
```

Shart bilan foydalanuvchini filtrlash:

```
```sql
```

```
SELECT * FROM users
```

```
WHERE id = 1;
```

```
```
```

Natijalarni tartiblash:

```
```sql
```

```
SELECT * FROM users
```

```
ORDER BY name ASC;
```

```
```
```

`SELECT` so'rovi boshqa SQL amallar (WHERE, ORDER BY, GROUP BY, JOIN va boshqalar) bilan ham biriktirilishi mumkin. Bu ma'lumotlarni tanlab olish uchun keng imkoniyatlar yaratadi.

Qo'shish va ko'rish amallarini bajarish uchun SQLite keng qo'llaniladi, chunki ular ma'lumotlar bilan ishlashning asosiy operatsiyalaridir

SQLite so'rovlari yordamida ma'lumotlarni qo'shish va ko'rish juda sodda va oson. Quyida ma'lumotlarni qo'shish (`INSERT`) va ko'rish (`SELECT`) uchun asosiy SQL so'rovlari keltirilgan.

Ma'lumot Qo'shish (INSERT INTO)

Jadvalga ma'lumot qo'shish uchun `INSERT INTO` so'rovini ishlatamiz.

# Sintaksis

```
```sql
```

```
INSERT INTO table_name (column1, column2, column3, ...)
```

```
VALUES (value1, value2, value3, ...);
```

```
```
```

# Misol

Faraz qilaylik, bizda `students` nomli jadval mavjud bo'lib, unda `name`, `age`, va `grade` ustunlari bor. Talabalarni qo'shish uchun quyidagi so'rovni ishlatamiz:

```
```sql
```

```
INSERT INTO students (name, age, grade) VALUES ('Alice', 20, 'A');
```

```
INSERT INTO students (name, age, grade) VALUES ('Bob', 22, 'B');
```

```
INSERT INTO students (name, age, grade) VALUES ('Charlie', 23, 'C');
```

```
```
```

Ma'lumotni Ko'rish (SELECT)

Jadvaldagi ma'lumotlarni ko'rish uchun `SELECT` so'rovini ishlatamiz.

# Sintaksis

```
```sql
```

```
SELECT column1, column2, ...
```

```
FROM table_name
```

```
WHERE condition
```

```
ORDER BY column1, column2, ...;
```

```
```
```

# Misollar

1. Barcha Ma'lumotlarni Ko'rish

```
```sql
```



```
SELECT * FROM students;
```

```
---
```

Bu so'rov `students` jadvalidagi barcha yozuvlarni ko'rsatadi.

S139. ROOM ma'lumotlar bazasidagi yozuvlar bo'yicha harakatlar bilan ishlash

Room kutubxonasi yordamida ma'lumotlar bazasi bilan ishlash Android ilovalarida juda qulay va samarali. Quyida Room yordamida yozuvlar bo'yicha qo'shish, yangilash, o'chirish va ko'rish operatsiyalari qanday amalga oshirilishini ko'rsataman.

ROOM ma'lumotlar bazasidagi yozuvlar bo'yicha harakatlar bilan ishlash uchun quyidagi amallarni bajarishingiz mumkin:

1. Yozuv qo'shish (INSERT) :

```
```kotlin  

val user = User(name = "John Doe", email = "john@example.com", age = 35)

db.userDao().insert(user)
```
```

2. Yozuvlarni o'qish (SELECT) :

```
```kotlin  

val users = db.userDao().getAllUsers()
```
```

3. Shart orqali yozuvlarni o'qish (SELECT) :

```
```kotlin  

val user = db.userDao().getUserById(userId)
```
```

4. Yozuvni yangilash (UPDATE) :

```
```kotlin  

user.name = "New Name"

db.userDao().update(user)
```
```

5. Yozuvni o'chirish (DELETE) :

```
```kotlin  

db.userDao().delete(user)
```

...

Yuqoridagi kodlar "user" jadvali uchun yozuvlar bilan ishlashni namoyish etadi. `db` o'zgaruvchisi `AppDatabase` dan olindi, va `userDao()` metodi orqali `UserDao` obykti olish uchun ishlatildi.

`UserDao` interfeysining funksiyalari yozuvlar bilan ishlash uchun ishlatiladi. Ma'lumotlarni o'qish, qo'shish, yangilash va o'chirish amallarini amalga oshirish uchun bu funksiyalardan foydalanishingiz mumkin.

Bu misollar yalpi ma'lumotlarni qo'llab-quvvatlaydi. Ba'zi holatlarda transaksiya, shartlar, limitlar va boshqalar bilan kompleks so'rovlar ham ishlatishingiz mumkin.

## S140. Xabar almashish. SMS va MMS yuborish

Xabar almashish, SMS va MMS yuborish uchun Android platformasida quyidagi yondashuvlardan foydalanishingiz mumkin:

1. SMS (Short Message Service) yuborish :

```
```kotlin
val smsManager = SmsManager.getDefault()
smsManager.sendTextMessage(
    "+1234567890", // recipient phone number
    null, // service center address (optional)
    "Hello, this is a test SMS message.", // message text
    null, // pending intent (optional)
    null // delivery report intent (optional)
)
```
```

2. MMS (Multimedia Messaging Service) yuborish :

```
```kotlin
val context = applicationContext
val sendIntent = Intent(Intent.ACTION_SEND)
sendIntent.type = "image/jpeg" // or any other MIME type
// Attach image to the intent
val imageUri = Uri.parse("file:///path/to/image.jpg")
sendIntent.putExtra(Intent.EXTRA_STREAM, imageUri)
// Add message text
sendIntent.putExtra(Intent.EXTRA_TEXT, "Hello, this is a test MMS message.")
// Start activity to send MMS
context.startActivity(Intent.createChooser(sendIntent, "Share Image"))
```
```

### 3. Xabar qabul qilish :

```
```kotlin
```

```
class SmsReceiver : BroadcastReceiver() {  
    override fun onReceive(context: Context, intent: Intent) {  
        if (intent.action == Telephony.Sms.Intents.SMS_RECEIVED_ACTION) {  
            val bundle = intent.extras  
            if (bundle != null) {  
                val pdus = bundle.get("pdus") as Array<Any>?  
                val messages: Array<SmsMessage?> = arrayOfNulls(pdus!!.size)  
                for (i in pdus.indices) {  
                    messages[i] = SmsMessage.createFromPdu(pdus[i] as ByteArray)  
                }  
  
                for (smsMessage in messages) {  
                    val messageBody = smsMessage?.messageBody  
                    val senderNumber = smsMessage?.originatingAddress  
                    // Process received SMS message  
                    // ...  
                }  
            }  
        }  
    }  
}
```

Ushbu kodlar SMS va MMS xabarlarini yuborish uchun foydalanish mumkin. Xabar qabul qilishda esa `BroadcastReceiver` orqali `SMS_RECEIVED_ACTION` bronyasini qabul qilib, kerakli amallarni bajarish mumkin.

Eslatmalar:

- SMS/MMS funksiyalaridan foydalanish uchun `SEND_SMS` va `READ_SMS` ruxsatnomalari kerak.
- Xabarlarni yuborish va qabul qilish uchun zarur ruxsatnomalarga ega ekanligi tekshirilishi lozim.
- Xabarlarni saqlash, ko'rsatish, boshqarish kabi qo'shimcha amallar dasturda amalga oshirilishi kerak.

Yuqoridagi misollar yordamida Android ilovasi orqali SMS va MMS xabarlarini qanday yuborishni ko'rdik. `SmsManager` yordamida SMS yuborish juda oddiy va tez amalga oshiriladi. MMS yuborish esa `Intent` orqali amalga oshiriladi va bu foydalanuvchining harakatlarini talab qiladi. MMS uchun to'liq fon jarayonida ishlashni xohlasangiz, qo'shimcha kutubxonalar yoki xizmatlar yordamida amalga oshirishingiz mumkin.

S141. Mobil ilovalarda tarmoqli kengligi. HttpURLConnection sinfidan foydalanish

Mobil ilovalarda tarmoqli kengligi (HTTP) bilan ishlash uchun qaysi versiyadan foydalanish kerakligi mobil ilovangizning Android yoki iOS platformasiga bog'liq.

1. Android uchun :

Android platformasida tarmoqli kengliklar uchun `HttpURLConnection` sinfi yoqiladi. Quyidagi misol `HttpURLConnection` orqali GET so'rovini yuborishni namoyish etadi:

```
```kotlin
```

```
import java.io.BufferedReader
```

```
import java.io.IOException
```

```
import java.io.InputStreamReader
```

```
import java.net.HttpURLConnection
```

```
import java.net.URL
```

```
class MainActivity : AppCompatActivity() {
```

```
 override fun onCreate(savedInstanceState: Bundle?) {
```

```
 super.onCreate(savedInstanceState)
```

```
 setContentView(R.layout.activity_main)
```

```
 Thread {
```

```
 val url = URL("https://api.example.com/data")
```

```
 val connection = url.openConnection() as HttpURLConnection
```

```
 connection.requestMethod = "GET"
```

```
 try {
```

```
 val responseCode = connection.responseCode
```

```
 if (responseCode == HttpURLConnection.HTTP_OK) {
```

```

 val inputStream = connection.inputStream
 val reader = BufferedReader(InputStreamReader(inputStream))
 val response = StringBuilder()
 var line: String?
 while (reader.readLine().also { line = it } != null) {
 response.append(line)
 }
 reader.close()

 // Response handling
 val responseData = response.toString()
 runOnUiThread {
 // Update UI with the response data
 }
 }
} catch (e: IOException) {
 e.printStackTrace()
} finally {
 connection.disconnect()
}
}.start()
}
}

```

2. iOS uchun :iOS platformasida Applening `NSURLSession` va `URLRequest` sinflaridan foydalaniladi. Quyidagi Swift misol `NSURLSession` orqali GET so'rovini yuborishni namoyish etadi:



```

```swift
import UIKit

class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()

        let url = URL(string: "https://api.example.com/data")!
        let task = URLSession.shared.dataTask(with: url) { (data, response, error) in
            if let error = error {
                print("Error: \(error)")
            } else if let data = data {
                let responseString = String(data: data, encoding: .utf8)

                // Response handling
                DispatchQueue.main.async {
                    // Update UI with the response data
                }
            }
        }
        task.resume()
    }
}
```

```

Bu misollar GET so'rovini yuborish uchun `URLConnection` va `URLSession` sinflaridan foydalanishni namoyish etadi. Ba'zi holatlarda POST, PUT, DELETE so'rovlari ham amalga oshirish mumkin, ulardan foydalanish uchun qo'shimcha konfiguratsiya kerak bo'ladi.

Muhim e'tibor: Tarmoqli kenglikda ma'lumotlar yuborish va qabul qilishning amaliyotlari tez-tez alohida potoklarda (thread) bajarilishi lozim, shuning uchun bu misolda to'plangan. Shuningdek, tarmoqli kenglikda amalga oshiriladigan so'rovlar uchun internetga kirish ruxsatnomasining berilishi kerak.

Mobil ilovalarda tarmoqli kenglikni (network bandwidth) samarali foydalanish juda muhimdir. Tarmoqli kenglikni boshqarish va samarali ishlatish orqali foydalanuvchi tajribasini yaxshilash mumkin. HTTP so'rovlarini amalga oshirish uchun Android platformasida bir nechta usullar mavjud bo'lib, ulardan biri `URLConnection` sinfidir.

`URLConnection` sinfidan foydalanish

`URLConnection` sinfi HTTP so'rovlarini amalga oshirish uchun juda qulay vosita bo'lib, u HTTP protokolini qo'llab-quvvatlaydi va foydalanuvchiga HTTP orqali ma'lumotlarni yuborish va olish imkonini beradi.

## # 1. Internet Ruxsatini Qo'shish

`AndroidManifest.xml` fayliga internetga ulanish ruxsatini qo'shish kerak:

```
```xml
<uses-permission android:name="android.permission.INTERNET"/>
```
```

## # 2. Asosiy `URLConnection` Foydalanish Misoli

Quyida, `URLConnection` sinfidan foydalanib HTTP GET va POST so'rovlarini amalga oshirish bo'yicha asosiy misollar keltirilgan.

### ## GET So'rovi

```
```kotlin
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
```

```
import java.net.URL
import java.lang.StringBuilder
import android.os.AsyncTask

class GetRequestTask : AsyncTask<String, Void, String>() {
    override fun doInBackground(vararg params: String?): String {
        val urlString = params[0]
        var result = ""
        val url = URL(urlString)
        val connection = url.openConnection() as HttpURLConnection

        try {
            connection.requestMethod = "GET"
            val inputStream = connection.inputStream
            val reader = BufferedReader(InputStreamReader(inputStream))
            val stringBuilder = StringBuilder()
            var line: String?

            while (reader.readLine().also { line = it } != null) {
                stringBuilder.append(line).append("\n")
            }

            reader.close()
            result = stringBuilder.toString()
        } catch (e: Exception) {
            e.printStackTrace()
        } finally {
```

```

        connection.disconnect()
    }

    return result
}

override fun onPostExecute(result: String) {
    super.onPostExecute(result)
    // Natijani qayta ishlash
}
}
```

POST So'rovi

```kotlin
import java.io.OutputStream
import java.net.HttpURLConnection
import java.net.URL
import android.os.AsyncTask

class PostRequestTask : AsyncTask<String, Void, String>() {
    override fun doInBackground(vararg params: String?): String {
        val urlString = params[0]
        val postData = params[1] // POST ma'lumotlari

        var result = ""
    }
}

```

```
val url = URL(urlString)
val connection = url.openConnection() as HttpURLConnection

try {
    connection.requestMethod = "POST"
    connection.doOutput = true
    val outputStream: OutputStream = connection.outputStream
    outputStream.write(postData?.toByteArray())
    outputStream.flush()
    outputStream.close()

    val inputStream = connection.getInputStream
    val reader = BufferedReader(InputStreamReader(inputStream))
    val stringBuilder = StringBuilder()
    var line: String?

    while (reader.readLine().also { line = it } != null) {
        stringBuilder.append(line).append("\n")
    }

    reader.close()
    result = stringBuilder.toString()
} catch (e: Exception) {
    e.printStackTrace()
} finally {
    connection.disconnect()
}
```

```

        return result
    }

    override fun onPostExecute(result: String) {
        super.onPostExecute(result)
        // Natijani qayta ishlash
    }
}
```

```

### # 3. So'rovlarni Yuborish

Ushbu kod yordamida GET va POST so'rovlarini yuborish mumkin:

```

```kotlin
// GET so'rovi
GetRequestTask().execute("https://jsonplaceholder.typicode.com/posts")
// POST so'rovi
val postData = "param1=value1&param2=value2"
PostRequestTask().execute("https://jsonplaceholder.typicode.com/posts", postData)
```

```

### Tarmoqli Kenglikni Samarali Boshqarish

Tarmoqli kenglikni samarali boshqarish uchun quyidagi tavsiyalarni e'tiborga olish kerak:

#### 1. Keshdan Foydalanish :

- HTTP so'rovlari natijalarini keshlash orqali tarmoq yukini kamaytirish mumkin. Android `HttpURLConnection` sinfida keshlashni qo'llab-quvvatlash uchun `HttpResponseBody` sinfidan foydalanish mumkin.

#### 2. Ma'lumotlarni Siqish :

- Ma'lumotlarni siqish orqali tarmoq orqali o'tkaziladigan ma'lumotlar hajmini kamaytirish mumkin. Server va klient tarafida `gzip` siqishni qo'llab-quvvatlashni ko'rib chiqing.

### 3. Ma'lumotlarni Birlashtirish :

- Kichik ma'lumotlarni birlashtirib, bir vaqtda yuborish orqali tarmoq so'rovlarining sonini kamaytirish mumkin.

### 4. Foydalanuvchi Tarmoqli Kengligi :

- Foydalanuvchining tarmoq holatini (Wi-Fi yoki mobil data) tekshirib, mos ravishda katta hajmdagi ma'lumotlarni yuklash yoki yuklamaslikni hal qilish.

### 5. Oshkor qilish/ko'rinmasdan yuklash (Lazy Loading) :

- Foydalanuvchi interfeysida faqat kerakli ma'lumotlarni yuklash va ortiqcha yukni kamaytirish.

`HttpURLConnection` sinfi HTTP so'rovlarini amalga oshirish uchun kuchli vosita hisoblanadi. Biroq, tarmoqli kenglikni samarali boshqarish uchun keshlash, siqish, ma'lumotlarni birlashtirish, foydalanuvchi tarmog'i holatini tekshirish va oshkor qilish kabi strategiyalardan foydalanish lozim. Ushbu tavsiyalar mobil ilovalarning tarmoqli kenglikdan samarali foydalanishini ta'minlaydi va foydalanuvchi tajribasini yaxshilaydi.

## S142. Mobil ilovalarda tarmoqli kengligi. Bluetooth va Wi-Fi-dan foydalanish

Mobil ilovalarda Bluetooth va Wi-Fi-dan foydalanish uchun kerakli funksiyalarni ishlatish mumkin. Quyidagi misollar Bluetooth va Wi-Fi bilan bog'liq amallarni namoyish etadi:

Bluetooth bilan ishlash :

1. Bluetooth qurilmalarini qidirish va ularga bog'lanish :

```
```kotlin
```

```
val bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()
```

```
if (bluetoothAdapter == null) {
```

```
    // Bluetooth qo'llanmagan
```

```
} else {
```

```
    if (!bluetoothAdapter.isEnabled) {
```

```
        // Bluetooth o'chirilgan
```

```
        val enableBtIntent = Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
```

```
        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT)
```

```
    } else {
```

```
        // Bluetooth yoqilgan, qurilmalarga bog'lanishni boshlash mumkin
```

```
        val device = bluetoothAdapter.getRemoteDevice("00:11:22:33:44:55")
```

```
        val socket = device.createRfcommSocketToServiceRecord(MY_UUID)
```

```
        socket.connect()
```

```
        // Qurilmaga xabar yuborish va o'qish mumkin
```

```
    }
```

```
}
```

```
```
```

2. Bluetooth orqali ma'lumot almashish :



```

```kotlin
// Qurilmaga xabar yuborish
val outputStream = socket.outputStream
outputStream.write("Hello, Bluetooth device!".toByteArray(Charsets.UTF_8))

// Qurilmadan xabar o'qish
val inputStream = socket.inputStream
val buffer = ByteArray(1024)
val bytesRead = inputStream.read(buffer)
val receivedMessage = String(buffer, 0, bytesRead, Charsets.UTF_8)
```

```

Wi-Fi bilan ishlash :

1. Wi-Fi qurilmalarini qidirish va ularga bog'lanish :

```

```kotlin
val wifiManager = applicationContext.getSystemService(Context.WIFI_SERVICE)
as WifiManager

if (!wifiManager.isWifiEnabled) {
    // Wi-Fi o'chirilgan
    wifiManager.isWifiEnabled = true
}

val wifiConfig = WifiConfiguration()
wifiConfig.SSID = "\"MyNetworkSSID\""
wifiConfig.preSharedKey = "\"MyNetworkPassword\""
val netId = wifiManager.addNetwork(wifiConfig)
wifiManager.enableNetwork(netId, true)
```

```

```

2. Wi-Fi orqali ma'lumot almashish :

```
```kotlin
val httpClient = DefaultHttpClient()
val httpGet = HttpGet("http://example.com/data")
val response = httpClient.execute(httpGet)
val statusCode = response.statusLine.statusCode
if (statusCode == HttpStatus.SC_OK) {
 val entity = response.entity
 val inputStream = entity.content
 // Ma'lumotlarni o'qish
}
```
```

Ushbu misollar Bluetooth va Wi-Fi bilan bog'liq amallarni namoyish etadi. Ulardan foydalanish uchun loyihada Bluetooth va Wi-Fi ruxsatnomalariga ega bo'lishi kerak. Misollarda faqat asosiy amallar ko'rsatilgan, komplekslik va boshqa muammo ham mavjud bo'lishi mumkin.

Mobil ilovalarda Bluetooth va Wi-Fi orqali tarmoqli kenglikni samarali boshqarish va foydalanish turli ilovalar uchun muhim ahamiyatga ega. Quyida Bluetooth va Wi-Fi dan foydalanish bo'yicha asosiy ma'lumotlar keltirilgan.

Bluetooth orqali tarmoq ulanishi

Bluetooth yordamida mobil ilovalar orasida ma'lumot uzatish va olish uchun quyidagi qadamlarni amalga oshirish kerak:

1. Ruxsatnomalarni Qo'shish

`AndroidManifest.xml` fayliga Bluetooth bilan ishlash uchun kerakli ruxsatnomalarni qo'shing:

```
```xml
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```
```

2. Bluetooth-ni Yoqish

Bluetooth adapterini olish va uni yoqish:

```
```kotlin
val bluetoothAdapter: BluetoothAdapter? = BluetoothAdapter.getDefaultAdapter()
if (bluetoothAdapter == null) {
 // Qurilmada Bluetooth qo'llab-quvvatlanmaydi
} else {
 if (!bluetoothAdapter.isEnabled) {
 val enableBtIntent = Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
 startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT)
 }
}
```
```

3. Bluetooth Qurilmalarni Qidirish

Qurilmalarni qidirish va ulanish uchun:

```

```kotlin
bluetoothAdapter?.startDiscovery()
val filter = IntentFilter(BluetoothDevice.ACTION_FOUND)
registerReceiver(receiver, filter)

private val receiver = object : BroadcastReceiver() {
 override fun onReceive(context: Context, intent: Intent) {
 val action: String = intent.action!!
 if (BluetoothDevice.ACTION_FOUND == action) {
 val device: BluetoothDevice? =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE)
 val deviceName = device?.name
 val deviceAddress = device?.address // MAC manzil
 // Qurilmani qayta ishlash
 }
 }
}
}
```

```

4. Bluetooth orqali Ma'lumot Uzatish

Ulangan qurilma bilan ma'lumot uzatish uchun `BluetoothSocket` dan foydalaning:

```

```kotlin
class ConnectThread(device: BluetoothDevice) : Thread() {
 private val mmSocket: BluetoothSocket? by lazy(LazyThreadSafetyMode.NONE)
 {
 device.createRfcommSocketToServiceRecord(MY_UUID)
 }
}

```

```

override fun run() {
 bluetoothAdapter?.cancelDiscovery()
 mmSocket?.use { socket ->
 socket.connect()
 // Ma'lumot uzatish va qabul qilish uchun InputStream va OutputStream
 }
}

fun cancel() {
 try {
 mmSocket?.close()
 } catch (e: IOException) {
 Log.e(TAG, "Could not close the client socket", e)
 }
}
}
}

```

## Wi-Fi orqali tarmoq ulanishi

Wi-Fi yordamida tarmoq ulanishini boshqarish va ulanishlarni amalga oshirish uchun quyidagi qadamlarni bajarish kerak:

### # 1. Ruxsatnomalarni Qo'shish

`AndroidManifest.xml` fayliga Wi-Fi bilan ishlash uchun kerakli ruxsatnomalarni qo'shing:

```

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

```

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
...

```

## # 2. Wi-Fi-ni Yoqish

Wi-Fi adapterini olish va uni yoqish:

```
```kotlin
val wifiManager = applicationContext.getSystemService(Context.WIFI_SERVICE)
as WifiManager

if (!wifiManager.isWifiEnabled) {
    wifiManager.isWifiEnabled = true
}
...

```

3. Wi-Fi Tarmog'ini Qidirish

Wi-Fi tarmoqlarini qidirish va ulanish uchun:

```
```kotlin
val wifiManager = applicationContext.getSystemService(Context.WIFI_SERVICE)
as WifiManager

val wifiScanReceiver = object : BroadcastReceiver() {
 override fun onReceive(context: Context, intent: Intent) {
 val success =
 intent.getBooleanExtra(WifiManager.EXTRA_RESULTS_UPDATED, false)
 if (success) {
 scanSuccess()
 }
 }
}

```

```

 } else {
 scanFailure()
 }
 }
}

val intentFilter = IntentFilter()
intentFilter.addAction(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)
registerReceiver(wifiScanReceiver, intentFilter)
wifiManager.startScan()

private fun scanSuccess() {
 val results = wifiManager.scanResults
 // Tarmoqlarni qayta ishlash
}

private fun scanFailure() {
 // Tarmoqni qidirishda xatolik
}

...

```

#### # 4. Wi-Fi Tarmog'iga Ulanish

Wi-Fi tarmog'iga ulanish uchun `WifiConfiguration` dan foydalanish:

```
```kotlin
```

```

val wifiConfig = WifiConfiguration().apply {
    SSID = String.format("\"%s\"", "SSID_name")
    preSharedKey = String.format("\"%s\"", "password")
}

val wifiManager = applicationContext.getSystemService(Context.WIFI_SERVICE)
as WifiManager

val netId = wifiManager.addNetwork(wifiConfig)

```

```
wifiManager.disconnect()
```

```
wifiManager.enableNetwork(netId, true)
```

```
wifiManager.reconnect()
```

```
...
```

Bluetooth va Wi-Fi yordamida tarmoqli kenglikni samarali boshqarish va ulanishlar amalga oshirish Android ilovalarida keng qo'llaniladi. Yuqoridagi misollardan foydalanib, siz Bluetooth va Wi-Fi yordamida ma'lumotlarni qanday uzatish va olishni o'rganishingiz mumkin. Bu usullar foydalanuvchilarga samarali va ishonchli tarmoq ulanishlarini taqdim etadi.

S143. Server bilan ishlash. AsyncTask sinfidan foydalanish

Server bilan kommunikatsiya o'rnini olish Android ilovalari uchun juda muhimdir va bunga ko'p usullar mavjud. `AsyncTask` sinfi, ma'lumotlar olish yoki jo'natish jarayonlarini qulayroq va samaraliroq qilish uchun qulay usuldir. Quyidagi misol, server bilan aloqani o'rnatish va so'rovlar yuborish uchun `AsyncTask` sinfindan foydalanishni ko'rsatadi:

```
```kotlin

import android.os.AsyncTask
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL

class ServerCommunicationTask : AsyncTask<String, Void, String>() {

 override fun doInBackground(vararg params: String?): String {
 val urlString = params[0] // Server URL
 var result = ""

 try {
 val url = URL(urlString)
 val connection = url.openConnection() as HttpURLConnection
 connection.requestMethod = "GET"

 val responseCode = connection.responseCode
 if (responseCode == HttpURLConnection.HTTP_OK) {
 val inputStream = connection.inputStream
 val reader = BufferedReader(InputStreamReader(inputStream))
 val stringBuilder = StringBuilder()

```

```
var line: String?
```

```
while (reader.readLine().also { line = it } != null) {
 stringBuilder.append(line).append("\n")
}
```

```
reader.close()
```

```
result = stringBuilder.toString()
```

```
} else {
```

```
 result = "Error: $responseCode"
```

```
}
```

```
} catch (e: Exception) {
```

```
 e.printStackTrace()
```

```
 result = "Error: ${e.message}"
```

```
}
```

```
return result
```

```
}
```

```
override fun onPostExecute(result: String) {
```

```
 super.onPostExecute(result)
```

```
 // Natijani qayta ishlash
```

```
}
```

```
}
```

```
...
```

Yuqoridagi kodda, `ServerCommunicationTask` nomli `AsyncTask` sinfi `doInBackground` funksiyasida server bilan aloqani o'rnatadi. `onPostExecute` funksiyasi esa amal natijasini qaytaradi.

`AsyncTask` dan foydalanish quyidagi shaklda bo'ladi:

```
```kotlin
val serverUrl = "http://example.com/api/data"
ServerCommunicationTask().execute(serverUrl)
```
```

Bu kod serverga so'rov jo'natadi va javobni qaytaradi. Siz server URL sifatida kerakli manzilni yuboring.

`AsyncTask` sinfi ishga tushirish uchun quyidagi qatordan foydalanishingiz mumkin:

```
```kotlin
ServerCommunicationTask().execute(serverUrl)
```
```

Bu, `ServerCommunicationTask` sinfiga `serverUrl` ma'lumotini uzatadi va server bilan aloqani boshlaydi. Natijani `onPostExecute` funksiyasida qaytaradi. Bu usul sizga server bilan aloqani asinxron ravishda amalga oshirish imkonini beradi.

Android platformasida server bilan ishlash uchun `AsyncTask` sinfi yordamli qo'llaniladi. `AsyncTask` sinfi asinxron operatsiyalarni bajarish uchun qulayliklarga ega.

Quyidagi misol `AsyncTask` sinfidan foydalangan holda server bilan GET so'rovini yuboradi:

```
```kotlin
```

```
class GetDataTask : AsyncTask<Void, Void, String>() {  
    override fun doInBackground(vararg params: Void?): String {  
        val url = URL("https://api.example.com/data")  
        val connection = url.openConnection() as HttpURLConnection  
        connection.requestMethod = "GET"  
  
        return try {  
            val responseCode = connection.responseCode  
            if (responseCode == HttpURLConnection.HTTP_OK) {  
                val inputStream = connection.inputStream  
                val reader = BufferedReader(InputStreamReader(inputStream))  
                val response = StringBuilder()  
                var line: String?  
                while (reader.readLine().also { line = it } != null) {  
                    response.append(line)  
                }  
                reader.close()  
                response.toString()  
            } else {  
                "Error: $responseCode"  
            }  
        } catch (e: IOException) {  
            "Error: ${e.message}"  
        } finally {  
            connection.disconnect()  
        }  
    }  
}
```

```

    }
}

override fun onPreExecute() {
    super.onPreExecute()
    // UI-ni yangilash, masalan, yuklash indikatorini ko'rsatish
}

override fun onPostExecute(result: String) {
    super.onPostExecute(result)
    // Olangan ma'lumotlarni qabul qilish va UI-ni yangilash
    updateUI(result)
}
}
```

```

Yuqoridagi misol quyidagicha ishlatiladi:

```

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val getDataTask = GetDataTask()
        getDataTask.execute()
    }
}
```

```

```

private fun updateUI(data: String) {
 // Olangan ma'lumotlarni ekranda ko'rsatish
 textView.text = data
}
}
...

```

Bu misolda `GetDataTask`` sinfi `AsyncTask`` sinfidan meros oladi va asynchron operatsiyalarni bajaradi:

1. `doInBackground()` metodi - server bilan bog'lanish va ma'lumotlarni olish uchun ishlatiladi. Bu metod asynchron tarzda bajariladi.
2. `onPreExecute()` metodi - asynchron operatsiyadan oldin bajariladi. Bu yerda UI-ni yangilash (masalan, yuklash indikatorini ko'rsatish) uchun foydalanish mumkin.
3. `onPostExecute()` metodi - asynchron operatsiya yakunlangandan keyin bajariladi. Bunda olgan ma'lumotlarni qabul qilib, UI-ni yangilash mumkin.

`AsyncTask`` sinfi erkin thread (asosiy UI thread dan tashqari) yaratadi va unda operatsiyalarni bajaradi. Bu asosiy UI threadni bloklashdan saqlaydi va ilovaning javob berish xususiyatini yaxshilaydi.

Diqqat: Android 11 (API level 30) dan boshlab, `AsyncTask`` sinfi tavsiya etilmaydi. Uning o'rniga Kotlin coroutines yoki RxJava kabi zamonaviy yondashuvlardan foydalanish tavsiya etiladi.

#### 144. Server bilan ishlash. Volley texnologiyasidan foydalanish

Volley, Android ilovalarida tarmoq so'rovlarini qulay va samarali boshqarish uchun ishlatiladigan kuchli kutubxonadir. Uning foydalanish orqali tarmoq so'rovlari yuborish, javobni qabul qilish, va asinxron ravishda ma'lumotlarni olish mumkin. Quyidagi misol, Volley kutubxonasi orqali server bilan aloqani o'rnatishni ko'rsatadi:

## 1. Volley Kutubxonasini Qo'shish

Volley kutubxonasini o'rnatish. Bu Android Studio Gradle faylida qilinishi mumkin:

```
```gradle
dependencies {
    implementation 'com.android.volley:volley:1.2.0'
}
```
```

## 2. So'rov Yuborish va Javobni Qabul Qilish

Quyidagi kod, serverga GET so'rovi yuboradi va javobni qabul qiladi:

```
```kotlin
import com.android.volley.Request
import com.android.volley.Response
import com.android.volley.toolbox.StringRequest
import com.android.volley.toolbox.Volley

val queue = Volley.newRequestQueue(context)
val url = "http://example.com/api/data"

val stringRequest = StringRequest(Request.Method.GET, url,
    Response.Listener<String> { response ->
        // Serverdan kelgan javobni qayta ishlash
    }
})
```

```

    },
    Response.ErrorListener { error ->
        // Xatolik yuz berdi
    })

queue.add(stringRequest)
```

```

### 3. POST So'rovi Yuborish

Agar serverga POST so'rovi yuborish kerak bo'lsa, quyidagi usulni ishlatishingiz mumkin:

```

```kotlin
val url = "http://example.com/api/data"
val postData = HashMap<String, String>()
postData["param1"] = "value1"
postData["param2"] = "value2"

val stringRequest = object : StringRequest(Request.Method.POST, url,
    Response.Listener<String> { response ->
        // Serverdan kelgan javobni qayta ishlash
    },
    Response.ErrorListener { error ->
        // Xatolik yuz berdi
    }) {
    override fun getParams(): Map<String, String> {

```



```

        return postData
    }
}

queue.add(stringRequest)
```

```

Bu kodda, serverga POST so'rovi yuboriladi va kerakli ma'lumotlar quyidagi ko'rinishda joylashtiriladi: `param1=value1&param2=value2`.

#### 4. JSON Ma'lumotlarni Olish va Jo'natish

Volley kutubxonasini JSON ma'lumotlar bilan ishlash uchun ham ishlatishingiz mumkin. Masalan, serverdan JSON ma'lumotlarni olish uchun:

```

```kotlin
val url = "http://example.com/api/data"
val jsonObjectRequest = JsonObjectRequest(Request.Method.GET, url, null,
    Response.Listener { response ->
        // JSON javobni qayta ishlash
    },
    Response.ErrorListener { error ->
        // Xatolik yuz berdi
    })

queue.add(jsonObjectRequest)
```

```

Va serverga JSON ma'lumotlarni jo'natish uchun:

```
```kotlin
```

```
val url = "http://example.com/api/data"
```

```
val jsonObject = JSONObject()
```

```
jsonObject.put("key", "value")
```

```
val jsonObjectRequest = JsonObjectRequest(Request.Method.POST, url, jsonObject,
```

```
    Response.Listener { response ->
```

```
        // JSON javobni qayta ishlash
```

```
    },
```

```
    Response.ErrorListener { error ->
```

```
        // Xatolik yuz berdi
```

```
    })
```

```
queue.add(jsonObjectRequest)
```

```
```
```

Volley kutubxonasini ishlatish bilan, Android ilovalarida tarmoq so'rovlari yuborish va javobni qabul qilish juda osonroq va samaraliroq bo'ladi.

Android platformasida server bilan ishlash uchun Volley texnologiyasidan foydalanish mumkin. Volley kutubxonasi asynchron so'rovlarni bajarish, keshirovchi, so'rovlarni navbatga qo'yish va boshqa imkoniyatlarni taqdim etadi.

Quyidagi misol Volley texnologiyasidan foydalangan holda server bilan GET so'rovini yuboradi:

```

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val queue = Volley.newRequestQueue(this)
        val url = "https://api.example.com/data"

        val stringRequest = StringRequest(
            Request.Method.GET, url,
            { response ->
                // Olingan ma'lumotlarni qabul qilish va UI-ni yangilash
                textView.text = response
            },
            { error ->
                // Xato holatlari uchun ishlov berish
                textView.text = "Error: ${error.message}"
            }
        )

        queue.add(stringRequest)
    }
}
```

```

Yuqoridagi misolda quyidagilar bajariladi:

1. Yangi Volley request navbatini yaratish (`Volley.newRequestQueue(this)`).
2. Server manzilini e'lon qilish (`"https://api.example.com/data"`).
3. `StringRequest` obyektini yaratish:
  - Bu obyekt HTTP GET so'rovini bajaradi.
  - Muvaffaqiyatli javob olinganda, `response` parametri orqali ma'lumotlarni olish va UI-ni yangilash.
  - Xato yuzberganda, `error` parametri orqali xato xabarini olish va UI-ni yangilash.
4. Yaratilgan so'rovni navbatga qo'shish (`queue.add(stringRequest)`).

Bu misolda faqat GET so'rovi namoyish etilgan, biroq Volley texnologiyasi POST, PUT, DELETE va boshqa so'rovlarni ham bajarishi mumkin.

Volley texnologiyasi boshqa kompaktroq va qulayroq yondashuvlarni taklif etadi. Volley asynchron so'rovlarni bajaradi, so'rovlarni navbatga qo'yadi, keshirovchi qiladi, network xatolarini aniqlaydi va boshqa funktsiyalarni taqdim etadi.

Diqqat: Volley Android 9 (API level 28) dan beri tavsiya etilmaydi. Uning o'rniga Retrofit yoki Ktor kabi zamonaviy kutubxonalardan foydalanish tavsiya etiladi.

## 145. Server bilan ishlash. Qayta jihozlash texnologiyasidan foydalanish

Android platformasida server bilan ishlash uchun Retrofit texnologiyasidan foydalanish mumkin. Retrofit kuchli va moslashuvchan HTTP client bo'lib, asynhron so'rovlarni bajarish, JSON ma'lumotlarini avtomatik marshaling qilish, URL dinamik parametrlarini qo'llab-quvvatlash kabi xususiyatlarga ega.

Quyidagi misol Retrofit texnologiyasidan foydalangan holda server bilan GET so'rovini yuboradi:

```
```kotlin

// API interface
interface DataApi {
    @GET("data")
    suspend fun getData(): Response<DataResponse>
}

// Data model
data class DataResponse(
    val id: Int,
    val name: String,
    val description: String
)

class MainActivity : AppCompatActivity() {
    private val dataApi = Retrofit.Builder()
        .baseUrl("https://api.example.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()
        .create(DataApi::class.java)
```

```

override fun onCreate(savedInstanceState: Bundle?) {

    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Coroutines orqali server bilan ishlash
    viewModelScope.launch {
        try {
            val response = dataApi.getData()
            if (response.isSuccessful) {
                // Olingan ma'lumotlarni qabul qilish va UI-ni yangilash
                val data = response.body()
                updateUI(data)
            } else {
                // Xato holatlari uchun ishlov berish
                Toast.makeText(this@MainActivity, "Error: ${response.message()}",
                    Toast.LENGTH_SHORT).show()
            }
        } catch (e: Exception) {
            // Umumiy xatolar uchun ishlov berish
            Toast.makeText(this@MainActivity, "Error: ${e.message}",
                Toast.LENGTH_SHORT).show()
        }
    }
}

private fun updateUI(data: DataResponse?) {
    // Olingan ma'lumotlarni ekranda ko'rsatish

```

```
        textView.text = "${data?.name} - ${data?.description}"
    }
}
...

```

Yuqoridagi misolda quyidagilar bajariladi:

1. `DataApi` interfeysi yaratiladi. Bu interface server bilan ishlashning API-ni ifodalaydi. Bunda `@GET("data")` annotatsiyasi orqali GET so'rovini aniqlab qo'yiladi.
2. `DataResponse` data sinfi yaratiladi. Bu sinf server tomonidan qaytariladigan ma'lumotlar modelini ifodalaydi.
3. `Retrofit` servisi yaratiladi. Bunda base URL, konvertorlar (masalan, Gson) va boshqa parametrlar belgilanadi.
4. `DataApi` interfeysi orqali server bilan ishlash uchun `getData()` funksiyasi chaqiriladi. Bu funksiya `suspend` funktsiya bo'lib, Coroutines yordamida asynchron bajariladi.
5. Server so'rovining natijasi qabul qilinadi va UI-ga yangilash uchun uzatiladi.

Retrofit texnologiyasi Android rivojlanishida eng keng tarqalgan yondashuvlardan biridir. U HTTP so'rovlarini bajarish uchun qulayliklar taqdim etadi, bunda REST API-larini "interface" sifatida ifodalash imkonini beradi. Retrofit Coroutines yoki RxJava kabi zamonaviy yondashuvlar bilan yaxshi integratsiya qilinadi.

Qayta jihozlash (Retrofit) Android ilovalarida server bilan ishlash uchun kuchli va samarali bir texnologiyadir. Bu, RESTful API'lar bilan ishlash uchun oson interfeyslar yaratish va foydalanish imkonini beradi. Quyidagi qadamlar orqali Retrofit kutubxonasidan foydalanish mumkin:

1. Retrofit Kutubxonasini Qo'shish

Gradle faylida Retrofit kutubxonasini qo'shing:

```
```gradle
dependencies {
 implementation 'com.squareup.retrofit2:retrofit:2.9.0'
 implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
}
```
```

2. RESTful API Interfeysini Yaratish

Server bilan interfeysni belgilab, qaysi so'rovlarni qanday qilib yuborish va qabul qilish kerakligini aniqlang:

```
```kotlin
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.POST
import retrofit2.http.Body

interface ApiService {
 @GET("api/data")
 fun fetchData(): Call<DataResponse>

 @POST("api/send")
 fun sendData(@Body postData: Data): Call<Void>
}
```
```

3. Retrofit Ma'lumot Interfeysini O'rnatish

Retrofit ma'lumot interfeysini o'rnatish va serverga aloqani boshqarish uchun foydalanish:

```
```kotlin
import retrofit2.Retrofit
```



```
import retrofit2.converter.gson.GsonConverterFactory
val retrofit = Retrofit.Builder()
 .baseUrl("http://example.com/")
 .addConverterFactory(GsonConverterFactory.create())
 .build()
val apiService = retrofit.create(ApiService::class.java)
```

```

4. Ma'lumotlarni Olish va Jo'natish

Serverdan ma'lumotlarni olish va jo'natish uchun yaratilgan interfeysni ishlatish:

```
```kotlin
// Serverdan ma'lumotlarni olish
val call = apiService.fetchData()
call.enqueue(object : Callback<DataResponse> {
 override fun onResponse(call: Call<DataResponse>, response:
Response<DataResponse>) {
 if (response.isSuccessful) {
 val dataResponse = response.body()
 // Ma'lumotlar muvaffaqiyatli qabul qilindi
 } else {
 // Xatolik yuz berdi
 }
 }
})
override fun onFailure(call: Call<DataResponse>, t: Throwable) {
 // Serverga so'rovni yuborishda xatolik yuz berdi
}
}
// Ma'lumotlarni serverga jo'natish
```

```

```

val data = Data("value1", "value2")
val sendCall = apiService.sendData(data)
sendCall.enqueue(object : Callback<Void> {
    override fun onResponse(call: Call<Void>, response: Response<Void>) {
        if (response.isSuccessful) {
            // Ma'lumotlar muvaffaqiyatli jo'natildi
        } else {
            // Xatolik yuz berdi
        }
    }
    override fun onFailure(call: Call<Void>, t: Throwable) {
        // Serverga so'rovni yuborishda xatolik yuz berdi
    }
})
...

```

Bu kodlar yordamida Retrofit kutubxonasi yordamida server bilan interfeysni o'rnatish, ma'lumotlarni olish va jo'natishning asosiy qadamlarini ko'rsatilgan.

S146. Ilovada Google xaritalarini ulash va sozlash

Android platformasida ilovangizga Google xaritalarini ulash va sozlash uchun quyidagi bosqichlarni bajarish tavsiya etiladi:

1. Google Maps SDK uchun API kaliti olish :

- Google Cloud Console platformasida yangi loyiha yarating.
- "APIs & Services" bo'limiga o'ting va "Library" orqali "Google Maps Android API" ni qo'shing.
- Keyin "Credentials" bo'limiga o'ting va "Create credentials" tugmasini bosib, "API key" ni oling.

2. Google Play Services kiritish :

- Loyihangizning `build.gradle` (Module: app) faylida Google Play Services-ni kiritish uchun quyidagi dependency-ni qo'shing:

```
...  
  
dependencies {  
    implementation 'com.google.android.gms:play-services-maps:18.1.0'  
}  
...
```

- Loyihangizning `build.gradle` (Project) faylida Google Play Services-ga oid Google Maven repo-ni qo'shing:

```
...  
  
allprojects {  
    repositories {  
        google()  
    }  
}  
...
```

3. XML'da xaritani joylashtirishni sozlash :

- `activity_main.xml` (yoki boshqa activity layout faylida) ``<fragment>`` elementini quyidagicha joylashtirishingiz mumkin:

```
```xml
<fragment
 android:id="@+id/map"
 android:name="com.google.android.gms.maps.SupportMapFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent" />
```
```

4. Java/Kotlin kodida xaritani sozlash :

- `MainActivity.kt` (yoki boshqa activity) faylida quyidagi kodni yozing:

```
```kotlin
class MainActivity : AppCompatActivity(), OnMapReadyCallback {
 private lateinit var googleMap: GoogleMap

 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)

 val mapFragment = supportFragmentManager
 .findFragmentById(R.id.map) as SupportMapFragment
 mapFragment.getMapAsync(this)
 }

 override fun onMapReady(map: GoogleMap) {
 googleMap = map
 }
}
```

```

// Xaritani sozlash uchun kodlar

googleMap.mapType = GoogleMap.MAP_TYPE_NORMAL
googleMap.uiSettings.isZoomControlsEnabled = true
val location = LatLng(40.7128, -74.0060) // New York City
googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(location,
12f))
 }
}
...

```

## 5. Manifest faylida Google Maps API kalitini belgilash :

- `AndroidManifest.xml` faylida Google Maps SDK uchun API kalitini quyidagicha belgilang:

```

<?xml
<application>
 <!-- ... -->
 <meta-data
 android:name="com.google.android.geo.API_KEY"
 android:value="YOUR_API_KEY" />
 </application>
...

```

Bu bosqichlarni bajarish natijasida Google xaritalarini ilovangizga qo'shishingiz va sozlashingiz mumkin. Xaritalarni samarali va interaktiv qilish uchun API-ning boshqa funktsiyalaridan ham foydalanishingiz mumkin, masalan, markerlarga, poligonlarga, marshrut yo'nalishlariga va boshqalarga.

Google xaritalarini ilovangizga qo'shish va sozlash uchun quyidagi qadamlarni amalga oshirishingiz mumkin:

### 1. Google Cloud Console dan API yaratish

- Google Cloud Console ga kirib, "APIs & Services" bo'limiga o'ting.
- "Library" dan "Maps SDK for Android" ni izlash va uni yoqish.
- "Credentials" bo'limiga o'ting va yangi API Key yaratish.

## 2. API Key-ni ilovangizga qo'shish

- Android ilovangizning `AndroidManifest.xml` faylidagi `` tegiga quyidagi satrni qo'shing:

```
```xml
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY"/>
```
```

- "YOUR\_API\_KEY" ni o'zingiz yaratgan API Key bilan almashtiring.

## 3. Google Xaritalarini o'rnatish

- Ilovangizda Google Xaritalarni ishlatish uchun `build.gradle` faylga quyidagi kutubxonani qo'shing:

```
```gradle
implementation 'com.google.android.gms:play-services-maps:18.0.0'
```
```

## 4. Xaritalarni oynatish uchun Fragment qo'shish

- Ilovangizning o'zi turli qismlarida xarita ko'rsatish uchun `SupportMapFragment` yoki `MapView` qo'shishingiz mumkin.

- `activity\_main.xml` faylga quyidagi kodni qo'shing:

```
```xml
<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```
```

## 5. Xaritalarni sozlash

- Xaritalarni sozlash va boshqarish uchun `GoogleMap` obyektiga murojat qiling. Masalan, aktivitetda quyidagi kodni ishlatish mumkin:

```
```kotlin
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment

class MapsActivity : AppCompatActivity(), OnMapReadyCallback {
    private lateinit var mMap: GoogleMap

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
    }
}
```

```

        val mapFragment = supportFragmentManager.findFragmentById(R.id.map)
        as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap

        // Xarita sozlash va boshqarish kodlari...
    }
}

```

- Xaritaning koordinatalarini qo'yish, markerlar qo'shish, joylashuvlarni belgilash va boshqalar kabi amallarni bajarish uchun `GoogleMap` obyektini ishlatish mumkin.

Bu qadamlar orqali siz ilovangizga Google Xaritalarini qo'shishingiz va xaritalarni sozlashingiz mumkin. Umid qilamanki, bu yordam sizning ilovangizda qulay va samarali xaritalar integratsiyasini ta'minlaydi.

S147. Google Xaritalar yordamida joylashuvni aniqlash.

Joylashuvni aniqlash uchun Google Xaritalar API dan foydalanishingiz mumkin. Quyidagi kodni ishlatib, joylashuvni aniqlashingiz mumkin:

1. Google Xaritalar API-ni yoqish

Google Cloud Console da, API & Services bo'limiga kirib, "Maps SDK for Android" ni yoqing va uni yoqishingiz kerak.

2. API Key olish

API-ni yoqqaningizdan so'ng, API Key olishingiz kerak bo'ladi.

3. Google Xaritalarni o'rnatish

Ilovangizga Google Xaritalarni o'rnatish uchun `build.gradle` faylga quyidagi kutubxonani qo'shing:

```
```gradle
implementation 'com.google.android.gms:play-services-maps:18.0.0'
```
```

4. Xaritalarni boshlash

`activity_main.xml` faylga quyidagi kodni qo'shing:

```
```xml
<fragment
 android:id="@+id/map"
 android:name="com.google.android.gms.maps.SupportMapFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent"/>
```
```

5. Kod yordamida joylashuvni aniqlash

Java yoki Kotlin klasslarida joylashuvni aniqlash uchun `GoogleMap` obyektiga murojat qiling:

```

```kotlin

import android.Manifest
import android.content.pm.PackageManager
import android.os.Bundle
import android.widget.Toast
import androidx.core.app.ActivityCompat
import androidx.fragment.app.FragmentActivity
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.LatLng

class MapsActivity : FragmentActivity(), OnMapReadyCallback {
 private lateinit var mMap: GoogleMap

 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_maps)
 val mapFragment = supportFragmentManager.findFragmentById(R.id.map) as
SupportMapFragment
 mapFragment.getMapAsync(this)
 }

 override fun onMapReady(googleMap: GoogleMap) {
 mMap = googleMap
 }
}

```

```

 val permission = ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)

 if (permission == PackageManager.PERMISSION_GRANTED) {

 mMap.isMyLocationEnabled = true

 } else {

 ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), 1)

 }

 mMap.setOnMyLocationChangeListener { location ->

 // Foydalanuvchi joylashuvi o'zgarishi uchun kodlar

 val latLng = LatLng(location.latitude, location.longitude)

 Toast.makeText(this, "Lat: ${location.latitude}, Long:
${location.longitude}", Toast.LENGTH_SHORT).show()

 mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,
15f))

 }

 }

}

...

```

Bu kod foydalanuvchi joylashuvi o'zgarishlarini eslab turish uchun xaritada markerlarni yangilab turadi. Kod foydalanuvchining joylashuvi o'zgarishi bilan birga xaritani ham yangilaydi va foydalanuvchiga yangi joylashuvi to'g'risidagi xabar chiqaradi.

Bu kod Google Xaritalar API yordamida joylashuvni aniqlash uchun bir misol hisoblanadi. Umid qilamanki, bu qadamlar sizning ilovangizda foydalanuvchi joylashuvini muvaffaqiyatli aniqlashga yordam beradi.

Google Xaritalar platformasi orqali joylashuvni aniqlash uchun quyidagi bosqichlarni bajarishingiz mumkin:

### 1. Google Play Services-ni tekshirish :

- Ilovangizda Google Play Services mavjudligini tekshiring. Ushbu xizmatlar telefoningizda o'rnatilgan bo'lishi kerak.

- `MainActivity.kt` (yoki ilovadagi boshqa aktivlik) faylida quyidagi kodni qo'shing:

```
```kotlin

private fun isGooglePlayServicesAvailable(): Boolean {

    val googleApiAvailability = GoogleApiAvailability.getInstance()

    val resultCode = googleApiAvailability.isGooglePlayServicesAvailable(this)

    return resultCode == ConnectionResult.SUCCESS

}

```
```

### 2. Joylashuvning aniqlanishi uchun Permissionlarni sozlash :

- `AndroidManifest.xml` faylida joylashuvni aniqlash uchun quyidagi permissionlarni qo'shing:

```
```xml

<manifest>

    <!-- ... -->

    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />

    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />

</manifest>

```
```

### 3. Joylashuvni aniqlash va o'rnini yangilash :

- `MainActivity.kt` faylida joylashuvni aniqlash va o'rnini yangilash uchun quyidagi kodni qo'shing:

```
```kotlin
```

```
private val LOCATION_PERMISSION_REQUEST_CODE = 100
```

```
private fun requestLocationPermission() {
```

```
    ActivityCompat.requestPermissions(
```

```
        this,
```

```
        arrayOf(
```

```
            Manifest.permission.ACCESS_FINE_LOCATION,
```

```
            Manifest.permission.ACCESS_COARSE_LOCATION
```

```
        ),
```

```
        LOCATION_PERMISSION_REQUEST_CODE
```

```
    )
```

```
}
```

```
private fun isLocationPermissionGranted(): Boolean {
```

```
    val fineLocationPermission =
```

```
        ContextCompat.checkSelfPermission(this,
```

```
        Manifest.permission.ACCESS_FINE_LOCATION)
```

```
    val coarseLocationPermission =
```

```
        ContextCompat.checkSelfPermission(this,
```

```
        Manifest.permission.ACCESS_COARSE_LOCATION)
```

```
    return fineLocationPermission == PackageManager.PERMISSION_GRANTED  
&&
```

```
        coarseLocationPermission ==
```

```
        PackageManager.PERMISSION_GRANTED
```

```
}
```

```

private fun getCurrentLocation() {
    if (isLocationPermissionGranted()) {
        val fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
        fusedLocationClient.lastLocation
            .addOnSuccessListener { location: Location? ->
                if (location != null) {
                    val latitude = location.latitude
                    val longitude = location.longitude
                    // Joylashuv ma'lumotlari (latitude va longitude) bilan ishlash
                    // ...
                }
            }
    } else {
        requestLocationPermission()
    }
}
}

```

4. Joylashuvni tekshirish va o'rnini olish :

- `MainActivity.kt` faylida `onCreate()` metodida quyidagi kodni qo'shing:

```
```kotlin
```

```

override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout.activity_main)
}

```

```

 if (isGooglePlayServicesAvailable()) {
 getLocation()
 } else {
 // Google Play Services mavjud emas
 // ...
 }
 }
}
...

```

##### 5. Permissionni tekshirish va natijasini olish :

- `MainActivity.kt` faylida joylashuv permissionlarini tekshirish uchun `onRequestPermissionsResult` metodini qo'shing:

```

```kotlin

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<String>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (requestCode == LOCATION_PERMISSION_REQUEST_CODE) {
        if (grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            getLocation()
        } else {
            // Joylashuvga ruxsat berilmadi
            // ...
        }
    }
}

```

```
}  
}  
...
```

Ushbu bosqichlar natijasida ilovangiz joylashuvni aniqlash uchun Google Xaritalar yordamida foydalanishi mumkin. Joylashuv ma'lumotlarini (`latitude` va `longitude`) olishdan so'ng siz ularga kerakli ishni bajarishingiz mumkin, masalan, xaritada joylashuvni belgilash, marker yoki poligon qo'shish, yo'nalishni hisoblash va boshqalar.

S148. Google Xaritalar yordamida joy nomlari va ma'lumotlarini aniqlash

Google Xaritalar platformasi orqali joy nomlari va ma'lumotlarini aniqlash uchun quyidagi bosqichlarni bajarishingiz mumkin:

1. Geocoding API-dan foydalanish :

- Geocoding - bu joylashuv ma'lumotlaridan (GPS koordinatalari) joy nomini aniqlash jarayonidir.

- Buni amalga oshirish uchun Google Xaritalar Geocoding API-dan foydalanishingiz kerak.

- API kalyutini olish va ilovangizda sozlash talab etiladi.

2. Geocoding so'rovini amalga oshirish :

- Geocoding so'rovini amalga oshirish uchun quyidagi kodni yozishingiz mumkin:

```
```kotlin
```

```
private fun getAddressFromLocation(latitude: Double, longitude: Double) {
```

```
 val geocoder = Geocoder(this, Locale.getDefault())
```

```
 val addresses = geocoder.getFromLocation(latitude, longitude, 1)
```

```
 if (addresses.isNotEmpty()) {
```

```
 val address = addresses[0]
```

```
 // Quyidagi ma'lumotlardan foydalanishingiz mumkin:
```

```
 val addressLine = address.getAddressLine(0)
```

```
 val city = address.locality
```

```
 val state = address.adminArea
```

```
 val country = address.countryName
```

```
 val postalCode = address.postalCode
```

```
 // ...
```

```
 } else {
```

```
 // Joylashuv ma'lumotlari topilmadi
```

```
}
}
...
```

### 3. Natijalarni qayta ishlash :

- Geocoding so'rovi natijasida olingan ma'lumotlar (`addressLine`, `city`, `state`, `country`, `postalCode` va boshqalar) orqali kerakli ma'lumotlardan foydalanishingiz mumkin.

- Masalan, joy nomini, manzilingizni yoki boshqa ma'lumotlarni ko'rsatish, xaritada belgilash, yoki boshqa maqsadlar uchun ishlatishingiz mumkin.

### 4. Reverse Geocoding :

- Reverse Geocoding - bu joy nomidan joylashuv ma'lumotlarini (GPS koordinatalarini) aniqlash jarayonidir.

- Buni amalga oshirish uchun quyidagi kodni yozishingiz mumkin:

```
```kotlin  
  
private fun getLocationFromAddress(address: String) {  
    val geocoder = Geocoder(this, Locale.getDefault())  
    val addresses = geocoder.getFromLocationName(address, 1)  
    if (addresses.isNotEmpty()) {  
        val location = addresses[0]  
        // Quyidagi ma'lumotlardan foydalanishingiz mumkin:  
        val latitude = location.latitude  
        val longitude = location.longitude  
        // ...  
    } else {  
        // Joylashuv ma'lumotlari topilmadi  
    }  
}
```

```
}  
...  

```

5. Natijalarni qayta ishlash :

- Reverse Geocoding so'rovi natijasida olingan ma'lumotlar (`latitude`, `longitude`) orqali kerakli ma'lumotlardan foydalanishingiz mumkin.
- Masalan, xaritada belgilash, koordinatalarni saqlab olish yoki boshqa maqsadlar uchun ishlatishingiz mumkin.

Ushbu bosqichlar natijasida ilovangiz Google Xaritalar yordamida joy nomlari va ma'lumotlarini olib, ulardan kerakli maqsadlar uchun foydalana oladi.

Joy nomlari va ma'lumotlarini aniqlash uchun Google Xaritalar API ning Places API sifatida taqdim etilgan qismidan foydalanishingiz mumkin. Bu API yordamida joylarni topish, joy nomlarini qidirish, joy ma'lumotlarini olish va boshqalar kabi funktsiyalarni amalga oshirishingiz mumkin. Quyidagi qadamlar orqali joy nomlari va ma'lumotlarini aniqlashni amalga oshirishingiz mumkin:

1. Google Xaritalar API-ni yoqish

Google Cloud Console da, API & Services bo'limiga kirib, "Places API" ni yoqing va uni yoqishingiz kerak.

2. API Key olish

API-ni yoqqaningizdan so'ng, API Key olishingiz kerak bo'ladi.

3. Retrofit kutubxonasi o'rnatish (optional)

Joy nomlarini qidirish uchun API-ga so'rov yuborishda Retrofit kutubxonasidan foydalanishingiz yaxshi bo'ladi.

4. Joy nomlarini qidirish

Retrofit kutubxonasidan foydalanish orqali joy nomlarini qidirish so'rovi yuboring:

```
```kotlin
interface PlacesApiService {
 @GET("maps/api/place/findplacefromtext/json")
 fun findPlaces(
 @Query("input") input: String,
 @Query("inputtype") inputType: String,
 @Query("fields") fields: String,
 @Query("key") apiKey: String
): Call<PlacesResponse>
}
```
```

5. Ma'lumotlarni qabul qilish

Joy nomlarini qidirganingiz, javobni qabul qiling va kerakli ma'lumotlarni oling:

```
```kotlin
val retrofit = Retrofit.Builder()
 .baseUrl("https://maps.googleapis.com/")
 .addConverterFactory(GsonConverterFactory.create())
```

```
.build()
```

```
val placesApiService = retrofit.create(PlacesApiService::class.java)
```

```
val call = placesApiService.findPlaces(
 input = "restaurant",
 inputType = "textquery",
 fields = "name,formatted_address,opening_hours,rating",
 apiKey = "YOUR_API_KEY"
)
```

```
call.enqueue(object : Callback<PlacesResponse> {
 override fun onResponse(call: Call<PlacesResponse>, response:
Response<PlacesResponse>) {
 if (response.isSuccessful) {
 val placesResponse = response.body()
 // Joy nomlarini olish uchun javobni qayta ishlash
 } else {
 // Xatolik yuz berdi
 }
 }
})
```

```
override fun onFailure(call: Call<PlacesResponse>, t: Throwable) {
 // Serverga so'rovni yuborishda xatolik yuz berdi
}
})
...
```

## 6. Natijalarni ishlash

Qabul qilingan ma'lumotlardan kerakli ishlarni bajaring, masalan, xabarlar chiqaring yoki ma'lumotlarni oynang.

Bu qadamlar orqali siz Google Xaritalar API orqali joy nomlarini qidirishingiz va ma'lumotlarini olishingiz mumkin. Umid qilamanki, bu qadamlar sizning ilovangizga joy nomlarini aniqlashda yordam beradi.

## S149. Platformalararo dasturlash. Dart dasturlash tilining asosiy tushunchalari

Dart dasturlash tilini platformalararo dasturlash uchun foydalanish mumkin. Dart, Google tomonidan ishlab chiqilgan, platformalararo va ob'ektga yo'naltirilgan dasturlash tili hisoblanadi. Bu til Android va iOS ilovalarini o'rnatish, veb ilovalarni yaratish, server tomonidagi dasturlar yaratish va boshqa platformalar uchun ham foydalaniladi. Quyidagi asosiy tushunchalari bilib olishingiz yaxshi bo'ladi:

### 1. OOP va asinxron operatsiyalar

Dart ob'ektga yo'naltirilgan dasturlash tili hisoblanadi. Bu, ob'ektga yo'naltirilgan bo'lib, ob'ektlarni yaratish, klasslarni yaratish, o'yinchaklar, interfeyslar va ko'p qo'shimcha funksiyalarni qo'llash imkonini beradi. Asinxron operatsiyalar bilan ishlash uchun Dartda `Future` va `Stream` ob'ektlari mavjud. Bu, uzluksiz, bloklashmaydigan va samarali operatsiyalar bajarish imkonini beradi.

### 2. Just-In-Time (JIT) va Ahead-Of-Time (AOT) kompilyatsiya

Dart kodlari Just-In-Time (JIT) va Ahead-Of-Time (AOT) kompilyatsiyasi orqali boshqariladi. JIT kompilyatsiya yoqilib qolganda, Dart kodlari ishlayotgan dasturlash muhitida quriladi va o'zgarishlar to'g'risida tezkor natijalar beradi. AOT kompilyatsiyasi esa Android va iOS ilovalari uchun kodni tartiblangan natijalar sifatida jo'natish uchun ishlatiladi.

### 3. Flutter dasturlash freymvorki

Dart, xususi ravishda Google tomonidan Flutter dasturlash freymvorki uchun yaratilgan. Flutter, ishlab chiqilgan qo'llanma interfeyslar (UI) yaratish uchun qulay va samarali bir vosita hisoblanadi. Dart tilini ishlatib, Flutter orqali Android va iOS ilovalari, veb ilovalar va desktop ilovalarni ham yaratishingiz mumkin.

### 4. Dart VM va JavaScript

Dart kodlari Dart Virtual Machine (VM) yoki JavaScript kodiga o'zgartirilishi mumkin. Bu, Dart kodlarini server tomonidagi dasturlash, veb ilovalar yaratish va boshqa platformalar uchun ham qulay va kuchli qiladi.

### 5. Ko'p platformalar uchun dasturlash

Dart, qulaylik va samarali xususiyatlari tufayli, ko'p platformalar uchun dasturlashda mashxurdir. Bu, boshqa dasturlash tillari bilan tanishligi va uning platformalararo qulayliklari tufayli, yuqori darajada platformalararo dasturlash uchun tanlov qilinadi.

Dart tilining asosiy tushunchalari bu bo'lsa-da, siz yana ko'proq tushunchalar va bilimlarni o'rganish uchun Dart tilining rasmiy veb-sahifasiga yoki onlayn darsliklarga murojaat qilishingiz mumkin.

Platformalararo dasturlash kontekstida Dart dasturlash tilining asosiy tushunchalari quyidagilar:

1. Ob'ekt Yo'naltirilgan Dasturlash (OOD) : Dart OOD tillaridan biri bo'lib, dasturlarni ob'ektlar va ularning o'zaro munosabatlari asosida yozishga imkon beradi. Dart dasturchilar sinflar, ob'ektlar, meros, polimorfizm va boshqa OOD tushunchalaridan foydalanadilar.
2. Funktsional Dasturlash : Dart tilida funktsional dasturlash yondashuvi ham qo'llab-quvvatlanadi. Funktsiyalar birinchi darajali ob'ektlar hisoblanadi va ularni o'zgaruvchilarga saqlab, parametr sifatida uzatish mumkin.
3. Asinkron Dasturlash : Dart asinkron dasturlashni keng qo'llab-quvvatlaydi. Async/await va Futures kabi tushunchalar orqali asinkron amallarni oson amalga oshirish mumkin.
4. Tiplar Tizimi : Dart statik va dinamik tiplar tizimini qo'llab-quvvatlaydi. Barcha o'zgaruvchilarga ularning tipini aniqlash shart emas, Dart kompilatori buni avtomatik ravishda aniqlay oladi.
5. Modullar : Dart dasturlarni modullar shaklida tashkil etish imkoniyatini beradi. Modullar orqali kodni qayta ishlatish, modullarni import qilish va boshqalar amalga oshiriladi.
6. Kutubxonalar : Dart katta ekotizimga ega bo'lib, turli xil vazifalarni bajaruvchi ko'plab kutubxonalar mavjud. Flutter, Angular, Node.js kabi platformalarda Dart tilidan foydalanish mumkin.
7. Xavfsizlik : Dart tili xavfsizlikka katta e'tibor beradi. Masalan, null xavfsizligi tushunchasi orqali dasturchilar null qiymatlardan kelib chiqadigan xatoliklarni oldini olishi mumkin.
8. Platformalararo Qo'llab-quvvatlash : Dart dasturlar turli xil platformalarda (iOS, Android, web, desktop) ishlash imkoniyatiga ega. Flutter framework orqali cross-platform ilovalar yaratish mumkin.



9. Interpretatsiya va Kompilyatsiya : Dart tili hem interpretatsiya qilinishi, ham kompilatsiya qilinishi mumkin. Boshqacha qilib aytganda, Dart dasturlarini "just-in-time" (JIT) va "ahead-of-time" (AOT) usullarida bajarish mumkin.

Ushbu tushunchalar Dart dasturlash tilining asosiy xususiyatlari hisoblanadi va platformalararo dasturlashda uni yaxshi qo'llab-quvvatlanishiga olib keladi.

## S150. Platformalararo dasturlash. Dart dasturlash tilida klasslar va ob'ektlar bilan ishlash

Dart dasturlash tilida klasslar va ob'ektlar bilan ishlash ob'ektga yo'naltirilgan dasturlashning asosiy qismidir. Klasslar ob'ektning qurilmasi uchun kerakli belgilar va xususiyatlarni (o'zelliklarni) ta'riflash uchun ishlatiladi, ob'ektlar esa yaratilgan klasslardan amalga oshiriladi. Quyidagi misolda klass va ob'ektlar bilan ishlashni ko'rsataman:

```
```dart
// Klassning ta'rifini qilish
class Car {
  // Xususiyatlar (o'zelliklar)
  String brand;
  String model;
  int year;

  // Konstruktor
  Car(String brand, String model, int year) {
    this.brand = brand;
    this.model = model;
    this.year = year;
  }

  // Metod
  void display() {
    print('Brand: $brand, Model: $model, Year: $year');
  }
}
```

```

}

void main() {
    // Ob'ekt yaratish
    Car myCar = Car('Toyota', 'Corolla', 2020);

    // Ob'ektning xususiyatlariga murojat
    print(myCar.brand); // Toyota
    print(myCar.model); // Corolla
    print(myCar.year); // 2020

    // Ob'ektning metodini chaqirish
    myCar.display(); // Brand: Toyota, Model: Corolla, Year: 2020
}

```

Yuqoridagi kodda, `Car` klassi ta'riflandi va unga `brand`, `model` va `year` xususiyatlari (o'zelliklari) qo'shildi. `Car` klassining konstruktori yaratildi va ob'ekt yaratilganida bu xususiyatlarga qiymatlar o'rnatiladi. `display` metodi ob'ektning xususiyatlari bilan ma'lumotni chiqaradi.

Kodningizda klasslarni va ob'ektlarni yaxshi tushunish uchun quyidagi muhim nuqtalarga e'tibor bering:

- Klasslar (sinflar) : Klasslar ob'ektning xususiyatlari va metodi bilan ta'riflanadi. Ularning o'z xususiyatlari va metodlari bo'ladi.
- Ob'ektlar : Klasslardan yaratilgan vaqtincha yoki doimiy ma'lumotlarni o'z ichiga oladi.

- Konstruktorlar : Klasslardan ob'ektlarni yaratish uchun ishlatiladi va o'zgaruvchilarni boshlang'ich qiymatlari bilan birlashtiriladi.
- Xususiyatlar (o'zelliklar) : Ob'ektning ma'lumotlarini saqlaydigan o'zgaruvchilar.
- Metodlar : Klassning funksiyalari, ob'ektlar bilan amalga oshiriladigan ishlarni bajarish uchun foydalaniladi.

Dartda klasslar va ob'ektlar yordamida dasturlar yaratish odatiy va kuchli amaldir. Bu konseptlarni yaxshi tushunish va ularni qo'llash, dasturlashning o'zini o'rganish uchun muhimdir.

Dart dasturlash tilida klasslar va ob'ektlar bilan ishlash quyidagi tarzda amalga oshiriladi:

1. Klass Yaratish :

```

```dart

class Person {
 String name;
 int age;

 Person(this.name, this.age);

 void greet() {
 print('Hello, my name is $name and I am $age years old.');
```

Bu yerda `Person` degan klass yaratildi. Unda `name` va `age` o'zgaruvchilari va `greet()` metodi bor.

## 2. Ob'ekt Yaratish :

```
```dart
void main() {
  var person = Person('John Doe', 35);
  person.greet(); // Output: Hello, my name is John Doe and I am 35 years old.
}
```
```

`main()` funksiyasida `Person` klassining bir nusxasi (`person` o'zgaruvchisi) yaratildi va `greet()` metodi chaqirildi.

## 3. Konstruktorlar :

```
```dart
class Person {
  String name;
  int age;

  // Parametrsiz konstruktor
  Person() {
    name = 'Unknown';
    age = 0;
  }

  // Parametrli konstruktor
  Person(this.name, this.age);
}
```

```
}  
...
```

Klassda bir nechta konstruktorlar bo'lishi mumkin. Yuqoridagi misolda `Person()` va `Person(this.name, this.age)` konstruktorlari mavjud.

4. Meros (Inheritance) :

```
```dart  
class Student extends Person {
 String school;

 Student(String name, int age, this.school) : super(name, age);

 @override
 void greet() {
 print('Hello, my name is $name and I am a student at $school.'); }
}
...
```

`Student` klassi `Person` klassidan meros oladi. `super()` chaqiruvida ota-ona klassning konstruktori chaqiriladi. `@override` annotatsiyasi bilan `greet()` metodi qayta yozildi.

#### 5. Sinflar O'rtasidagi Munosabatlar :

- Assotsiatsiya : Bir klass o'zining asosiy vazifasiga oid boshqa klasslar o'rtasidagi bog'liqlikni bildiradi.
- Agregatsiya : Bir klass boshqa klasslar to'plamini o'z ichiga oladi.
- Kompozitsiya : Bir klass boshqa klasslardan iborat bo'ladi va ularning hayot sikli bir-biriga bog'liq bo'ladi.

Dart dasturlash tilida klasslar va ob'ektlar bilan ishlash yuqoridagi usullar yordamida amalga oshiriladi. Ushbu tushunchalar platformalararo dasturlash uchun juda muhim bo'lib, turli xil ilovalar va tizimlar yaratishda foydalaniladi.

## **S151. Algoritmning asosiy turlari haqida ma'lumot.**

Algoritmning asosiy turlari dasturlashning asosiy qismlaridan biridir va har bir dasturchi algoritmni tushunish va yozishda ehtiyoj duyadigan bir qator konseptlarni o'rganishi kerak. Quyidagi algoritmning asosiy turlari mavjud:

1. Sekvensial (Izohlangan) algoritm : Bular odatda har bir qadamni izohlangan turdagi algoritmlar hisoblanadi. Yuqoridan pastga tartibda qadam qadam bilan amalga oshiriladi.
2. Conditional (Shartli) algoritm : Shartli algoritmlar, ma'lum sharoitlarda bir nechta amalni bajarishda ishlatiladi. Misol uchun, "agar x sharoiti bajarilgan bo'lsa, yni boshqa amalni bajaring" kabi.
3. Loop (Takrorlash) algoritmi : Takrorlash algoritmlari belgilangan shartga muvofiq bir amalni bir necha marta bajaradi. Bu, shartni tekshirish va shart bajarilguncha takrorlash imkonini beradi.
4. Recursive (Qaytaruvchi) algoritmi : Qaytaruvchi algoritmlar o'zini qaytaruvchi chaqirish (rekursiya) yordamida yaratiladi. Bu, bir amalni bajarish uchun o'zini takrorlash imkonini beradi.
5. Divide and Conquer (Bo'lib va Hakam) algoritmi : Bu algoritmlar masalani kichik dasturlarga bo'lib vaqtincha yechish uchun yoki dastur yozish uchun ishini asanlash uchun bo'lishadi.
6. Greedy (Larri) algoritmi : Greedy algoritmlari har bir qadamda eng yaxshi variantni tanlash prinsipi asosida ishlaydi. Har qadamda qo'llaniladigan variant tanlanadi, lekin bu, eng optimal natijaga olib kelishi shart emas.
7. Dynamic Programming (Dinamik dasturlash) algoritmi : Bu algoritmlar masalani o'zini qaytaruvchi chaqirish yordamida yechish, keyinchalik natijani saqlash va



qaytaruvchi chaqirish orqali ishlaydi. Bu, mashhur "top-down" va "bottom-up" yondashuvlarga ega.

8. Backtracking (Orqaga qaytish) algoritmi : Backtracking algoritmlari variantlar ketma-ketligini sinovlashda ishlatiladi va hal etish mumkin bo'lgan har bir variantni sinab ko'rishga asoslanadi.

Bu algoritmnining asosiy turlari bazi misollar bilan ta'minlanadi va amaliy dasturlashda ko'p mashhur va foydali usullardir. Har bir algoritm turining o'ziga xos foydalanishning va qo'llanishning ko'p variantlari mavjud, shuningdek, ko'p masalalar kelajakdagi dasturlash vazifalarini hal qilishda bir qatorda turli algoritmlarini kombinatsiyalarni talab qilishi mumkin.

**S153. Rasmda keltilgan grafni Dijkstra algoritmi asosida eng qisqa yo'lni topish misolida yeching va Dijkstra algoritmini ishlash tamoyilini tushuntiring.**

Algoritmning asosiy turlari quyidagilar:

1. Ketma-ket Algoritm (Sequential Algorithm) :

- Bu eng oddiy algoritm turi bo'lib, operatsiyalar ketma-ket bajariladi.
- Bir operatsiya yakunlangach, keyingi operatsiya bajariladi.
- Masalan, mashinada yuvish algoritmi - kir kiyimlarni to'plash, kir yuvish, quritish, joylashtirish.

2. Tarmoqlanuvchi Algoritm (Branching Algorithm) :

- Algoritm bajarilishining qaysi yo'nalishda davom etishi turli sharoitlarga bog'liq bo'ladi.
- Qaror qabul qilish operatorlari (if-else, switch) yordamida amalga oshiriladi.
- Masalan, telefon raqamini to'g'ri kiritish yoki kiritilmagan taqdirda xato xabarini chiqarish.

3. Takroriy Algoritm (Iterative Algorithm) :

- Algoritmning bir qismi ko'p marta takrorlanadi.
- Takrorlanish soni ma'lum yoki ma'lum bo'lmagan bo'lishi mumkin.
- Masalan, massivdagi elementlarni ajratish, sonlarni yig'ish, for, while, do-while operatorlari yordamida amalga oshiriladi.

4. Rekursiv Algoritm (Recursive Algorithm) :

- Algoritm o'z-o'zini chaqiradi.
- Masalani kichik qismlarga bo'lish va ularni yechish asosida ishlaydi.
- Masalan, faktorialini topish, Fibonachchi ketma-ketligini topish.

5. Parallel Algoritm (Parallel Algorithm) :

- Algoritmning turli qismlari bir vaqtning o'zida bajariladi.
- Masalani parallel ravishda yechish maqsadida ishlatilib, samaradorlikni oshiradi.
- Masalan, ko'p yadrolik protsessorlarda ishlash, grafik protsessorlardan foydalanish.

6. Taqsimiy Algoritm (Distributed Algorithm) :

- Algoritm turli kompyuterlar o'rtasida taqsimlangan holda bajariladi.
- Tarmoq tizimlarida, bulutli hisoblashlarda keng qo'llaniladi.
- Masalan, torrent-illovalar, blockchain tizimlar.

Ushbu algoritm turlari dasturlash va masalalarni yechishda keng qo'llaniladi. Har bir algoritm turi turli xil muammolarni hal qilishda samara beradi.

## S152. Qidrish algoritmlari va ularning bir-biridan farqi.

Qidrish algoritmlari muayyan ma'lumotlarni yoki ob'ektlarni izlash va topish uchun ishlatiladi. Qidrish algoritmlarining asosiy turlari quyidagilar:

### 1. Chiziqli Qidrish (Linear Search) :

- Eng oddiy qidrish algoritmi.
- Ketma-ket ravishda ro'yxatdagi har bir element tekshiriladi.
- Samaradorligi  $O(n)$  bo'lib, ro'yxat uzunligiga to'g'ridan-to'g'ri bog'liq.
- Qidirilayotgan element topilganda yoki ro'yxat oxiriga yetilganda algoritm to'xtatiladi.

### 2. Ikkilamchi Qidrish (Binary Search) :

- Tartiblangan ro'yxatlarda qidirish uchun ishlatiladi.
- Har bir qadamda qidirilayotgan element o'rtadagi element bilan taqqoslanadi.
- Agar element kichik bo'lsa, ro'yxatning birinchi yarimidan, katta bo'lsa, ikkinchi yarimidan qidiriladi.
- Samaradorligi  $O(\log n)$  bo'lib, ro'yxat o'lchamiga kamroq bog'liq.
- Ro'yxat tartiblangan bo'lishi shart.

### 3. Interpolatsiya Qidishi (Interpolation Search) :

- Ikkilamchi qidirish kabi, tartiblangan ro'yxatlarda qidirishda ishlatiladi.
- Qidirilayotgan element taxminiy joylashuvini hisoblab, shu joydan qidirishni boshlaydi.
- Samaradorligi  $O(\log \log n)$  bo'lib, ro'yxat to'g'ri taqsimlanganida eng samarali.
- Ro'yxat tartiblangan va taqsimlanishi bir tekis bo'lishi shart.

### 4. Eksponentsial Qidrish (Exponential Search) :

- Tartiblangan ro'yxatlarda ishlatiladigan algoritim.

- Qidirilayotgan element joylashuvini taxminiy topish uchun eksponentsial qadamlar orqali izlanadi.
- Samaradorligi  $O(\log n)$  bo'lib, Ikkilamchi qidirish kabi samarali.
- Ro'yxat tartiblangan bo'lishi shart.

Ushbu qidirish algoritmlarining asosiy farqi ularning samaradorligi va qo'llanish sohalarida namoyon bo'ladi. Qidirilayotgan element tashqi ko'rinishiga, ro'yxat tuzilishiga va taqsimlanishiga qarab, eng samarali algoritmlar tanlanadi.

Qidirish algoritmlari, boshqa ma'lumotlarning o'rtasidan kerakli ma'lumotni topish uchun ishlatiladi. Bundan tashqari, ular ma'lumotlar to'plamida, masalan, ro'yxatda yoki jadvalda kerakli ma'lumotni izlashda ham foydalaniladi. Bu, kompyuter dasturlashning juda keng qo'llaniladigan qismini tashkil etadi. Qidirish algoritmlari quyidagi turkumlarga bo'linadi:

1. Linear Search (Tartiblangan qidirish) : Bu algoritmlar, kerakli elementni ma'lumotlar to'plamida qidirish uchun foydalaniladi. Ro'yxatni boshidan boshlab to'laqonni bir-biriga solish orqali kerakli elementni topish uchun ishlatiladi. Uning murakkabligi  $O(n)$  bo'lib,  $n$  ro'yxat uzunligi.
2. Binary Search (Ikkilik qidirish) : Bu algoritmlar, ro'yxatda tartiblangan ma'lumotlar to'plamida yuqori xususiyatlarni izlaydi. Ushbu algoritmlar faqat tartiblangan ro'yxatlarda ishlaydi. Bu algoritmlar murakkabligi  $O(\log n)$  bo'lib,  $n$  ro'yxat uzunligi.
3. Hashing (Xashlash) : Bu algoritmlar, kerakli ma'lumotni kiritish va uni topish uchun ma'lumotni xashlaydi. Bu algoritmlar murakkabligi odatda  $O(1)$  bo'lib, ammo xashlovchi funktsiyani yaratish jarayonida vaqt ketadi.

Farq:

- Linear Search: Bu algoritmlar, to'laqonning har bir elementini tekshirish orqali kerakli elementni topadi. Bu odatda katta ro'yxatlar uchun noaniqlikka olib keladi.

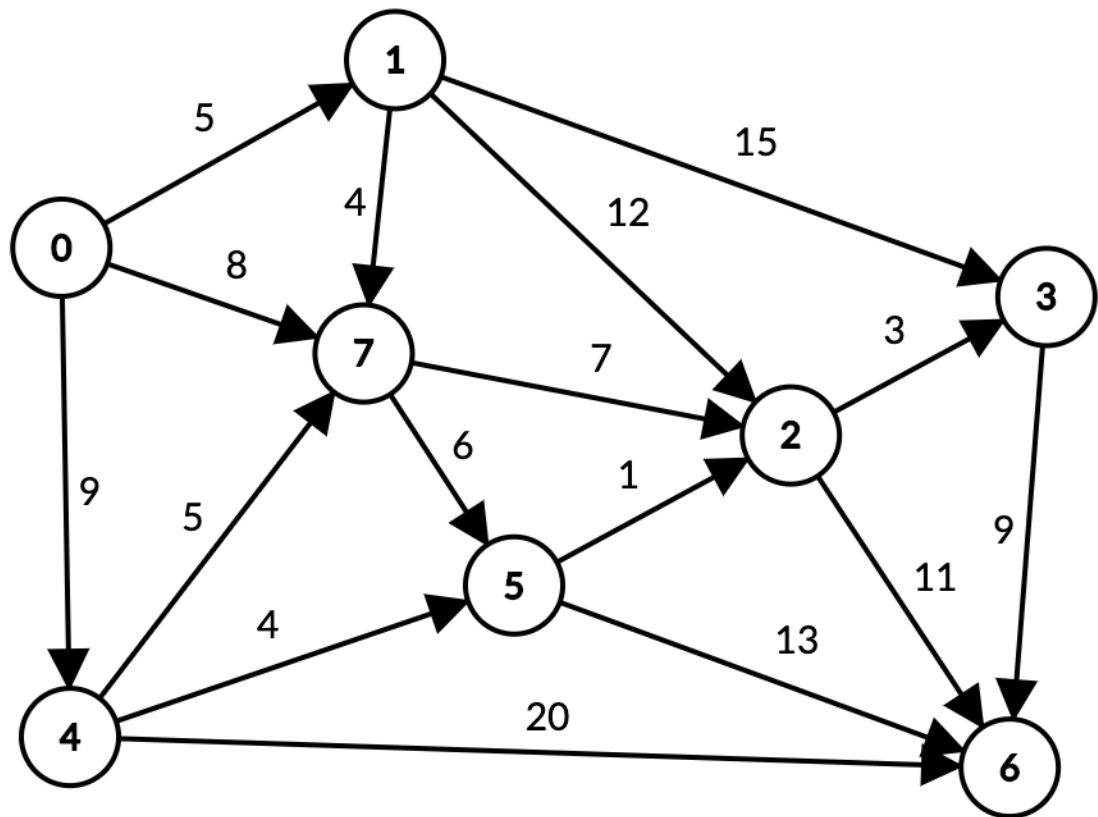
- Binary Search: Bu algoritim, ro'yxatni ikkilik qidirish prinsipi asosida qidiradi. Uning murakkabligi Linear Search dan kattaroq, ammo faqat tartiblangan ro'yxatlarda ishlaydi.

- Hashing: Bu algoritim, ma'lumotlarni boshqa ma'lumotlar to'plamiga bog'lash orqali kerakli ma'lumotni topish uchun ishlatiladi. Bu odatda  $O(1)$  murakkablikka ega, lekin xashlovchi funksiyaning aniqlikka ega bo'lishi kerak.

Algoritmning umumiy qoidalari quyidagicha:

- Har safar vertex'ni tekshirmoqchi bo'lganimizda, doim weight'i kichkina edge'li vertex'ni tanlaymiz.
- Vertex'ga kelganimizda, uning qo'shni vertex'larini tekshiramiz.
- Har bir qo'shni vertex uchun, boshlang'ich node'dan qo'shni node'gacha bo'lgan edge weight'lari yig'indisini hisoblab chiqamiz.
- Agar qo'shni node'ga hisoblangan weight yig'indisi undagi bor qiymatdan kichik bo'lsa, uning qiymatini yangilaymiz (**edge relaxation**).

Ishlash tartibini misolda ko'rib chiqamiz. Bizda quyidagi graph bor.



**Tekshirishni 0 vertex'dan boshlaymiz.** Tekshirish davomida boshlang'ich vertex'dan boshqa vertex'larga masofalarni `distTo[]` ga yig'amiz. Masofa deb aytdim, aslida **masofa o'lchovi edge weightlarining yig'indisi hisoblanadi.**

0 dan 1, 7 va 4 vertex'larga edge bor. Ular uchun 0 vertex'dan masofani hisoblaymiz.

v	distTo	edgeTo
---	--------	--------

0	0	—
---	---	---

1	5	0
---	---	---

2	—	
---	---	--

v	distTo	edgeTo
3	—	
4	9	0
5	—	
6	—	
7	8	0

0 dan unga bog'langan vertex'lardagi eng kichik masofa – 5, u 1 vertexda. **1 vertexdan davom ettiramiz.**

1 dan 3, 2, 7 vertex'ga masofa bor. Ular uchun boshlang'ich nuqta – 0 vertex'dan masofani hisoblaymiz.

v	distTo[]	edgeTo[]
0	0	—
1	5	0
2	$5 + 12 = 17$	1
3	$5 + 17 = 20$	1
4	9	0



v	distTo[]	edgeTo[]
5	—	
6	—	
7	8	0

0 -> 1 -> 7 masofasi = 9 chiqdi. Bizdagi 0 -> 7 masofa kichikroq – 8 ga teng. Shuning uchun 7 ga masofani yangilamaymiz. Faqat yangi masofa eskisidan kichik bo'lgandagina yangilanadi Bu kabi yangilashni **edge relaxation** deyiladi.

1 ni tekshirib bo'lganimiz uchun unga boshqa qaytmaymiz.

distTo qiymatlariga ko'ra, navbatdagi kichik masofa 7-vertex'da. **7-vertex'dan tekshirishni davom ettiramiz.**

7 dan 2 va 5 ga edge bor. Ular uchun boshlang'ich nuqta – 0 vertex'dan masofani hisoblaymiz.

v	distTo	edgeTo
0	0	—
1	5	0
2	<del>5 + 12 = 17</del> <b>8 + 7 = 15</b>	7
3	5 + 17 = 20	1
4	9	0

v	distTo	edgeTo
5	$8 + 6 = 14$	7
6	—	
7	8	0

2-vertex'ning masofasi yangilandi, sababi  $0 \rightarrow 1 \rightarrow 2$  dan ko'ra  $0 \rightarrow 7 \rightarrow 2$  yaqinroq ekan. 5 ga esa yangi qiymat qo'shildi (**edge relaxation**).

Navbatdagi kichik masofa – 4-vertexda. **4-vertex'dan tekshirishni davom ettiramiz.**

4 dan 5, 6, 7 ga edge bor. Ular uchun boshlang'ich nuqta – 0 vertex'dan masofani hisoblaymiz.

v	distTo	edgeTo
0	0	—
1	5	0
2	<del><math>5 + 12 = 17</math></del> $8 + 7 = 15$	7
3	$5 + 17 = 20$	1
4	9	0
5	<del><math>8 + 6 = 14</math></del> $9 + 4 = 13$	4

v	distTo	edgeTo
6	$9 + 20 = 29$	4
7	8	0

0 -> 7 -> 5 masofasi 14 ga teng edi, 0 -> 4 -> 5 = 13 chiqdi, demak 5-vertex uchun distTo dagi qiymatni yangilaymiz.

6 bo'sh edi, unga yangi qiymat 29 qo'shildi.

7-vertex'ga yo'l – 0 -> 4 -> 7 masofasi 14, bor qiymat 8 dan kichik. 7-vertex'ning masofasi 8 ligicha qoladi.

Navbadagi kichik masofa 5-vertex'da. **5-vertex'dan tekshirishni davom ettiramiz.**

5 dan 2 va 6 ga edge bor. Ular uchun boshlang'ich nuqta – 0 vertex'dan masofani hisoblaymiz.

v	distTo	edgeTo
0	0	—
1	5	0
2	<del><math>5 + 12 = 17</math></del> <del><math>8 + 7 = 15</math></del> <b><math>9 + 4 + 1 = 14</math></b>	5
3	$5 + 17 = 20$	1
4	9	0

v	distTo	edgeTo
5	<del>8 + 6 = 14</del> <b>9 + 4 = 13</b>	4
6	<del>9 + 20 = 29</del> <b>9 + 4 + 13 = 26</b>	4
7	8	0

0 dan 2 gacha masofa ( 0 -> 4 -> 5 -> 2 )  $9 + 4 + 1 = 14$  ga teng. 14 hozirgi qiymat 15 dan kichkina. 2 ning masofasini yangilaymiz (**edge relaxation**).

0 dan 6 gacha masofa ( 0 -> 4 -> 5 -> 6 )  $9 + 4 + 13 = 26$  ga teng. 26 hozirgi qiymat 29 dan kichik. 6 ning masofasini yangilaymiz (**edge relaxation**).

Navbadagi kichik masofa 2-vertex'da. **2-vertex'dan tekshirishni davom ettiramiz.**

2 dan 3 va 6 ga edge bor. Ular uchun boshlang'ich nuqta – 0 vertex'dan masofani hisoblaymiz.

v	distTo	edgeTo
0	0	—
1	5	0
2	<del>5 + 12 = 17</del> <del>8 + 7 = 15</del> <b>9 + 4 + 1 = 14</b>	5
3	<del>5 + 17 = 20</del> <b>9 + 4 + 1 + 3 = 17</b>	2
4	9	0

v	distTo	edge
5	<del>8 + 6 = 14</del> <b>9 + 4 = 13</b>	4
6	<del>9 + 20 = 29</del> <del>9 + 4 + 13 = 26</del> <b>9 + 4 + 1 + 11 = 25</b>	2
7	8	0

0 dan 3 gacha masofa ( 0 -> 4 -> 5 -> 2 -> 3 )  $9 + 4 + 1 + 3 = 17$  ga teng. 17 hozirgi qiymat 20 dan kichkina. 3 ning masofasini yangilaymiz. (**edge relaxation**).

0 dan 6 gacha masofa ( 0 -> 4 -> 5 -> 2 -> 6 )  $9 + 4 + 1 + 11 = 25$  ga teng. 25 hozirgi qiymat 26 dan kichkina. 6 ning masofasini yangilaymiz. (**edge relaxation**).

Navbadagi kichik masofa 3-vertex'da. **3-vertex'dan tekshirishni davom ettiramiz.**

3-vertex'da faqat bitta edge bor – 6-vertex'ga. 6 uchun boshlang'ich nuqta – 0 vertex'dan masofani hisoblaymiz.

v	distTo	edge
0	0	–
1	5	0
2	<del>5 + 12 = 17</del> <del>8 + 7 = 15</del> <b>9 + 4 + 1 = 14</b>	5
3	<del>5 + 17 = 20</del> <b>9 + 4 + 1 + 3 = 17</b>	2
4	9	0

v	distTo	edgeTo
5	<del>8 + 6 = 14</del> <b>9 + 4 = 13</b>	4
6	<del>9 + 20 = 29</del> <del>9 + 4 + 13 = 26</del> <b>9 + 4 + 1 + 11 = 25</b>	2
7	8	0

0 dan 6 gacha masofa ( 0 -> 4 -> 5 -> 2 -> 3 -> 6 )  $9 + 4 + 1 + 3 + 9 = 26$  ga teng. 26 hozirgi qiymat 25 dan katta. 6 ning masofasio'zgarmaydi.

Navbadagi kichik masofa 6-vertex'da. **6-vertex'dan tekshirishni davom ettiramiz.**

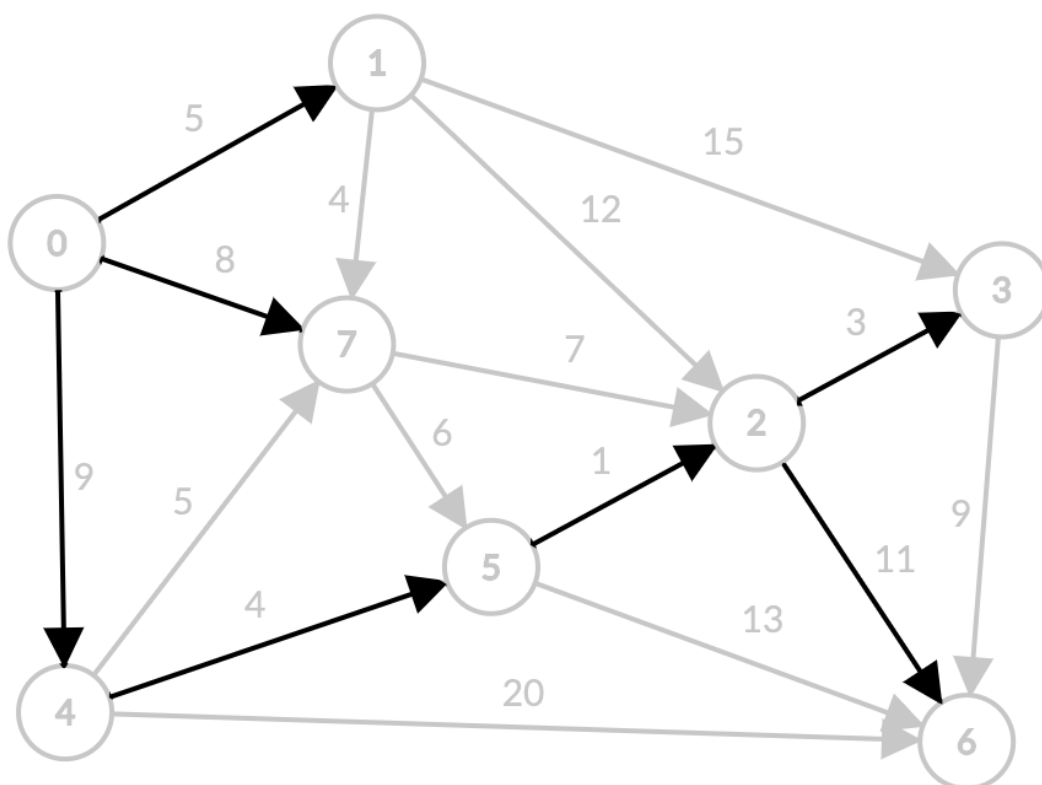
6-vertex'ning edge'lari yo'q, demak hisoblashlar tugadi. Yakuniy jadval quyidagicha bo'ladi.

v	distTo	edgeTo
0	0	—
1	5	0
2	14	5
3	17	2
4	9	0
5	13	4

v	distTo	edgeTo
6	25	2
7	8	0

edgeTo ma'lumotlari asosida path'larni yig'adigan bo'lsak qisqa path'lar quyidagicha bo'ladi:

- 0 dan 1 gacha: 0->1
- 0 dan 2 gacha: 0 -> 4 -> 5 -> 2
- 0 dan 3 gacha: 0 -> 4 -> 5 -> 2-> 3
- 0 dan 4 gacha: 0-> 4
- 0 dan 5 gacha: 0 -> 4 -> 5
- 0 dan 6 gacha: 0 -> 4 -> 5 -> 2-> 6
- 0 dan 7 gacha: 0 -> 7



Kod

```

function dijkstraSP(graph, start = 0) {
 // Collect edge reference
 const edgeTo = Array(graph.length).fill(null)

 // Collect distances
 const distTo = Array(graph.length).fill(Number.POSITIVE_INFINITY)

 // Priority queue is used to process graph
 const PQ = []

 // Start from 0 vertex. Starting point has 0 distance
 distTo[start] = 0
 PQ.push({ v: start, distance: 0 })

 while (PQ.length > 0) {
 // Get first (smallest) value from the priority queue
 const queueItem = PQ.shift()

 // Check if vertex has edges
 if (graph[queueItem.v]) {
 // Relax them
 graph[queueItem.v].forEach((edge) => relax(edge))
 }
 }

 // Edge relaxation
 function relax(edge) {
 // Get edge vertices
 const v = edge.get('from')
 const w = edge.get('to')

 // Check if new distance is smaller than current one
 if (distTo[w] > distTo[v] + edge.get('weight')) {
 // Update the distance
 distTo[w] = distTo[v] + edge.get('weight')
 }
 }
}

```



```

// Add reference
edgeTo[w] = v

// Find index of current edge vertex
const index = PQ.findIndex(item => item.v === w)
if (index !== -1) {
 // Update distance in PQ as well
 PQ[index].distance = distTo[w]
} else {
 // Add new item to PQ
 PQ.push({ v: w, distance: distTo[w] })
}

// Actually sorting is not necessary
// Code can process without sorting as well
PQ.sort((a,b) => a.distance - b.distance)
}
}

// Return references
return edgeTo
}

// Helper function to generate
function pathGenerator(edgeTo, start, end) {
 if (edgeTo[end] === undefined) {
 return
 }
}

const vertices = []
let current = end

while (current) {
 vertices.unshift(current)
}

```

```

 current = edgeTo[current]
 }

 if (vertices.length) {
 vertices.unshift(start)
 }

 return vertices
}

/
* USAGE
*/

// Sometimes we get edge list instead of graph
const weightedEdgeList = [
 [4,7], 5],
 [7,2], 7],
 [1,3], 15],
 [0,4], 9],
 [3,6], 9],
 [2,3], 3],
 [1,7], 4],
 [1,2], 12],
 [0,7], 8],
 [2,6], 11],
 [7,5], 6],
 [4,6], 20],
 [0,1], 5],
 [5,6], 13],
 [4,5], 4],
 [5,2], 1]
]

// If we have edge list, need to fill graph

```

```
const graph = []

// Assign vertices and weight to adjacent list graph
weightedEdgeList.forEach(item => {
 const vertices = item[0]
 const edge = new Map()

 // Add vertices
 edge.set('from', vertices[0])
 edge.set('to', vertices[1])

 // Add weight
 edge.set('weight', item[1])

 if (!graph[vertices[0]]) {
 graph[vertices[0]] = []
 }

 graph[vertices[0]].push(edge)
});

// Set starting point
const start = 0

// Process a graph to get vertex references
const paths = dijkstraSP(graph, start)

// Get path from start to 6th vertex
console.log(pathGenerator(paths, start, 6))

// Get path to 7th vertex
```

```
console.log(pathGenerator(paths, start, 7))
```

## **S154. Navbat va Dek ma'lumotlar tuzilmasi va ularning qo'llanilishi.**

Navbat (Queue) va Dek (Deque) ma'lumotlar tuzilmalari quyidagi xususiyatlarga ega:

### **1. Navbat (Queue) :**

- Ta'rifi : Navbat ma'lumotlarni ketma-ket joylashtirib, birinchi kiritilgan element birinchi chiqariladi (FIFO - First In First Out) tarzida ishlaydi.

- Asosiy amallar :

- Qo'shish (Enqueue) - element navbatning oxiriga qo'shiladi.

- Chiqarish (Dequeue) - navbatning boshi element olinadi.

- Bosh (Front) - navbatning boshidagi elementni ko'rish.

- Oxir (Rear) - navbatning oxiridagi elementni ko'rish.

- Qo'llanilishi :

- Jarayonlarni navbatga olish (e'lon qilish, buyurtmalar, mashinalar navbatga olish).

- Grafik interfeyslarda elementlarni navbatga olish.

- Tarmoq portlarida ma'lumotlarni navbatga olish.

### **2. Dek (Deque - Double Ended Queue) :**

- Ta'rifi : Dek ikki tomondan ham elementlarni qabul qiluvchi navbat tuzilmasidir. Elementlar boshdan ham, oxirdan ham qo'shilishi yoki olinishi mumkin.

- Asosiy amallar :

- Boshdan qo'shish (Push Front) - elementni navbatning boshiga qo'shish.

- Oxirdan qo'shish (Push Back) - elementni navbatning oxiriga qo'shish.

- Boshdan olib tashlash (Pop Front) - navbatning boshidagi elementni o'chirish.

- Oxirdan olib tashlash (Pop Back) - navbatning oxiridagi elementni o'chirish.

- Qo'llanilishi :

- Grafik interfeyslardagi oyna boshqaruvida (masalan, undo/redo amallarida).

- Labirintlar va o'yinlarda yo'llarni aks ettirish.

- Boshdan va oxirdan kiritish/olib tashlash amallariga ehtiyoj bo'lgan boshqa algoritmlar.

Navbat va Dek ma'lumotlar tuzilmalarining asosiy farqi ularning element qo'shish va chiqarish joyidir. Agar faqat boshdan yoki faqat oxirdan ishlash yetarli bo'lsa, navbatdan, agar har ikkala tomondan ham ishlash kerak bo'lsa, dekdan foydalanish maqsadga muvofiq.

"Navbat" va "dek" ma'lumotlar tuzilmasi yoki strukturalar, dasturlashda ma'lumotlarni tuzilgan holda saqlash uchun ishlatiladi. Ularning qo'llanilishi, ma'lumotlarni turli shakllarda tuzilgan holda saqlash va ularga murojaat qilishga imkon beradi. Quyidagi tariflar ulardan birinchi o'rindagi ma'lumot tuzilmasi va ularning qo'llanilishi haqida ma'lumot beradi:

#### 1. Navbat (Queue) :

- Tuzilmasi : Navbat (queue) tuzilmasi, birinchi kirdi, birinchi chiqariladi (First In, First Out - FIFO) qoidasi asosida ishlaydi. Bu tuzilma odatda boshqalarining boshqarish tizimi uchun ishlatiladi, masalan, fayllar uchun yuklab olish, amaliyotlar jadvalida amalga oshirilgan vazifalarni amalga oshirish, buyruqlarni qabul qilish va hokazolar ketma-ketligini saqlash kabi.

- Qo'llanilishi : Navbat ma'lumotlarni birinchi kirdi, birinchi chiqarish tartibida saqlaydi. Ma'lumotlar qo'shilganida, ular navbatning oxiriga joylashadi. Ma'lumotlar chiqarilganida, ular navbatning boshidan chiqariladi.

#### 2. Dek (Double-ended queue) :

- Tuzilmasi : Dek (deque) tuzilmasi, ma'lumotlarni ikkita joydan qo'llashga imkon beradi. Bu tuzilma ma'lumotlarni boshdan va oxiridan qo'llash uchun ishlatiladi.

- Qo'llanilishi : Dekda ma'lumotlar ikkita joydan qo'llaniladi: bosh va oxiridan. Ma'lumotlar qo'shilganida, ular boshga yoki oxiriga joylashadi. Ma'lumotlar chiqarilganida, ular ham boshdan, ham oxiridan chiqariladi.

Navbat va Dek strukturalari dasturlashda ko'p joyda foydalaniladi. Ular ma'lumotlar tuzilmasini yoki qatori chiqarilgan holda saqlash uchun qulay va samarali usullardir.

### **S155. Pufakchali saralash (Bubble sort) haqida tushuncha va uni ishlash algoritmi.**

Pufakchali saralash (Bubble Sort) - eng oddiy va klassik saralash algoritmlaridan biridir. U ro'yxatdagi elementlarni ketma-ket solishtirib, kattalarini ro'yxatning oxiriga, kichikroqlarini boshiga surib borish orqali ishlaydi.

Pufakchali saralash algoritmi :

1. Ro'yxatning birinchi va keyingi elementlarini solishtirish.
2. Agar birinchi element ikkinchidan katta bo'lsa, ularning o'rnini almashtirilib, kattasi ro'yxatning oxiriga suriladi.
3. Agar birinchi element ikkinchidan kichik bo'lsa, ular o'rin almashtirilmaydi va algoritmi keyingi juftlikka o'tadi.
4. Bu jarayon ro'yxatning oxirigacha davom etadi.
5. Agar bir aylanishda hech qanday almashinuv bo'lmasa, demak ro'yxat tartiblangan hisoblanadi va algoritmi to'xtatiladi.

Masalan, [5, 1, 4, 2, 8] ro'yxati uchun algoritmning ishlash ketma-ketligi quyidagicha bo'ladi:

1. [1, 5, 4, 2, 8] - birinchi va ikkinchi element almashtirilib, kattasi oxiriga surildi
2. [1, 4, 5, 2, 8] - uchinchi va to'rtinchi element almashtirilib, kattasi oxiriga surildi
3. [1, 4, 2, 5, 8] - to'rtinchi va beshinchi element almashtirilib, kattasi oxiriga surildi
4. [1, 4, 2, 5, 8] - bu aylanishda hech qanday almashinuv bo'lmadi, demak ro'yxat tartiblangan

Pufakchali saralash algoritmi samaradorligi :

- Eng yaxshi holatda (ro'yxat tartiblangan bo'lsa) -  $O(n)$
- Oddiy holatda -  $O(n^2)$



- Eng yomon holatda (ro'yxat teskari tartiblangan bo'lsa) -  $O(n^2)$

Pufakchali saralash algoritmining asosiy kamchiligi uning samaradorligining past bo'lishidir. Lekin uning soddaligi va qo'llanish sohalari (kichik o'lchamli ro'yxatlarni saralashda) sababli hozirgi kunda ham keng qo'llaniladi.

Pufakchali saralash (Bubble sort) algoritmi, qo'shimchalarni bittasidan boshlab o'rta yoki oxiriga qo'yish (taraqqiyot) orqali saralaydi. Har safar o'rtadagi ikkita elementni taqqoslash orqali, ularni o'rnini almashtiradi, agar birinchi element ikkinchisidan katta bo'lsa. Bu jarayon saralashdagi eng asosiy jarayonlardan biri hisoblanadi va odatda o'rtacha tovushli ravishda amalga oshiriladi.

Pufakchali saralash algoritmi quyidagi tartibda amalga oshiriladi:

1. Ro'yxatni boshidan boshlab bitta elementni olamiz.
2. Olgan elementni keyingi element bilan taqqoslaymiz.
3. Agar olingan element keyingi elementdan katta bo'lsa, ularni o'rnini almashtiramiz.
4. Keyingi elementni olingan element deb olamiz va keyingi elementga o'zgartiramiz.
5. Ro'yxat oxirigacha yuqoridagi jarayonlarni takrorlaymiz.
6. Qadamni takrorlash orqali, eng katta element oxiriga olib boriladi.
7. O'rtacha tartibda o'zgarishlar bo'lganidan so'ng, ro'yxat saralandi deb hisoblanadi.

Pufakchali saralash algoritmi odatda katta o'nliklarda bo'lsa ham eng yaxshi vaqt kompleksligi bo'lgan  $O(n^2)$  algoritmlardan biridir. Ammo, kichik ro'yxatlar yoki qiymatlarni saralashda qulay va ishonchli tanlov bo'lishi mumkin.

Quyidagi Kotlin kodida Bubble sort algoritmini ko'rsataman:

```
```kotlin
```

```
fun bubbleSort(arr: IntArray) {
```

```

val n = arr.size
for (i in 0 until n - 1) {
    for (j in 0 until n - i - 1) {
        if (arr[j] > arr[j + 1]) {
            // O'zgarishlarni amalga oshiramiz
            val temp = arr[j]
            arr[j] = arr[j + 1]
            arr[j + 1] = temp
        }
    }
}

fun main() {
    val arr = intArrayOf(64, 34, 25, 12, 22, 11, 90)
    bubbleSort(arr)
    println("Sorted array:")
    for (i in arr.indices) {
        print("${arr[i]} ")
    }
}

```

Ushbu kod saralash uchun Bubble sort algoritmini amalga oshiradi. Pufakchali saralash algoritmi orqali boshida berilgan ro'yxatni saralab chiqaradi.

S156. Merosxo'rlik tushunchasi va uning turlari. Polimorfizm.

Merosxo'rlik (Inheritance) va Polimorfizm - obyektga yo'naltirilgan dasturlash (OOP) paradigmasining asosiy tushunchalari hisoblanadi.

1. Merosxo'rlik (Inheritance) :

- Ta'rifi : Merosxo'rlik - ota-ona sinfnining xususiyatlari va metodlarini avlod sinfga meros qoldirishdir. Avlod sinfi ota-ona sinfnining barcha xususiyatlariga ega bo'ladi.
- Turlari :
 - Yagona merosxo'rlik : Avlod sinfi faqat bitta ota-ona sinfdan meros oladi.
 - Ko'p merosxo'rlik : Avlod sinfi bir nechta ota-ona sinflardan meros oladi.
- Afzalliklari :
 - Kodning qayta ishlatilishini ta'minlaydi.
 - Yangi sinf yaratishda vaqtni tejaydi.
 - Kod struktura va mantiqini osonlashtiradi.

2. Polimorfizm (Polymorphism) :

- Ta'rifi : Polimorfizm - bir xil nomli metodlarning har xil sinflar uchun turlicha amalga oshirilishidir. Polimorfizm merosxo'rlik orqali amalga oshadi.
- Turlari :
 - Statik polimorfizm (Overloading) : Bir xil nomli metodlar turli parametrlar bilan yoziladi.
 - Dinamik polimorfizm (Overriding) : Avlod sinfi ota-ona sinfining metodini o'zgartirib (qayta yozib) amalga oshiradi.
- Afzalliklari :
 - Kodning qayta ishlatilishini ta'minlaydi.
 - Kodni osonlashtiradi va modullilikni oshiradi.
 - Yangi vazifalar uchun kod yozishni qiyinlashtirmaydi.

Merosxo'rlik va polimorfizm OOPning asosiy tushunchalari bo'lib, ulardan foydalanish dasturlarni yaratishni osonlashtiradi, kod strukturasi aniqlashtiradi va ularning qayta ishlatilishini ta'minlaydi.

Merosxo'rlik (Polimorfizm) dasturlashda obyektlarning bir turdan bir qator amalni bajarishi uchun ishlatiladi, ammo ularga bir qator turdagi amalni bajarish imkoniyatini beradi. Bu, dasturchiga kodni qisqa va sodda qiladi va kodni ko'ngilga ishlab chiqishni osonlashtiradi.

Polimorfizmning turlari quyidagilardir:

1. Statik Polimorfizm : Bu turi o'z ichiga ayrim turlarga bo'lgan funksiyalarni yaratish bilan bog'liq. Bunda, dasturchi bir nechta funksiyalarni o'zlashtiradi va ularning nomlari va imzolari (parametrlar) farqli bo'lishi mumkin. Statik polimorfizm funksiyalarni muqobil joyda bir vaqtda amalga oshirishni ta'minlaydi.

2. Dinamik Polimorfizm : Bu turi turli obyektlar orqali amalga oshirilgan funksiyalarning o'z ichiga aynan shu obyektlar uchun turli turdagi amallarni bajarishga yordam beradi. Bu, turli klasslar orqali turli turdagi obyektlar uchun birlashgan interfeyslar, abstrakt klasslar yoki miras olish vaqti ichidagi funksiyalarni ishlatish bilan amalga oshiriladi.

Polimorfizmning bir mazmuni misol ko'rinishi quyidagi ko'dda ko'rsatilgan:

```
```kotlin
// Kotlin tilida klasslarni yaratish
open class Animal {
 open fun sound() {
 println("The animal makes a sound")
 }
}
```

```

class Dog : Animal() {
 override fun sound() {
 println("Woof woof")
 }
}

class Cat : Animal() {
 override fun sound() {
 println("Meow meow")
 }
}

fun main() {
 val animal1: Animal = Dog()
 val animal2: Animal = Cat()

 animal1.sound() // Woof woof
 animal2.sound() // Meow meow
}

```

Ushbu kodda, `Animal` klassi bor va undan boshqalar `Dog` va `Cat` klasslarini meros qilgan. `sound` funksiyasi har bir turdagi obyektlarning turiga qarab turli qismlar chiqaradi. Bu misolda, dinamik polimorfizm ko'rsatilgan, chunki har bir obyektning `sound` metodining o'zgartirilgan xususiyati bor va qaysi turdagi obyektни qabul qilganligi yo'qotilmagan.

### **S157. Binar daraxt turlari haqida tushuncha va ularning bir-biridan farqi.**

Binar daraxt - har bir tugunida maksimal ikkita farzand (o'ng va chap) bo'lgan daraxtdir. Binar daraxtlar turli xil ko'rinishlarda bo'lishi mumkin, ularning har biri o'ziga xos xususiyatlarga ega:

#### **1. Odiy Binar Daraxt (Binary Tree) :**

- Har bir tugunida maksimal 2 ta farzand (o'ng va chap) bo'ladi.
- Har bir farzand tugun mustaqil ravishda o'z qiymatiga ega bo'ladi.
- Farzandlar qiymati ota-ona tugunidan kichik yoki katta bo'lishi shart emas.

#### **2. Ikkilamchi Binar Qidirish Daraxtlari (Binary Search Tree - BST) :**

- Har bir tugunidagi o'ng farzand qiymati ota-ona tugun qiymatidan katta, chap farzand qiymati esa kichik bo'lishi shart.
- Bu xossasi tufayli BST-da qidirish, qo'shish va o'chirish operatsiyalari samarali amalga oshiriladi.
- BST-da barcha oraliq elementlar ham o'z o'rnini topadi.

#### **3. Muvozanatlangan Binar Qidirish Daraxtlari (AVL Tree) :**

- BST-ga o'xshash, lekin tugunlar muvozanatlangan bo'lishi shart.
- Har bir tugunning chap va o'ng farzandlarining balandligi orasidagi farq 1 dan oshmasligi kerak.
- Muvozanatlash operatsiyalari (o'ng/chap aylanma) orqali saqlanadi.
- AVL dapaxti dinamik ravishda o'zini muvozanatlantiradi.

#### **4. Qizil-Qora Daraxtlari (Red-Black Tree) :**

- BST-ga o'xshash, lekin har bir tugun qizil yoki qora tusda bo'ladi.
- Har bir tugunning chap va o'ng farzandlari hamda ildizi qizil bo'lishi mumkin emas.
- Har bir tugun uchun qora tugunlar yo'li bir xil bo'lishi shart.

- Qizil-Qora daraxtlari AVL daraxtlarga qaraganda muvozanatlashtirishda samaraliroq.

Ushbu daraxt turlari orasidagi asosiy farq ularning strukturasi va qidirish, qo'shish, o'chirish kabi operatsiyalarni bajarish samaradorligidadir. Bu xossalari ularning qo'llanilish sohasini belgilaydi.

Binar daraxtlar dasturlashda ma'lumotlarni organizatsiyalash va qidirish uchun ishlatiladi. Ular dasturlashning ko'p joylarida, masalan, algoritmik va ma'lumotlar tuzilmasi dasturlarida, matematikada, tarmoq va elektronika sohaslarida, amaliyotlar va qo'llanmalar dasturlashda ishlatiladi.

Binar daraxtlarning bir nechta tur bo'lib, ularning bir-biridan farqli xususiyatlari mavjud. Quyidagi binar daraxt turkumlari mavjud:

1. Og'izli (Unary) daraxt : Bu turdagi daraxtlar bitta bo'sh yoki bir qo'shniqli uzunlikdagi boshqa daraxtlardan iboratdir. Har bir daraxtning faqat bitta ulari (tugunlari) mavjud.

2. Ikkiylik (Binary) daraxt : Bu turdagi daraxtlar har bir tugun (yoki ustun) uchun ikki qo'shniqli uzunlikdagi boshqa daraxtlardan iboratdir. Har bir tugun (yoki ustun) ikki qo'shniqli bo'lib, ularning o'ng va chap qol tugunlari bo'lishi mumkin.

3. Balansli (Balanced) daraxt : Bu turdagi daraxtlarda, barcha uchunlar (tugunlar) oraliq'ida belgilangan oraliq balansli bo'lib, uning o'lchamining balandligi o'nlik turg'unlar bo'ylab ularning eng yuqori va eng pastki bo'ylab bir-biriga o'sish orqali belgilanadi.

4. Balanssiz (Unbalanced) daraxt : Bu turdagi daraxtlarda, barcha uchunlar (tugunlar) oraliq'ida belgilangan oraliq balanssiz bo'lib, bu, tugunlar oraliq'ida o'lchamning balandligining biror oraliq tugun balandligidan yuqori yoki past bo'lishi mumkin.

Binar daraxtlarning farqli turkumlari daraxtlarning tuzilish tartibini, ularning balansli yoki balanssizligini va ularning bir-biriga qanday ulanishini ifodalaydi. Balansli daraxtlar qidiruv algoritmlari uchun ko'plab foydalaniladi, chunki ularning o'lchami o'nlik tugunlar bo'ylab oshiriladi va bu qidiruv jarayonini yengillashtiradi. Balanssiz daraxtlar esa odatda qaror qabul jarayonlarida ishlatiladi, chunki ularning tuzilishi odatda tezda yaratiladi va qisqa vaqt ichida ishlatilishi mumkin.



## **S158. Directed va undirected graflar, Bog'langan va bog'lanmagan, Vaznli va vaznsiz graflar.**

Graflar turli xil ko'rinishlarda bo'lishi mumkin. Ularning asosiy turlari quyidagilar:

1. Directed (yo'naltirilgan) va Undirected (yo'naltirilmagan) graflar :

- Directed Graf : Har bir yon (edge) ma'lum bir yo'nalishga ega bo'ladi (masalan,  $a \rightarrow b$ ).

- Undirected Graf : Yon (edge) ikki tugun o'rtasidagi aloqani bildiradi, lekin u yo'naltirilmagan (masalan,  $a - b$ ).

2. Bog'langan (Connected) va Bog'lanmagan (Disconnected) graflar :

- Bog'langan Graf : Grafning har bir tuguni boshqa tuguni bilan biror yo'l orqali bog'langan bo'ladi.

- Bog'lanmagan Graf : Grafda alohida komponentlar (subgraflar) mavjud bo'lib, ular bir-biri bilan bog'lanmagan bo'ladi.

3. Vaznli (Weighted) va Vaznsiz (Unweighted) graflar :

- Vaznli Graf : Har bir yon (edge) ma'lum bir vaznga (qiymatga) ega bo'ladi.

- Vaznsiz Graf : Grafning barcha yonlari bir xil vaznga (masalan, 1) ega bo'ladi.

Graflarning ushbu turlari ularning qo'llanilish sohalari va algoritmlarda qo'llanilishini belgilaydi. Masalan, yo'naltirilgan graflar transportda, kompyuter tarmoqlarida qo'llanilsa, vaznsiz graflar grafik nazariya masalalarida qo'llaniladi. Bog'langan va bog'lanmagan graflar esa turli xil klaster tahlili, tarmoq topologiyasi kabi masalalarda qo'l keladi.

Graflar tarmoq topologiyasida ma'lumotlarni ko'rsatish uchun ishlatiladigan diskret strukturalar hisoblanadi. Graflar to'plamidagi elementlar "butun" (dugoh) va "yallig" (bo'g')lar deb ataladi. Bu elementlar orqali, obyektlar va ularning ulanishlari ko'rsatiladi.

Quyidagi turlarda graflar mavjud:

1. Directed (Yo'nalishli) va Undirected (Yo'nalishsiz) Graflar :

- Directed Graflar : Bu graflar uchun alohida tomonlar ko'rsatilgan. Ya'ni, alohida "bosh" va "oxir" qatnashuvchilar (bog'lanma) mavjud. Misol uchun, uchish-kelish tarmoqlari uchun yo'nalishli grafikani ko'rib chiqing, qayerda "bosh" va "oxir" tomonlar aniqlanadi.

- Undirected Graflar : Bu graflar uchun bosh va oxir yo'nalishi ma'qul emas. Uchish-kelish tarmoqlari uchun yo'nalishsiz grafikani ko'rib chiqing. Uchish-kelish bog'lanmalari qaysi yo'nalishda bo'lganligi ko'rsatilmagan.

## 2. Bog'langan (Connected) va Bog'lanmagan (Disconnected) Graflar :

- Bog'langan Graflar : Bu graflar uchun har bir bo'g' "bog'langan" hisoblanadi. Ya'ni, har ikki tugun o'rtasida bir yo'nalishli yo'lovchi bo'lib, har ikki tugun orasida alohida yo'nalishli yo'lovchi (yo'lni bor va kel yo'li) mavjud.

- Bog'lanmagan Graflar : Bu graflar uchun ba'zi tugunlar bog'lanmaydi, ya'ni ularga yo'nalishli yo'lovchi yo'q. Masalan, bir qismini bosh qismini bo'sh qoldirgan.

## 3. Vaznli va Vaznsiz Graflar :

- Vaznli Graflar : Bu graflarda har bir yo'nalishga mos vazn berilgan. Bu vazn odatda yo'lovchilarni birlashtirish uchun kerak bo'ladi, masalan, yo'lovchilarning uzunligi, hajmi, narxi yoki qiyinchilik darajasi bo'lishi mumkin.

- Vaznsiz Graflar : Bu graflarda bog'lanma yo'lovchilariga hech qanday vazn berilmagan. Ularning bog'lanma o'zgaruvchilariga qarab aniqlanadi.

Shunday qilib, graflar farkli xususiyatlarga ega bo'lishi mumkin va ularga qo'llanish maqsadiga qarab turli xil turlarda bo'lishi mumkin. Yangi struktura yoki algoritmlar yaratishda, graflarning turini va xususiyatlarini tushunish juda muhimdir.

### **S159. Tez saralash (Quick sort) haqida tushuncha va uni ishlash algoritmi.**

Tez saralash (Quick Sort) - eng mashhur va samarali algoritmlardan biri bo'lib, massivni saralashda keng qo'llaniladi. Algoritm quyidagi qadamlarda amalga oshiriladi:

Algoritm:

1. Pivot tanlash : Massiv ichidan tasodifiy yoki birinchi/oxirgi element sifatida pivot (ajratuvchi) element tanlash.
2. Partitionlash (bo'lish) : Massivni pivot element atrofida ikkita qismga bo'lish:
  - Pivot elementdan kichik elementlar chap tomonga,
  - Pivot elementdan katta elementlar o'ng tomonga joylashtiriladi.
3. Rekursiv chaqirish : Hosil bo'lgan chap va o'ng qismlarni alohida-alohida tez saralash algoritmi yordamida saralash:
  - Chap qism uchun tez saralash algoritmini rekursiv chaqirish,
  - O'ng qism uchun tez saralash algoritmini rekursiv chaqirish.
4. To'plash : Chap, pivot va o'ng qismlarni birlashtirib, butun massivni saralangan holatga keltirish.

Psevdokod:

...

```
function quickSort(array, low, high):
```

```
 if low < high:
```

```
 pi = partition(array, low, high)
```

```
 quickSort(array, low, pi - 1)
```

```
 quickSort(array, pi + 1, high)
```

```
function partition(array, low, high):
```

```
 pivot = array[high]
```

```
 i = low - 1
```

```
 for j = low to high - 1:
```

```

 if array[j] < pivot:
 i = i + 1
 swap array[i] and array[j]
 swap array[i + 1] and array[high]
 return i + 1
...

```

Tez saralash afzalliklari:

- Samarali - o'rtacha va eng yaxshi holatda  $O(n \log n)$  murakkablikka ega.
- O'z-o'zidan muvozanatlangan - massiv qanchalik katta bo'lsa, samaradorligi shunchalik yaxshi.
- Qo'shimcha xotira kerak emas - asl massivni o'zgartiradi.

Kamchiliklari:

- Eng yomon holatda (massiv to'liq teskari tartibda joylashgan)  $O(n^2)$  murakkablikka ega.
- Pivot tanlash noto'g'ri bo'lsa, algoritm samarasiz ishlashi mumkin.

Tez saralash (Quick sort) algoritmi, faqat bir qadamda bir elementni o'rniga olib, unga bir elementni o'rniga olib, o'zgarishlarni amalga oshiradi. Bu jarayon qisqa vaqt davomida amalga oshiriladi va odatda o'nliklarda yoki undan kichik ro'yxatlarni saralash uchun ishlatiladi. Quick sort, ortalama holatda  $O(n \log n)$  vaqt kompleksligiga ega bo'lgan, ammo eng yomon holatda  $O(n^2)$  bo'lishi mumkin.

Tez saralash algoritmi quyidagi tartibda ishlaydi:

1. Bir elementni "pivot" deb tanlash. Ushbu elementni saralash jarayonida asosiy belgilaydigan bo'ladi.

2. Saralash tartibidagi barcha elementlarni "pivot"ning chap tomoniga o'girish va uning o'ng tomoniga chiqarish.
3. "Pivot"ni xotirasiga joylashgandan so'ng, ro'yxat birlashtiriladi. "Pivot"ning chap tomonida bo'lgan elementlar o'zlarining chap tomonidagi elementlarga nisbatan kichik, o'ng tomonida bo'lgan elementlar esa katta bo'ladi.
4. Bu qadam "pivot"ning o'ng va chap tomonlariga alohida to'rtlik yaratish orqali takrorlanadi.
5. Tashqi yulduzchadan asosiy yulduzchaga qadar, hamma elementlar ortiqcha yoki kam emasligini tekshirish orqali saralash jarayoni takrorlanadi.

Quyidagi Kotlin kodida Quick sort algoritmini ko'rsataman:

```
```kotlin
```

```
fun quickSort(arr: IntArray, low: Int, high: Int) {  
    if (low < high) {  
        val pivotIndex = partition(arr, low, high)  
        quickSort(arr, low, pivotIndex - 1)  
        quickSort(arr, pivotIndex + 1, high)  
    }  
}
```

```
fun partition(arr: IntArray, low: Int, high: Int): Int {  
    val pivot = arr[high]  
    var i = low - 1  
    for (j in low until high) {  
        if (arr[j] < pivot) {  
            i++  
            val temp = arr[i]
```

```

        arr[i] = arr[j]
        arr[j] = temp
    }
}
val temp = arr[i + 1]
arr[i + 1] = arr[high]
arr[high] = temp
return i + 1
}

fun main() {
    val arr = intArrayOf(64, 34, 25, 12, 22, 11, 90)
    quickSort(arr, 0, arr.size - 1)
    println("Sorted array:")
    for (i in arr.indices) {
        print("${arr[i]} ")
    }
}

```

Ushbu kod saralash uchun Quick sort algoritmini amalga oshiradi. Algoritim asosiy qismida rekursiv tartib bilan elementlarni ixtiyoriy "pivot" elementi bilan taqsimlash (partitsiya) va keyingi tartiblash (rekursiv qismlar) jarayonlari bor.

S160. To'liq va tarqoq graf. Ochiq va yopiq mashrut. Ochiq va yopiq zanjir

To'liq graf va tarqoq graf

1. To'liq graf (Complete Graph) : Grafning barcha tugunlari o'zaro bog'langan bo'ladi, ya'ni har bir tugun boshqa barcha tugunlarga ulangandir. To'liq grafda n tugun bo'lsa, yon soni $n(n-1)/2$ ga teng bo'ladi.
2. Tarqoq graf (Sparse Graph) : Grafning tugunlari bir-biriga kam sonli yon (edge) orqali bog'langan bo'ladi. Tarqoq grafda yon soni tugun soniga nisbatan kichik bo'ladi.

Ochiq mashrut va yopiq mashrut

1. Ochiq mashrut (Open Path) : Grafda biror tugundan boshlanib, boshqa biror tugunda yakunlanuvchi mashrut. Ochiq mashrutda birinchi va oxirgi tugunlar bir-biridan farq qiladi.
2. Yopiq mashrut (Closed Path) : Grafda biror tugundan boshlanib, yana o'sha tugun bilan tugaydigan mashrut. Yopiq mashrutda birinchi va oxirgi tugunlar bir xil bo'ladi.

Ochiq zanjir va yopiq zanjir

1. Ochiq zanjir (Open Chain) : Grafda biror tugundan boshlanib, boshqa biror tugunda yakunlanadigan zanjir. Ochiq zanjirda birinchi va oxirgi tugunlar bir-biridan farq qiladi.
2. Yopiq zanjir (Closed Chain) : Grafda biror tugundan boshlanib, yana o'sha tugun bilan tugaydigan zanjir. Yopiq zanjirda birinchi va oxirgi tugunlar bir xil bo'ladi.

Ushbu tushunchalar graflar nazariyasi, algoritmlar va kompyuter fanining turli sohalarida keng qo'llaniladi. Masalan, to'liq va tarqoq graflar tarmoq topologiyasi, ochiq/yopiq zanjirlar esa, mashinasozlik va logistikada qo'l keladi.

Graflar diskret matematika va kompyuter dasturlashda ma'lumotlar tizimini tasvirlash uchun ishlatiladi. Ularni bir qator qarorlar (tugunlar) orqali bog'langan bo'lishi

mumkin. Graf elementlari (bo'g'lar) uchun bog'lanma turi, ularga ko'rsatilgan tomonlar (yoki yo'nalishlar), uzunligi, vazni va boshqa xususiyatlar bo'lishi mumkin. Quyidagi asosiy graf turkumlari mavjud:

1. To'liq va Tarqoq Graf :

- To'liq Graf : To'liq grafda, har ikki tugun (bo'g') uchun barcha tugunlar bilan bog'langan bo'lishi mumkin. Ya'ni, har ikki tugun uchun uchun qo'llanilgan bo'g'lar soni (elementlar soni) $(n * (n - 1)) / 2$ ga teng bo'ladi. To'liq graf matematikada jadvallar (matris) yordamida tasvirlanadi.

- Tarqoq Graf : Tarqoq grafda, har ikki tugun (bo'g') uchun bazi tugunlar bilan bog'langan bo'lishi mumkin. Bu turdagi grafning bog'lanma miqdori to'liq grafdan kam bo'ladi. Tarqoq graf yordamida yig'indilar (array) yoki ro'yxatlar yordamida tasvirlanadi.

2. Ochiq va Yopiq Mashrut :

- Ochiq Mashrut : Ochiq mashrutda, bir yo'lni (bog'ni) boshqalar bilan uyg'otish mumkin. Ya'ni, yo'lni bir qatordan boshqa qatorga o'tish mumkin, lekin qayta o'tish mumkin emas.

- Yopiq Mashrut : Yopiq mashrutda, bir yo'lni (bog'ni) boshqalar bilan uyg'otish mumkin va bir qatordan boshqa qatorga qayta o'tish (qayta yo'lni yaratish) mumkin.

3. Ochiq va Yopiq Zanjir :

- Ochiq Zanjir : Ochiq zanjir, grafning boshidan oxirigacha bir yo'lni (bog'ni) bor yo'lni taqdim etadi. Ya'ni, har bir tugun (bo'g') faqat bir marta ishlatiladi.

- Yopiq Zanjir : Yopiq zanjir, har bir tugun (bo'g') qayta ishlatilmasligi shartiga bog'lanadi. Buning natijasida, to'liq bo'lmagan bir yo'lni o'rnatish mumkin emas.

Shunday qilib, graf turklarining bir-biridan farqli xususiyatlari va ularning qanday maqsadlar uchun ishlatilishi kerakligi mavjud. Tasavvurlash va dasturlash jarayonlarida, graf turini va uchun iste'mol qilinadigan xususiyatlarni tushunish juda muhimdir.

S161. Algoritm tushunchasi va algoritmni to'liq qurish bosqichlari.

Algoritm tushunchasi:

Algoritm, bir nechta amallar qatoridan iborat tartiblangan jarayon bo'lib, uning maqsadi belgilangan vazifani bajarish uchun qadam-qadam qo'llaniladigan tartiblangan qo'llanma, dastur yoki qadam-qadam qo'llanadigan jarayonlarni ifodalaydi. Algoritm, problemani yechishning tartiblangan qadamini aniqlash uchun matematik modellar va qo'llanmalar bilan ishlaydi.

Algoritmni to'liq qurish bosqichlari:

1. Aniqlash : Muammo yoki vazifani tushunish va kerak bo'lgan natijani belgilash. Muammo tavsifi va talablarini ko'rib chiqish.
2. Dastlabki qadamni belgilash : Algoritmning boshlanishi uchun qadamni belgilash. Boshlash nuqtasini va kerak bo'lgan ma'lumotlarni ko'rsatish.
3. Qadamni tavsiflash : Har bir qadamni tavsiflash, ma'lumotlarni qabul qilish, ularni qaytarish va ulardan foydalanishni belgilash. Har bir qadamni to'g'ri bajarish uchun kerak bo'lgan amallarni aniqlash.
4. Qadamni bajarish : Har bir qadamni bajarish, alohida tartibda bajarilishi kerak bo'lgan amallarni belgilash. Muammoga oid ma'lumotlarni qabul qilib, amallarni bajarish va natijani olish.
5. Natijani tekshirish : Algoritm bajarildikdan keyin natijani tekshirish. To'g'ri bo'lganligini va talab etilgan natijani to'g'ri olishini tekshirish.
6. Tizimni to'liq qayta ko'rib chiqish : Algoritmda to'g'ri ishlashdan emasligini aniqlab, zarur o'zgartirishlarni kiritish va boshidan boshlab qayta sinash.

Algoritmning to'liq qurilishi uchun ushbu bosqichlar amalga oshiriladi, bu esa vazifani aniq tushunish, qadamlarni belgilash, ularni tavsiflash, bajarish, natijani tekshirish va optimallashtirishga imkon beradi.

Algoritm - bu amaliyotlarni bajarish uchun tartiblangan qadam-qadamlik qo'llanma. Algoritm, ma'lumotlar turlarini olish, ma'lumotlarni qayta ishlash va natijalarni chiqarishning o'rtasidagi qadamli amalni aniqlaydi. Algoritmni to'liq qurish uchun esa bir nechta bosqichlar mavjud:

1. Tushunchani aniqlash : Muammo yoki vazifani tushunish, aniq maqsadlarni tuzish va algoritmda kerakli amalni aniqlash.

2. Ma'lumotlarni to'plash : Amaliyot uchun kerakli ma'lumotlar to'plamini (masalan, kiritishlar, turli muammoni yechish uchun ma'lumotlar) olish.

3. Algoritmni yozish : Tushunchani va ma'lumotlarni olishni asosida algoritmni tuzish. Bu qadamda algoritmni dasturlash tili yoki dasturlash asoslari asosida yozish mumkin.

4. Kodni tekshirish : Yozilgan algoritmni tekshirish va to'g'ri ishlashi uchun uni sinovga olish. Bu jarayonda, xatoliklarni aniqlash va tuzatish keng qo'llaniladi.

5. Test qilish : Yozilgan algoritmni tushunchani qo'llanishini tekshirish uchun testlarni amalga oshirish. Bu testlar ma'lumotlarning kiritishidan chiqarishga qadar algoritmni amalga oshirishini tekshiradi.

6. Optimallashtirish : Algoritmni ishlash jarayonini va vaqt kompleksligini kengaytirish mumkin bo'lgan bo'lgan qismlarini identifikatsiya qilish va ularni optimallashtirish.

7. Tuzatish va takrorlash : Testlar orqali aniqlangan xatoliklarni tuzatish, algoritmni qulayligini va tezligini yaxshilash uchun takrorlash.

8. Hujjatlash : Algoritmni yozish jarayonida, shuningdek, uni ma'lum bir dasturni (yoki loyihani) amalga oshirish uchun qollanish uchun hujjatlarni tayyorlash.

Algoritmni to'liq qurish bosqichlari algoritmlarni ishga tushirish jarayonidagi muhim qadamatlardir. Har qadamda ishonchli va to'g'ri ishni ta'minlash uchun dastur tuzishdagi qat'iylik va dastlabki yozuvlar muhimdir.

S162. AVL tree haqida tushuncha va uning qo'llanilishi.

AVL tree (Adelson-Velskii va Landis tree)

AVL tree - mustaqil o'zini muvozanatlashtiradigan ikki yo'nalishli qidiruv daraxtlari sinfidir. U o'zini-o'zi muvozanatlash xususiyatiga ega bo'lib, bu uni yaxshi samaradorlikka ega qiladi.

Asosiy xususiyatlari:

- Har bir tugunning chapdaq va o'ngdagi ostog'i orasidagi balandlik farqi eng ko'pi 1 bo'lishi kerak.
- Har bir tugun uchun chapdaq ostog'i va o'ngdagi ostog'i balandliklarining farqi -1, 0 yoki 1 bo'lishi kerak.
- Agar balandlik farqi 2 yoki undan ortiq bo'lsa, daraxt muvozanatsiz hisoblanadi va uni muvozanatlash uchun aylanma amallar bajariladi.

Qo'llanilishi:

- Samarali ma'lumotlar tuzilmalari: AVL daraxtlari, ikki yo'nalishli qidirish, qo'shish va o'chirish amallarini bajarish uchun juda qulay.
- Kompyuter fanlari: AVL daraxtlari operatsion tizimlar, ma'lumotlar bazasi va boshqa kompyuter dasturlarida keng qo'llaniladi.
- Moliya va investitsiya: AVL daraxtlari ulkan ma'lumotlar to'plamlari uchun samarali indexlash yechimini taqdim etadi.
- Ommaviy axborot vositalari: Katta ma'lumotlar bazalarini boshqarish uchun AVL daraxtlaridan foydalaniladi.

AVL daraxtlari tezkor qidirish, qo'shish va o'chirish imkoniyatlarini taqdim etadi, shuning uchun ular ko'plab amaliy vazifalar uchun juda qulay.

AVL daraxti (Adelson-Velskiy va Landis daraxti) balanslangan binar daraxt hisoblanadi. Ushbu daraxtning asosiy maqsadi, daraxtdagi har bir tugun (uzlu) uchun o'sha tugun bo'yicha balans ko'rishni ta'minlashdir. Balanslanish, daraxtdagi har bir

tugun uchun o'ng va chap yig'indilarni (uchini) farq qilish uchun kerak bo'lgan qo'llanmalar (amaliyotlar) yordamida amalga oshiriladi.

AVL daraxtida, har bir tugun uchun "balans faktori" hisoblanadi, bu tugun o'ng va chap ustunlarining sonlarining farqi bo'ladi. Agar balans faktori 1 dan ortiq bo'lsa, tugun o'ng yuqoriga ko'rish uchun balanslanishi kerak bo'ladi. Agar balans faktori -1 dan kichik bo'lsa, tugun chap yuqoriga ko'rish uchun balanslanishi kerak bo'ladi.

AVL daraxtida tugunlar qo'shilish, o'chirish va qidirish amaliyotlari shunchaki odatda $O(\log n)$ vaqt kompleksligiga ega bo'ladi. Bu, daraxtdagi tugunlar soni (n) ga nisbatan juda yaxshi vaqt kompleksligi sifatida hisoblanadi.

AVL daraxti turli kompyuter ilovalarda, masalan, ma'lumotlar bazalarida va ishlab chiqarishda foydalaniladi. Ular ma'lumotlar bazalarini boshqarish, qidirish amaliyotlarini amalga oshirish va ma'lumotlar kiritish uchun qulay struktura sifatida ishlatiladi. Balanslangan AVL daraxtlari daraxtning taraqqiyotini saqlash va yig'indilarni o'rganishning oddiy va samarali usulini ta'minlashi bilan xos bo'ladi.

S163.Birlashtirib saralash (Merge sort) haqida tushuncha va uni ishlash algoritmi.

Birlashtirib saralash (Merge Sort) algoritmi

Birlashtirib saralash - bu bo'linib ketadigan va birlashtirib saraladigan (divide and conquer) algoritm turidir. U katta ro'yxatni kichik qismlarga bo'ladi, keyin ularni saralangan holda birlashtirib, oxirida butun ro'yxatni saralangan qiladi.

Birlashtirib saralash algoritmining ishlash prinsipi:

1. Bo'lish (Divide) : Kiritilgan ro'yxatni yarmidan ikki qismga bo'ling. Agar qism bitta element yoki bo'sh bo'lsa, to'xtatish, aks holda har bir qismni qayta bo'ling.
2. Saralash (Conquer) : Har bir qismni alohida sarang. Buning uchun yana "bo'lish" va "saralash" amallarini bajaring.
3. Birlashtirish (Combine) : Saralangan qismlarni birlashtirib, yangi saralangan ro'yxatni hosil qiling.

Algoritm:

...

```
function mergeSort(arr):
```

```
    if length(arr) <= 1:
```

```
        return arr
```

```
    mid = length(arr) / 2
```

```
    left_half = arr[0:mid]
```

```
    right_half = arr[mid:]
```

```

left_half = mergeSort(left_half)
right_half = mergeSort(right_half)

return merge(left_half, right_half)

function merge(left, right):
    result = []
    left_index = 0
    right_index = 0

    while left_index < length(left) and right_index < length(right):
        if left[left_index] <= right[right_index]:
            result.append(left[left_index])
            left_index += 1
        else:
            result.append(right[right_index])
            right_index += 1

    result += left[left_index:]
    result += right[right_index:]
    return result

```

Birlashtirib saralash algoritmi vaqt murakkabligi $O(n \log n)$ bo'lib, katta ma'lumotlar to'plamlari uchun juda samarali hisoblanadi. U o'zini-o'zi muvozanatlash xususiyatiga ega bo'lgan algoritmlar turkumiga kiradi.

Birlashtirib saralash (Merge sort) algoritmi, qismiyoq daraxtlarning saralanishi uchun ishlatiladi. Bu algoritmning asosiy qadami, saralanayotgan ro'yxatni ikki qismga

bo'lish va har bir qismni alohida saralashdir. Keyin, saralgan qismni birlashtirish orqali o'zaro moslikni ta'minlashdir.

Birlashtirib saralash algoritmi quyidagi tartibda ishlaydi:

1. Bo'yoq bo'lish : Saralanayotgan ro'yxatni o'rtasidan ikki qism bo'lib bo'yoq bo'lish. Bu qadamlar rekursiv ravishda takrorlanadi, har bir bo'yoq bo'lish jarayonida qismni yarim yarim qismlarga bo'lib bo'lish.
2. Saralash : Har bir yarim saralgan qismni saralash. Bu amalga oshirilgan kengaytirilgan saralash algoritmi rekursiv ravishda ishlaydi.
3. Birlashtirish : Saralgan yarim qismlarni birlashtirish. Buning uchun birlashtirish algoritmi ishlatiladi, u qatorning chapdan o'ngga bo'lgan elementlarni solishtirib, ularni birlashtirib, natijani yangi ro'yxatga yozadi.

Birlashtirib saralash algoritmi odatda $O(n \log n)$ vaqt kompleksligiga ega bo'lgan, shuningdek, odatda vaqt miqdori kichik o'lchamdagi va katta ro'yxatlarni saralash uchun samarali algoritm hisoblanadi.

Quyidagi Kotlin kodida Birlashtirib saralash algoritmini ko'rsataman:

```
```kotlin
```

```
fun mergeSort(arr: IntArray) {
 if (arr.size > 1) {
 val mid = arr.size / 2
 val leftArray = arr.copyOfRange(0, mid)
 val rightArray = arr.copyOfRange(mid, arr.size)
```

```
mergeSort(leftArray)
mergeSort(rightArray)
```

```
var i = 0
var j = 0
var k = 0
```

```
while (i < leftArray.size && j < rightArray.size) {
 if (leftArray[i] < rightArray[j]) {
 arr[k++] = leftArray[i++]
 } else {
 arr[k++] = rightArray[j++]
 }
}
```

```
while (i < leftArray.size) {
 arr[k++] = leftArray[i++]
}
```

```
while (j < rightArray.size) {
 arr[k++] = rightArray[j++]
}
}
```

```
fun main() {
 val arr = intArrayOf(64, 34, 25, 12, 22, 11, 90)
```

```
mergeSort(arr)
println("Sorted array:")
for (i in arr.indices) {
 print("${arr[i]} ")
}
}
`
```

Ushbu Kotlin kodida Merge sort algoritmi ko'rsatilgan. Algoritm qadamli bo'lib saralash jarayonini takrorlab chiqaradi va natijada saralgan ro'yxatni chiqaradi.

## **S164. Daraxt ma'lumotlar tuzilmasi atamalari. Red-black tree qo'llanilishi**

Red-black tree (Qizil-qora daraxt) balanslangan binar daraxt hisoblanadi, bu binar daraxt algoritmidagi bir qator muhim xususiyatlarga ega. Qizil-qora daraxtlar odatda ma'lumotlarni qidirish, kiritish va o'chirish operatsiyalari uchun ishlatiladi. Ular ma'lumotlarni tez va ishonchli tarzda saqlash, qidirishni o'rganish, ma'lumotlarga yig'indilarni qo'shish va o'chirish uchun juda samarali daraxtlar hisoblanadi.

Red-black tree qo'llanishiga oid muhim xususiyatlar:

1. Balanslanish : Qizil-qora daraxtlar balanslanadi, ya'ni har bir tugun (uzel) uchun o'ziga xos "rang" (qizil yoki qora) bor. Bu ranglar daraxtning balanslanishini ta'minlashda kerakli bo'ladi.
2. O'rtacha sarlavha vaqt kompleksligi : Red-black tree ning o'rtacha sarlavha vaqt kompleksligi  $O(\log n)$  bo'ladi. Bu, daraxtning uzunligiga nisbatan juda yaxshi vaqt kompleksligi sifatida hisoblanadi.
3. Raqamlar turg'unlar va ustunlarda saqlanadi : Red-black tree da ma'lumotlar turg'unlar va ustunlarda saqlanadi. Bu, qidiruv va qo'shimcha amallarni o'rganish uchun juda samarali va ishonchli hisoblanadi.
4. Daraxtni boshqarish operatsiyalari : Qizil-qora daraxtlar odatda kiritish, o'chirish, qidirish va boshqa asosiy daraxtni boshqarish operatsiyalari uchun ishlatiladi. Ular, ma'lumotlar bazalarini boshqarishda juda samarali bo'lib, ma'lumotlarni qidirish va tahrirlash jarayonlarini tezlashtirishadi.

Red-black tree qo'llanishining muhim xususiyatlari va boshqarish operatsiyalari kerakli ehtiyojlar bo'yicha dastur yaratishda ishlatiladi. Bu daraxtlar odatda ishonchli va ishonchsiz ilovalarda qidiruv va ma'lumotlarni boshqarish uchun ishlatiladi.

Daraxt ma'lumotlar tuzilmasi atamalari:

1. Ildiz (Root): Daraxtning eng yuqori tugunini ildiz deb ataladi.
2. Tugun (Node): Daraxtdagi har bir element tugun deb ataladi.
3. Barglar (Leaves): Daraxtdagi oxirgi tugunlar barglar deb ataladi.
4. Bolalar (Children): Har bir tugunning o'ng va chap ostog'i bolalar deb ataladi.
5. Ota (Parent): Har bir tugunning ustidagi tugun ota tugun deb ataladi.
6. Daraxt balandligi (Height): Daraxtdagi eng uzoq yo'l uzunligi, ya'ni ildizdan eng uzoq bargga o'tish uchun talab qilinadigan tugunlar soni.

### Red-black tree (Qizil-qora daraxt)

Red-black tree - o'zini-o'zi muvozanatlash xossasiga ega bo'lgan ikki yo'nalishli qidiruv daraxtlari sinfidir. U qizil va qora rangli tugunlardan iborat bo'lib, muvozanatlik xususiyatini saqlab turadi.

#### Red-black tree xususiyatlari:

- Har bir tugun yoki qizil yoki qora rangli bo'ladi.
- Ildiz doim qora rangli.
- Har bir barglar qora rangli.
- Agar tugun qizil bo'lsa, uning bolalari doim qora rangli bo'ladi.
- Ildizdan bargga qadar bo'lgan har bir yo'lda qora rangli tugunlar soni bir xil.

#### Red-black tree qo'llanilishi:

- Operatsion tizimlar: Red-black tree Linux yadrosida qidiruv tizimi sifatida ishlatiladi.
- Ma'lumotlar bazasi: Red-black tree ma'lumotlar bazasida ma'lumotlarni saqlash uchun keng qo'llaniladi.
- Grafik redaktorlar: Red-black tree rastarli va vektorli grafikalarda element saqlovchisi sifatida ishlatiladi.

- Kompyuter o'yinlari: Red-black tree o'yinlar jarayonida joylarni, qahramonlarni va boshqa elementlarni saqlashda qo'llaniladi.

Red-black tree algoritmi samarali qidirish, qo'shish va o'chirish amallarini bajaradi, shuning uchun u ko'plab amaliy vazifalar uchun yaxshi mos keladi.

## **S165. Binar daraxt haqida tushuncha va uning turlari.**

### **Binar Daraxt (Binary Tree)**

Binar daraxt - bu ma'lumotlar tuzilmasi bo'lib, har bir tugun faqat ikkita bolaga (o'ng va chap) ega bo'ladi. Binar daraxtning asosiy xususiyatlari quyidagilar:

1. Ildiz (Root): Binar daraxtning eng yuqori joylashgan tuguni ildiz deb ataladi.
2. Barglari (Leaves): Binar daraxtning barcha yakunlovchi tugunlari barglar deb ataladi.
3. Ichki tugunlar (Internal Nodes): Ildiz va barglar orasida joylashgan tugunlar ichki tugunlar deb ataladi.

### **Binar daraxt turlari:**

1. To'liq binar daraxt (Full Binary Tree):
  - Barcha ichki tugunlarning ikkita bolasi bo'ladi.
  - Barcha barglar bir xil chuqurlikka ega bo'ladi.
2. Takrorlanuvchan binar daraxt (Complete Binary Tree):
  - Barcha ichki tugunlarning ikkita bolasi mavjud bo'lib, faqat eng pastki darajadagi tugunlardagina kamroq bolalari bo'lishi mumkin.
  - Barcha barglar bir xil yoki deyarli bir xil chuqurlikka ega bo'ladi.
3. Muvozanatlangan binar daraxt (Balanced Binary Tree):
  - Har bir ichki tugunning ikki bolasi orasidagi balandlik farqi 1 dan oshmasligi kerak.
  - Bu muvozanatlilik shart-sharoitini saqlab turish uchun qayta tashkil etish amallarini bajarish kerak bo'ladi.

#### 4. Ikkilamchi binar daraxt (Binary Heap):

- Ikki turi mavjud: min-heap va max-heap.
- Min-heap - ildiz tugun eng kichik qiymatga ega.
- Max-heap - ildiz tugun eng katta qiymatga ega.

Binar daraxtlar ma'lumotlarni saqlash, qidirish, qo'shish va o'chirish kabi operatsiyalarda keng qo'llaniladi. Turli xil binar daraxt turlari orqali muvozanatlilik, tezkorlik va samaradorlik darajalarini boshqarish mumkin.

Binar daraxt, ma'lumotlarni ifodalash uchun ishlatiladigan diskret struktura hisoblanadi. Bu daraxtning har bir tugunida bitta qiymat va ikki yoki undan ko'p bala bo'lishi mumkin. Bu balar odatda "chap" va "o'ng" deb ataladi. Chap bala, o'ng bala bilan taqqoslanadi va ularning oid bo'lgan qo'shimcha axvol (agar ular bo'lsa) ham mavjud. Boshlang'ich tugun "bosh" tugun deb ataladi.

Binar daraxtlar quyidagi turlarga bo'lishi mumkin:

1. To'liq Binar Daraxt : To'liq binar daraxt har bir tugun uchun barcha yig'indilarni (balarini) o'z ichiga oladi. To'liq binar daraxtning har bir tuguni shaklida ifodalangan ma'lumotlar bo'lishi mumkin, aks holda, mavjud bo'lmagan balar ko'rsatilmaydi.

To'liq binar daraxt, algoritmik ma'lumotlar tuzilmasida, matematikada, tarmoq dasturlashda va boshqa ko'p sohalarda foydalaniladi.

2. Tasodifiy Binar Daraxt : Tasodifiy binar daraxt barcha tugunlar uchun ko'rsatilgan balar o'rtasidagi aloqani ta'minlash uchun kerakli bo'lishi mumkin. Bunda, chap bala har bir tugunga oid bo'lgan qiymat o'ng bala bo'lib, o'ng bala esa o'ng bala bo'lgan qiymat bo'lishi mumkin.

3. Balanslangan Binar Daraxt : Balanslangan binar daraxtning barcha to'rtliklari (uzlu tugunlar) balanslanadi, ya'ni har bir tugun uchun o'ng va chap yig'indilari soni o'zaro mos bo'ladi. Bu daraxt balanslangan bo'lib, qidiruv operatsiyalarini tezlashtirish uchun ishlatiladi.



4. Binar Max-Heap va Min-Heap : Max-Heap va Min-Heap daraxtlar dastlabki elementi (bosh element) maksimal qiymatga (Max-Heap) yoki minimal qiymatga (Min-Heap) ega bo'lgan balanslangan binar daraxtlardir. Max-Heap va Min-Heap lar faqat bitta o'zgaruvchi yordamida saralanganligi uchun qidiruv amaliyotlarida va prioritetli qatorlarda foydalaniladi.

Binar daraxtlar dasturlash, algoritmalar, ma'lumotlar tuzilmasi va boshqa sohalarda ma'lumotlarni yaxshilash uchun samarali strukturalar hisoblanadi. Har bir turining o'ziga xos xususiyatlari va foydalanish sohalarining mavzusiga qarab, mos ravishda ishlatiladi.

## **S166. Graf va daraxtlar farqi. Graf atamalari.**

Graflar va daraxtlar kompyuter ilovalarni yaratishda va ma'lumotlarni ifodalashda foydalaniladigan ikkita turli struktura hisoblanadi. Bunday strukturalarning farqlari va ularning alohida atamalari mavjud:

### **1. Graf :**

- Tushuncha : Graf, ma'lumotlar tizimini ifodalovchi diskret strukturadir. Uning asosiy qismi tugunlar (bo'g'lar) va ularga bog'langan yoki bog'lanmagan ustunlar (uzlar)dan iborat.
- Atamalar : Grafda har bir tugun (bo'g') ma'lumotlarni ifodalash uchun kerakli obyektlarni o'z ichiga oladi. Misol uchun, foydalanuvchi uchun tugunlar do'stlar, shaharlar yoki telefon aloqalari bilan bog'lanishi mumkin.
- Misollar : SMM (Social Media Marketing) tarmoqlarining aloqalari, shaharlar va ularga yo'nalish yuklari yoki har xil kommunikatsiya tizimlari grafning misollariga kiradi.

### **2. Daraxt :**

- Tushuncha : Daraxt, boshlang'ich tugun (bosh ustun) bilan ifodalangan va undan keyin bo'lgan boshqacha ustunlar (uzlar) orqali bog'langan diskret strukturadir. Daraxtlarda har bir tugun (uz) ko'ngilochar bo'lgan maksimal bitta bog'lanma (o'ng bala) va bitta yoki undan ko'p pastki boshqacha bog'lanma (chap bala) bo'lishi mumkin.
- Atamalar : Daraxtda har bir tugun ma'lumotlarni tuzish uchun kerakli obyektlarni o'z ichiga oladi. Masalan, barcha nomlari doirasida joylashgan va o'ng va chap bo'lgan bosh ustun shaxslar, shaharlar yoki foydalanuvchilar ro'yxati bo'lishi mumkin.
- Misollar : Kelajakdagi qurilmalar va ularga bog'liq kataloglar, ma'lumotlarni tuzishda foydalanilgan ro'yxatlar, dastur yoki dasturlarni yaratish uchun foydalanilgan narsalar ro'yxati daraxtlarning misollariga kiradi.

Asosiy farq graf va daraxtning alohida tuzilmalarida yashaydi. Graf barcha tugunlarni o'z ichiga oladi, boshqa so'z bilan aytganda, bog'lanishlarni chizish mumkin. Daraxt esa o'ng bala va chap bala bog'lanmalari bilan tuziladi va har bir tugun o'ng va chap bo'lgan bir qator ustunlar bilan bog'lanadi. Yangi bog'lanmalar faqat bosh ustun

(tugun) uchun to'g'ri yoki pastki o'ng bala yoki chap bala ustunlar orqali amalga oshiriladi.

## Graf va Daraxt farqlari

Graf (Graph) va Daraxt (Tree) ma'lumotlar tuzilmasi sifatida bir-biridan farq qiladi:

### 1. Tuzilish:

- Graf: Tugunlar (Nodes) va ularni bog'lovchi yog'inlar (Edges) yig'indisi.
- Daraxt: Yagona ildizga ega bo'lgan tugunlar to'plami bo'lib, har bir tugunning faqat bitta otasi (Parent) bo'ladi.

### 2. Bog'lanish:

- Graf: Tugunlar o'rtasida bir nechta yo'llar (Paths) bo'lishi mumkin.
- Daraxt: Ildizdan har bir bargga faqat bitta yo'l mavjud.

### 3. Sikl:

- Graf: Grafda sikl (Cycle) mavjud bo'lishi mumkin.
- Daraxt: Daraxtda sikl bo'lmaydi.

### 4. Bog'lanishlik:

- Graf: Graf to'liq (Connected) yoki bo'lingan (Disconnected) bo'lishi mumkin.
- Daraxt: Daraxt har doim to'liq bog'langan bo'ladi.

## Graf atamalari:

1. Tugun (Vertex/Node): Grafning har bir element (nuqta) tugun deb ataladi.
2. Yog'in (Edge): Tugunlarni bog'lovchi chiziq yog'in deb ataladi.

3. Og'irlik (Weight): Yog'inlarga berilgan qiymat og'irlik deb ataladi.
4. Qo'shni (Neighbor): Bitta yog'in orqali bog'langan tugunlar qo'shni tugunlar deb ataladi.
5. Darajasi (Degree): Har bir tugunning unga ulanayotgan yog'inlar soni tugun darajasi deb ataladi.
6. Yo'l (Path): Tugunlarni bir-biriga bog'lovchi yog'inlar ketma-ketligi yo'l deb ataladi.
7. Sikl (Cycle): Boshlanish va tugash tugunlari bir xil bo'lgan yo'l sikl deb ataladi.
8. Bog'langan (Connected): Agar har bir tugun boshqa tugunlarga yo'l orqali bog'langan bo'lsa, graf bog'langan deb ataladi.
9. Bo'lingan (Disconnected): Agar graf tugunlari bir-biridan ajratilgan bo'lsa, u bo'lingan graf deb ataladi.

Graflar va daraxtlar kompyuter fanining turli sohalarida keng qo'llaniladi, masalan, tarmoqlar, algoritmlar, ma'lumotlar bazasi va boshqalar.

## S167. Tanlab saralash (Selection sort) haqida tushuncha va uni ishlash algoritmi

Tanlab saralash (Selection sort) algoritmi, bir ro'yxatdagi eng kichik elementni topib, uningni ro'yxatning boshiga o'rnatib, keyinchalik qolgan qismni yana saralash jarayoniga qo'shib yuboradi. Bu jarayon qisqa vaqt davomida amalga oshiriladi, ammo eng kichik elementni topish uchun qidiruvning  $O(n^2)$  vaqt kompleksligiga ega.

Tanlab saralash algoritmi quyidagi tartibda ishlaydi:

1. Ro'yxatni boshlang'ich elementi bilan boshlang.
2. Boshlang'ich elementni eng kichik element deb tanla.
3. Eng kichik elementni boshlang'ich element bilan almashtir.
4. Qolgan ro'yxatni qoldiqlarini olish uchun birinchi bosqichni o'zgartir.
5. Qolgan ro'yxatni qoldiqlarini tanlab, birinchi elementni eng kichik element deb o'lchab qo'y.
6. Ro'yxatni keyinchalik birinchi element sifatida ko'rsatilgan joydan boshlab qayta boshlash.

Quyidagi Kotlin kodida Tanlab saralash algoritmini ko'rsataman:

```
```kotlin
```

```
fun selectionSort(arr: IntArray) {  
    val n = arr.size  
    for (i in 0 until n - 1) {  
        var minIndex = i  
        for (j in i + 1 until n) {  
            if (arr[j] < arr[minIndex]) {  
                minIndex = j  
            }  
        }  
        val temp = arr[minIndex]  
        arr[minIndex] = arr[i]  
        arr[i] = temp  
    }  
}
```

```

    }
}

fun main() {
    val arr = intArrayOf(64, 34, 25, 12, 22, 11, 90)
    selectionSort(arr)
    println("Sorted array:")
    for (i in arr.indices) {
        print("${arr[i]} ")
    }
}
...

```

Ushbu Kotlin kodida Tanlab saralash algoritmi ko'rsatilgan. Algoritm qadamli bo'lib saralash jarayonini takrorlab chiqaradi va natijada saralgan ro'yxatni chiqaradi.

Tanlab saralash (Selection Sort)

Tanlab saralash - eng sodda va intuitiv algoritmlardandir. U ma'lumotlar to'plamini tartiblashtirishning asosiy usullaridan biri hisoblanadi.

Algoritm:

Tanlab saralash algoritmi quyidagi asosiy bosqichlardan iborat:

1. Massivning eng kichik elementi (indeks `i`) topiladi.
2. Ushbu eng kichik element massivning `i`-indeksli elementi bilan almashtiriladi.
3. `i` indeksi 1 ga oshiriladi.

4. 1-3 qadam takrorlanadi, to barcha element saralab chiqilgunga qadar.

Qisqacha `pseudocode` ko'rinishida algoritm quyidagicha ko'rinishga ega:

...

```
function selectionSort(arr):
```

```
    n = length(arr)
```

```
    for i = 0 to n-1:
```

```
        minIndex = i
```

```
        for j = i+1 to n-1:
```

```
            if arr[j] < arr[minIndex]:
```

```
                minIndex = j
```

```
        swap(arr[i], arr[minIndex])
```

```
    return arr
```

...

Ishlash prinsipi:

1. Birinchi qadam: Massivning eng kichik elementi (indeks `i`) topiladi. Buning uchun massivning qolgan qismini (indeks `i+1` dan `n-1` gacha) aylanib chiqilib, eng kichik element topiladi va uning indeksi `minIndex` ga saqlanadi.
2. Ikkinchi qadam: Topilgan eng kichik element (indeks `minIndex`) massivning `i` - indeksli elementi bilan almashtiriladi.
3. Uchinchi qadam: `i` indeksi 1 ga oshiriladi.
4. Birinchi uchta qadam takrorlanadi, to barcha element saralab chiqilgunga qadar.

Murakkablik:

Tanlab saralash algoritmi ikki darajali $O(n^2)$ murakkablikka ega, ya'ni massivning uzunligi n bo'lsa, algoritmning ishlash vaqti n^2 ga proporsional bo'ladi.

Tanlab saralash algoritmi oson tushuniladi, lekin samaradorligi nisbatan past hisoblanadi. Shuning uchun agar to'plam katta bo'lsa, boshqa algoritmlardan foydalanish tavsiya etiladi.

S168. Binar heap ko‘rinishidagi ma’lumotlar tuzilmasi.

Binar heap, balanslangan binar daraxtning max-heap yoki min-heap shaklida ifodalanganiga o'xshash ma'lumotlar tuzilmasidir. Ushbu tuzilma, o'rtacha sarlavha vaqt kompleksligi $O(\log n)$ bo'lgan qidiruv va qo'shimcha amallar uchun juda samarali bo'lib, boshqa ma'lumotlar tuzilmalariga nisbatan juda ishonchli va ishonchsiz ilovalarda foydalaniladi.

Binar heapning max-heap shakli, ularning boshlang'ich elementi (bosh ustun) maksimal qiymatga ega bo'lgan balanslangan binar daraxt hisoblanadi. Min-heap shakli esa boshlang'ich element minimal qiymatga ega bo'lgan balanslangan binar daraxtni ifodalaydi.

Binar heapning xususiyatlari:

1. Balanslangan daraxt : Binar heapning qidiruv va qo'shimcha amallar uchun kerakli vaqt kompleksligini ta'minlash uchun har bir ustun (uz) ustiga bir boshqa ustun (uz) yoki pastki (chap) bala bog'langan.
2. O'rtacha sarlavha vaqt kompleksligi : Binar heapning o'rtacha sarlavha vaqt kompleksligi $O(\log n)$ bo'ladi, bu daraxtning uzunligiga nisbatan juda tez amalga oshirilishi demakdir.
3. Boshqa daraxtlarga nisbatan yorqin foydalanish : Binar heap qidiruv va qo'shimcha amallar uchun yorqin foydalanish uchun ishlatiladi. Chunki ularning o'rtacha sarlavha vaqt kompleksligi juda kichik bo'lib, ularni katta miqdorda ma'lumotlarni saralash uchun ham foydalanish mumkin.

Binar heapning ma'lumotlar tuzilmasi esa odam o'qi, uynash va boshqa ilovalarda qidiruv, prioritetli qatorlar va statistik analizlar uchun foydalaniladi. Binar heapning algoritmlari dastlabki, hajmdagi vaqt kompleksligi kichik bo'lgan ilovalarda ishlatiladi.

Binar Heap (Binary Heap)

Binar Heap ma'lumotlar tuzilmasi bo'lib, to'liq ikkilamchi daraxt (Complete Binary Tree) shaklida ifodalanadi. Binar Heap ikki turga bo'linadi:

1. Min Heap - ildiz tugun (root node) qiymat jihatdan eng kichik bo'ladi.
2. Max Heap - ildiz tugun (root node) qiymat jihatdan eng katta bo'ladi.

Asosiy xususiyatlari:

1. Tuzilish: Binar Heap to'liq ikkilamchi daraxt sifatida aks ettiriladi.
2. Joylashishi: Elementlar daraxt ko'rinishida emas, balki massiv (array) ko'rinishida saqlanadi.
3. Tartiblash: Min Heap uchun ildiz element eng kichik, Max Heap uchun ildiz element eng katta qiymatga ega bo'ladi.

Massivdagi joylashish:

Binar Heap massiv ko'rinishida saqlanadi. i -indeksli tugun uchun:

- Ota (Parent) tuguni: $\text{floor}(i/2)$
- Chap farzand (Left child): $2*i$
- O'ng farzand (Right child): $2*i + 1$

Asosiy operatsiyalar:

1. Insert (Qo'shish): Yangi element oxiriga qo'shiladi va so'ng Heap xossalarini saqlash uchun "yuqoriga" ko'chiriladi.
2. Extract Min/Max (Olib tashlash): Ildiz tugun (eng kichik/katta qiymatli) olib tashlanadi va Heap xossasini saqlash uchun "pastga" ko'chiriladi.
3. Heapify (Qayta tashkil qilish): Mavjud massivni Heap xossasini qanoatlantirish uchun qayta tashkil qilish.

Murakkablik:

- Insert: $O(\log n)$
- Extract Min/Max: $O(\log n)$
- Heapify: $O(n)$

Binar Heap ma'lumotlar tuzilmasi algoritmlarda, xususan, sort qilish, eng qisqa yo'l, eng katta/kichik element topish kabi masalalarni yechishda qo'llaniladi.

S169. Algoritmning xossalari.

Algoritmning xossalari quyidagi:

1. Aniqlik : Algoritm o'ziga xos vazifalarni to'liq va aniq shaklda bajarishi kerak.
2. Aniq amaliyotlar : Algoritm barcha qadamlarini (har bir amaliyotni) aniq va tushunarli shaklda bajarishi kerak
3. Yuqori samarali : Algoritmning amaliyotlarini bajarish jarayoni samarali bo'lishi kerak. Buning ma'nosi, amaliyotlar tez va kuchli bo'lishi kerak.
4. Oson tushunilishi : Algoritmning tushunchalari oson va oqibatda tushunilishi kerak. Bu, algoritmni o'qish va uning natijalarini tushunishni osonlashtiradi.
5. Yuqori ishonch : Algoritmning to'g'ri va qat'iy ishlashiga ishonch hosil qilinishi kerak. Bu, algoritmning to'g'ri natijalarni bermasdan ma'lumotlarni ishlashni boshlashiga ishonchning ko'rsatkichidir.
6. Keng doira : Algoritmning foydalanishning keng doirasi kerak. Ya'ni, uning uyg'unligi va ko'p turli mashg'ulotlarda ishlatilishi mumkin bo'lishi kerak
7. Yuqori darajada optimallashtirish : Algoritmni optimallashtirish va uning amaliyotlarini tezlashtirish imkoniyatiga ega bo'lishi kerak. Bu, algoritmlar yoki ma'lumotlar tuzilmasining ishlashining boshqa yechimlarga nisbatan yuqori darajada samarali bo'lishini ta'minlaydi.

Algoritmlar umumiy ravishda matematik, kompyuter ilovalar, ma'lumotlar tuzilmasi va boshqa sohalarda qo'llaniladi. Ular samarali, aniq va yengil ishlashadi, va ko'p mashg'ulotlar uchun qo'llanishga ega. Algoritmlar kompyuter dasturlash, hisoblash va boshqa sohalarda asosiy qo'llaniladigan vositalardir.

Algoritmning Xossalari

Algoritm - ma'lum bir muammoni yechish uchun qo'llaniladigan, aniq va lo'nda talablardan iborat bo'lgan algoritmik jarayon. Algoritmlar o'zining bir qator xossalari ega:

1. Aniqlik (Definiteness):

- Algoritm har bir qadami aniq va lo'nda bo'lishi kerak.
- Algoritm har doim tushunarli va bir xilda bajarilishi lozim.

2. Hisoblanuvchanlik (Computability):

- Algoritm ma'lum bir sonli qadamlar orqali bajarilishi kerak.
- Algoritm cheksiz davom etmasligi lozim.

3. Natijalilik (Effectiveness):

- Algoritm berilgan muammoni aniq va to'g'ri yechishi kerak.
- Algoritm ma'qul vaqt va resurslarda ishlashi lozim.

4. Universallik (Generality):

- Algoritm muammoning barcha holatlari uchun mos kelishi kerak.
- Algoritm maxsus holatlar uchun emas, balki umumiy holatlar uchun yozilishi lozim.

5. Abstraksiyalanish (Abstraction):

- Algoritm muammoning mohiyatini aks ettirishi kerak.
- Algoritm qo'shimcha tafsilotlardan xoli bo'lishi lozim.

6. Modullilik (Modularity):

- Algoritm logik va funktsional jihatdan mustaqil bo'lgan qismlardan iborat bo'lishi kerak.
- Algoritm osonlik bilan o'zgartirilishi va yangilana olishi lozim.

7. Etakchi dizayn (Primary Design):

- Algoritm yechim uchun eng yaxshi va samarali yondashuv bo'lishi kerak.
- Algoritm muammoning o'ziga xos xususiyatlarini hisobga olishi lozim.

Ushbu xossalar algoritmni samarali, tushunarli va optimal ekanligini ta'minlaydi.
Ularni hisobga olish algoritmlarga qo'yiladigan asosiy talablar hisoblanadi.

S170. Binar qidiruv daraxti ma'lumotlar tuzilmasida daraxtdagi ma'lumotlar bilan ishlash.

Binar qidiruv daraxti, ma'lumotlar tuzilmasida o'rtacha sarlavha vaqt kompleksligi $O(\log n)$ ga ega bo'lgan samarali qidiruv uchun ishlatiladigan struktura hisoblanadi. Ushbu daraxt boshqa ma'lumotlar tuzilmalariga qaraganda, qidiruv, qo'shimcha, o'chirish va ma'lumotlarni tahrirlash jarayonlarini boshqarishda ishlatiladi.

Binar qidiruv daraxti quyidagi xususiyatlarga ega:

1. Balanslangan daraxt : Binar qidiruv daraxti balanslangan bo'lib, har bir tugun uchun o'ng va chap yig'indilari soni o'zaro mos bo'ladi. Bu daraxtning qidiruv va boshqa amaliyotlar uchun tez va samarali bo'lishini ta'minlaydi.
2. O'rtacha sarlavha vaqt kompleksligi : Binar qidiruv daraxti o'rtacha sarlavha vaqt kompleksligi $O(\log n)$ ga ega bo'lgani uchun, qidiruv amaliyotlari juda tez va samarali bo'ladi.
3. Ma'lumotlar bilan ishlash : Binar qidiruv daraxtida har bir tugun (uz) ma'lumotlarni ifodalash uchun kerakli ma'lumotlar saqlanadi. Bu ma'lumotlar odatda qidiruvni amalga oshirish uchun ishlatiladi. Masalan, qidiruv amaliyoti ketma-ketligi orqali boshlanadi va ma'lumotlar bilan solishtiriladi.
4. O'rtacha sarlavha vaqt kompleksligi : Binar qidiruv daraxti o'rtacha sarlavha vaqt kompleksligi $O(\log n)$ ga ega bo'lgani uchun, qidiruv amaliyotlari juda tez va samarali bo'ladi.

Binar qidiruv daraxti barcha ma'lumotlar tuzilmasining boshqarish tizimlarida qidiruv uchun keng doirada qo'llaniladi. U boshqa ma'lumotlar tuzilmasidagi ma'lumotlar bilan ishlash uchun samarali va tez amaliyotlarni ta'minlaydi.

Binar Qidiruv Daraxti (Binary Search Tree) - Ma'lumotlar bilan Ishlash

Binar Qidiruv Daraxti (Binary Search Tree - BST) ma'lumotlar tuzilmasida elementlar ikkilamchi daraxt ko'rinishida saqlanadi. BST-da ma'lumotlar saqlash va ular bilan ishlash quyidagi asosiy operatsiyalardan iborat:

1. Qo'shish (Insert):

- Yangi element qiymatiga qarab daraxtda tegishli o'ringa joylashtiriladi.
- Agar element mavjud bo'lsa, u almashtirilib qo'yiladi.
- Murakkablik: $O(\log n)$

2. O'chirish (Delete):

- O'chirilayotgan element daraxtdan olib tashlanadi.
- Elementni olib tashlash uchun 3 holat mavjud:
 1. Barglar (leaf node)
 2. Faqat bir farzand
 3. Ikki farzand
- Murakkablik: $O(\log n)$

3. Qidirish (Search):

- Qidirilayotgan element qiymatiga qarab daraxtda tegishli o'ringa borib topiladi.
- Agar element mavjud bo'lsa, uning joylashgan tuguni qaytariladi.
- Murakkablik: $O(\log n)$

4. Minimal/Maksimal element topish:

- Minimal element - chap pastki burchakdagi,
- Maksimal element - o'ng yuqori burchakdagi element.
- Murakkablik: $O(\log n)$

5. Takroriy aylanish (Traversal):

- Daraxt bo'ylab ma'lumotlarni ketma-ket ko'rib chiqish.
- Asosiy aylanish turlar: preorder, inorder, postorder.
- Murakkablik: $O(n)$

BST-da operatsiyalar samarali bajariladi, chunki elementlar tartiblangan holda saqlanadi. Murakkabligi logarifmik ($O(\log n)$) bo'lgani uchun BST katta hajmdagi ma'lumotlar bilan ishlashga xos.

S171. Kiritib saralash (Insertion sort) haqida tushuncha va uni ishlash algoritmi.

Kiritib Saralash (Insertion Sort) - Tushuncha va Algoritmi

Kiritib Saralash (Insertion Sort) - bu oddiy va samarali bo'lgan tartiblash algoritmlari oilasidan biri. Bunda elementlar orasidagi bog'liqlikni hisobga olgan holda ketma-ket tartiblanadi.

Kiritib Saralash Algoritmi:

1. Ikkinchi elementdan boshlab, ketma-ket elementlarni olib, ularni to'g'ri o'ringa kiritish.
2. Har bir element uchun, uni oldindagi tartiblanayotgan qismda to'g'ri o'ringa kiritish.
3. Barcha elementlar to'g'ri o'ringa kiritilgunga qadar algoritmni davom ettirish.

Algoritm Pseudo-kodi:

```

function insertionSort(arr):

    for i = 1 to length(arr):

        key = arr[i]

        j = i - 1

        while j >= 0 and arr[j] > key:

            arr[j+1] = arr[j]

            j = j - 1

        arr[j+1] = key

    return arr

```

Ishlash Prinsipi:

1. Birinchi element avtomatik ravishda tartiblanadi.
2. Keyingi elementlar orasidan eng kichigi topiladi va birinchi element bilan almashtiriladi.

3. Endi ikkinchi element uchun, u orasidagi elementlardan eng kichigini topib, u bilan almashtirish davom etadi.

4. Ushbu jarayon oxirigacha davom etadi.

Murakkablik Tahlili:

- Eng yaxshi holat: $O(n)$ - massivdagi elementlar allaqachon tartiblanagan bo'lsa.
- O'rtacha holat: $O(n^2)$ - massivdagi elementlar tasodifiy tartiblangan bo'lsa.
- Eng yomon holat: $O(n^2)$ - massivdagi elementlar teskari tartibda bo'lsa.

Kiritib Saralash oddiy, qulay va samarali algoritm bo'lib, kichik hajmdagi ma'lumotlar uchun yaxshi natija beradi. Ammo katta hajmdagi ma'lumotlar uchun boshqa samaraliroq algoritmlar mavjud.

Kiritib saralash (Insertion sort) algoritmi, ro'yxatdagi har bir elementni joyiga kiritish orqali saralashni amalga oshiradi. Uning asosiy qadamlari quyidagicha:

1. Ro'yxatni boshlang'ich holatda qabul qilamiz.
2. Keyinchalik, har bir elementni o'z joyiga joylashtirish jarayonini bajarish uchun bir necha bosqichlar takrorlanadi.
3. Elementni joyiga kiritish jarayonida, oldingi elementlarga qaraganda, yuqori elementni to'xtatib, keyinchalik elementni joyiga joylashtiriladi.
4. Bu jarayon barcha elementlar qatorini chiziqlash uchun amalga oshiriladi.

Insertion sort algoritmi quyidagi Kotlin kodida ifodalangan:

```
```kotlin
```

```
fun insertionSort(arr: IntArray) {
```

```
 val n = arr.size
```

```
 for (i in 1 until n) {
```

```
 val key = arr[i]
```

```
 var j = i - 1
```

```

// Oldingi elementlarni keyingi elementga nisbatan tekshirish
while (j >= 0 && arr[j] > key) {
 arr[j + 1] = arr[j]
 j--
}
arr[j + 1] = key
}
}

fun main() {
 val arr = intArrayOf(64, 34, 25, 12, 22, 11, 90)
 insertionSort(arr)
 println("Sorted array:")
 for (i in arr.indices) {
 print("${arr[i]} ")
 }
}

```

Ushbu Kotlin kodida Insertion sort algoritmi ko'rsatilgan. Algoritm har bir elementni joyiga kiritish orqali saralashni amalga oshiradi. Natijada saralgan ro'yxat chiqariladi.

## S172. “Bo‘lib tashla va hukumronlik qil” metodi toifasiga kiruvchi algoritmlar.

### "Bo'lib tashla va hukumronlik qil" Metodi Algoritmlar

"Bo'lib tashla va hukumronlik qil" (Divide and Conquer) - bu algoritmlar sinfi, murakkab masalalarni sodda va mustaqil bo'laklarga bo'lib, ularni alohida-alohida yechish orqali yechimga erishishni nazarda tutadi.

Bu metodni qo'llash algoritmlar uchun quyidagi umumiy prinsiplardan iborat:

1. Muammoni bo'lish (Divide) : Katta muammoni yechish uchun uni bir necha kichik mustaqil muammo yoki qism masalalarga bo'lish.
2. Yechish (Conquer) : Har bir kichik masalani alohida yechish.
3. Birlashtirish (Combine) : Kichik masalalarning yechimlarini birlashtirib, umumiy muammo yechimini hosil qilish.

Bu metodni qo'llagan algoritmlar qatoriga quyidagilar kiradi:

1. Tezkor Saralash (Quicksort)
  - Massiv elementlarini bir-biridan ajratib, ularni saralash.
  - Murakkabligi:  $O(n \log n)$  o'rtacha va eng yaxshi holat,  $O(n^2)$  eng yomon holat.
2. Birlashtirish Saralash (Merge Sort)
  - Massivni qismlarga bo'lib, ularni alohida saralash va so'ng birlashtirib saralash.
  - Murakkabligi:  $O(n \log n)$
3. Ikkilamchi Qidirish (Binary Search)
  - Tartiblanagan massivda elementni tezkor qidirish.
  - Murakkabligi:  $O(\log n)$

#### 4. Maximal Oraliq Yig'indisi (Maximum Subarray Sum)

- Massivdagi eng katta yig'indi oraliqni topish.
- Murakkabligi:  $O(n \log n)$

#### 5. Matris Ko'paytirish (Matrix Multiplication)

- Matrisalarni ko'paytirish.
- Murakkabligi:  $O(n^2.3)$

"Bo'lib tashla va hukumronlik qil" metodi qo'llangan algoritmlar ko'pincha juda samarali bo'lib, murakkab masalalarni yechishda keng qo'llaniladi.

"Bo'lib tashla va hukumronlik qil" metodi, bitta ma'lumot toifasiga kiruvchi algoritmni ifodalaydi. Bu algoritm odatda qidiruv amaliyotlari, ma'lumotlarni saqlash va tahlil qilish jarayonlarida foydalaniladi. Quyidagi algoritm turlari bitta ma'lumot toifasiga kiradi:

1. Linearni qidiruv (Linear Search) : Bu algoritm, berilgan ro'yxat yoki massivda berilgan ma'lumotni qidiradi. Elementni birinchilardan boshlab oxirigacha tekshiradi va berilgan qiymatni topganda uni qaytaradi. Shu sababli, bu algoritm odatda "bo'lib tashla va hukumronlik qil" metodiga kiradi.

2. Hash qidiruv (Hashing) : Hash funksiyalari ma'lumotlarni bitta ma'lumot toifasiga kiruvchi, ya'ni "hash" qiymatiga aylantiruvchi funksiyalardir. Bu funksiyalar obyektlarni ko'plik bilan asoslab, ularni unikal hash qiymatlarga o'zgartiradi. Keyin, bu hash qiymatlarni ma'lumotlar strukturasi ichida indeks sifatida ishlatish orqali ma'lumotlarni qidirish uchun ishlatiladi. Hashing algoritmi "bo'lib tashla va hukumronlik qil" metodiga kiradi, chunki ma'lumotlar unikal hash qiymatlarga o'xshab, bitta ma'lumot toifasiga kiradi.

3. Jadval qidiruv (Table Lookup) : Jadval qidiruv algoritmi, berilgan ma'lumotlar jadvalidan ma'lumotni qidiradi. Jadvalda kerakli ma'lumotlar "kalit" sifatida ishlatiladi

va ularning qiymatlari jadvalning boshqalariga bog'liq bo'lib qoladi. Bu algoritmi "bo'lib tashla va hukmronlik qil" metodiga kiradi, chunki kerakli ma'lumotlar jadvaldan olingan bo'lib, uni aniqlash va o'qish jarayonida amalga oshiriladi.

Ushbu algoritmlar bitta ma'lumot toifasiga kirib, uni qidirish va amalga oshirishda ishlatiladi. Bu algoritmlar odatda ma'lumotlar strukturasi yoki ma'lumotlarni saqlash uchun ishlatiladi.

### **S173. Chiziqli qidirish algoritmi va uning ishlash algoritmi tushuntiring.**

#### **Chiziqli Qidirish (Linear Search) - Tushuncha va Ishlash Algoritmi**

Chiziqli Qidirish (Linear Search) - bu eng oddiy qidirish algoritmi. Uning asosiy maqsadi berilgan ma'lumotlar to'plamida qidirayotgan elementni topish va uning indeksini aniqlashdir.

#### **Chiziqli Qidirish Algoritmi:**

1. Ma'lumotlar to'plamidagi birinchi elementdan boshlab barcha elementlarni tekshirish.
2. Agar tekshirayotgan element qidirayotgan elementga teng bo'lsa, topilgan vaqtni qaytarish.
3. Agar ma'lumotlar to'plamini oxiriga yetib kelinsa element topilmagan deb natija qaytarish.

#### **Algoritm Pseudo-kodi:**

...

```
function linearSearch(arr, target):
```

```
 for i = 0 to length(arr) - 1:
```

```
 if arr[i] == target:
```

```
 return i
```

```
 return -1
```

...

#### **Ishlash Tartibi:**



1. Berilgan ma'lumotlar to'plamida birinchi elementdan boshlab barcha elementlar tekshiriladi.
2. Har bir elementning qiymati qidirayotgan element bilan solishtiriladi.
3. Agar birortasi mos keladigan element topilsa, indeks qaytariladi.
4. Agar ma'lumotlar to'plamini oxiriga yetib kelinsa, element topilmaganligi (-1) natija qaytariladi.

#### Murakkablik Tahlili:

- Eng yaxshi holat:  $O(1)$  - qidirayotgan element to'plamning boshida joylashgan bo'lsa.
- O'rtacha holat:  $O(n)$  - elementlar tasodifiy tartibda yoki oxirida joylashgan bo'lsa.
- Eng yomon holat:  $O(n)$  - elementlar to'plamning boshida yo'q bo'lsa yoki butun to'plamni tekshirish kerak bo'lsa.

Chiziqli qidirish algoritmi soddadan foydalanish uchun mos keladi, lekin katta hajmdagi ma'lumotlar to'plamlarida murakkabliklari yuqori bo'ladi. Bunday holatlarda, tartiblangan ma'lumotlar to'plamida chiziqli qidirishdan foydalanish yoki boshqa samarali qidiruv algoritmlarini qo'llash maslahat beriladi.

Chiziqli qidiruv (Linear Search) algoritmi, berilgan ro'yxatda yoki massivda berilgan ma'lumotni qidirish uchun ishlatiladi. Bu algoritmda barcha elementlar bittadan boshlab oxirigacha tekshiriladi va berilgan qiymat topilganda uni qaytaradi. Chiziqli qidiruvning ishlash algoritmi quyidagi tartibda bajariladi:

1. Boshlang'ich ro'yxat indeksi  $i=0$  ga tengiladi.
2. Oxirgi ro'yxat indeksi  $(n-1)$  gacha tekshirish jarayonini boshlash uchun doira ochiladi.
3. Har bir qadamda  $i$ -indeksdagi ro'yxat elementi, qidirilayotgan qiymatga teng bo'lganligini tekshiriladi.
4. Agar ro'yxat elementi qidirilayotgan qiymatga teng bo'lsa,  $i$  qiymati qaytariladi va qidiruv to'xtatiladi.

5. Aks holda, i-indeksi birga oshiriladi va qidiruv davom ettiriladi.
6. Agar ro'yxat oxiriga yetilmasa va topilmagan bo'lsa, ma'lumot mavjud emas deb bildiriladi va -1 qiymati qaytariladi.

Quyidagi Kotlin kodida Chiziqli qidiruv algoritmi ko'rsatilgan:

```
```kotlin
```

```
fun linearSearch(arr: IntArray, key: Int): Int {  
    val n = arr.size  
    for (i in 0 until n) {  
        if (arr[i] == key) {  
            return i // Qiymat topilgan, indeksni qaytarish  
        }  
    }  
    return -1 // Qiymat topilmagan  
}
```

```
fun main() {  
    val arr = intArrayOf(64, 34, 25, 12, 22, 11, 90)  
    val key = 22  
    val result = linearSearch(arr, key)  
    if (result != -1) {  
        println("$key qiymati $result indeksda topildi.")  
    } else {  
        println("$key qiymati topilmadi.")  
    }  
}
```

...

Ushbu Kotlin kodida Chiziqli qidiruv algoritmi ko'rsatilgan. Kodning natijasida, qidirilayotgan qiymat topilgan bo'lsa, uni indeksini qaytaradi; aks holda, topilmagan deb -1 qiymatini qaytaradi.

S174. Graflarda Depth first search (DFS) algoritmi.

Depth First Search (DFS) algoritmi graflarda bo'lib, grafdagi barcha tugunlarni va ularga bog'langan boshqa tugunlarni o'zgarib, uchunadigan bir algoritm hisoblanadi. Bu algoritm barcha ko'rsatkichlar qatoridan o'tib, ko'rsatkichni o'zgartirib ketadi va undan keyin ko'rsatkichning o'zida qaytib ko'rinadi. Bu jarayon rekursiv tarzda amalga oshiriladi.

DFS algoritmi quyidagi tartibda bajariladi:

1. Boshlang'ich tugunni tanlash va uni "turing" holatiga olib kirish.
2. Tanlangan tugunni (va undan boshqa tugunlarni) "turing" holatiga olib kirish.
3. Tanlangan tugunni ko'rsatkichlar qatorida qo'shish va undan so'ng qidirishni davom ettirish.
4. Agar ko'rsatkichlar qatori bo'sh bo'lsa, tugunni tanlangan sifatida belgilash va keyin "turing" holatdan chiqish.
5. Agar ko'rsatkichlar qatori bo'sh bo'lmasa, ko'rsatkichlar qatorining birinchi elementini olib, uni "turing" holatiga olib kirish va qidirishni davom ettirish.
6. Qidiruv barcha tugunlarni va ularga bog'langan boshqa tugunlarni "turing" holatiga olib kirib, qidiruvni to'xtatish orqali yakunlanadi.

Quyidagi Kotlin kodida Depth First Search (DFS) algoritmi ko'rsatilgan:

```
```kotlin
```

```
class Graph(private val V: Int) {
 private val adj: Array<MutableList<Int>> = Array(V) { mutableListOf() }

 fun addEdge(v: Int, w: Int) {
 adj[v].add(w)
 }

 private fun DFSUtil(v: Int, visited: BooleanArray) {
```

```
visited[v] = true
```

```
print("$v ")
```

```
val i: Iterator<Int> = adj[v].listIterator()
```

```
while (i.hasNext()) {
```

```
 val n = i.next()
```

```
 if (!visited[n])
```

```
 DFSUtil(n, visited)
```

```
}
```

```
}
```

```
fun DFS(startVertex: Int) {
```

```
 val visited = BooleanArray(V)
```

```
 DFSUtil(startVertex, visited)
```

```
}
```

```
}
```

```
fun main() {
```

```
 val g = Graph(4)
```

```
 g.addEdge(0, 1)
```

```
 g.addEdge(0, 2)
```

```
 g.addEdge(1, 2)
```

```
 g.addEdge(2, 0)
```

```
 g.addEdge(2, 3)
```

```
 g.addEdge(3, 3)
```

```
 println("DFS traversal starting from vertex 2:")
```

```
g.DFS(2)
}
...
```

Ushbu Kotlin kodida, grafdagi Depth First Search (DFS) algoritmi ko'rsatilgan. Algoritm boshlang'ich tugunni olib, DFSUtil funksiyasiga uzatiladi va undan so'ng qidirishni boshlaydi. Natijada, grafdagi barcha tugunlar Depth First tartibda chiqariladi.

### Depth-First Search (DFS) Algoritmi

Depth-First Search (DFS) - bu graflarni qidirish algoritmlaridan biri. Uning asosiy g'oyasi grafdagi tugunlarni chuqurlik bo'yicha izlab chiqish.

#### DFS Algoritmi Ishlash Tartibi:

1. Grafning ixtiyoriy boshlanish nuqtasidan (tugunidan) boshlanadi.
2. Boshlang'ich tugunda joylashgan, yana borilmagan qo'shni tugunlardan birini tanlab, unga o'tadi.
3. Tanlab olingan yangi tugundagi qo'shni tugunlardan yana bittasini tanlab, unga o'tadi.
4. Agar hozirgi tugunda borilmagan qo'shni tugun qolmasa, oldingi tugun (ota-tuguni) ga qaytadi.
5. Barcha tugunlar borib chiqilgunga qadar (yoki qidirilayotgan element topilgunga qadar) algoritmni davom ettiradi.

#### Algoritm Pseudo-kodi:

```
...
function DFS(Graph, startVertex):
```

mark startVertex as visited

for each unvisited neighbor v of startVertex:

DFS(Graph, v)

...

Murakkablik Tahlili:

- Maydoni (Space Complexity):  $O(|V|)$  - grafning barcha tugunlari uchun qidiruv qiladi.
- Vaqti (Time Complexity):  $O(|V| + |E|)$  - grafning barcha tugunlari va aloqa (edge)larini tekshiradi.

DFS algoritmi qo'llaniladigan sohalar:

- Murakkab graflar va tarmoqlar tuzilmasini tekshirish
- Bog'liqlikning mavjudligini aniqlash
- Sikl mavjudligini aniqlash
- Topologik saralash
- Qidiruv va izlash muammolarini yechish

DFS algoritmi qo'llaniladigan asosiy sohalarda turli muammolarni yechish uchun ishlatiladi. Shu bilan birga, u graflarni qidirish, tahlil qilish va boshqa masalalarni hal etish uchun keng qo'llaniladi.

## S175.Shell saralash (Shell sort) haqida tushuncha va uni ishlash algoritmi.

### Shell Sort - Tushuncha va Ishlash Algoritmi

Shell Sort - bu klassik algoritmlarga kiruvchi, o'rtacha natijalar ko'rsatadigan saralash algoritmidir. U ko'p qiymatli va katta hajmdagi ro'yxatlarni saralashda samarali ishlaydi.

### Shell Sort Ishlash Prinsipi:

1. Ro'yxatni bir necha qismga (oraliq) bo'ladi.
2. Har bir oraliqni alohida saralaydi.
3. Oraliq oraliq'i kamaytirib, qayta saralash jarayoni davom etadi.
4. Oraliq 1 ga teng bo'lgandan so'ng, ro'yxat sanalgan hisoblanadi.

### Shell Sort Algoritmi Pseudo-kodi:

...

```
function shellSort(arr):
```

```
 n = length(arr)
```

```
 interval = n/2
```

```
 while interval > 0:
```

```
 for i = interval to n-1:
```

```
 temp = arr[i]
```

```
 j = i
```

```
 while j >= interval and arr[j-interval] > temp:
```

```
 arr[j] = arr[j-interval]
```

```
 j -= interval
```



```
arr[j] = temp
interval = interval/2

return arr
...
```

#### Shell Sort Ishlash Tartibi:

1. Ro'yxatning uzunligiga ( $n$ ) qarab boshlang'ich intervalni (oraliq) tanlash:  $n/2$
2. Boshlang'ich intervalni qo'llagan holda, elementlarni saralash
3. Interval qiymatini 2 ga bo'lib, qayta saralash jarayonini davom ettirish
4. Interval qiymati 1 ga teng bo'lguncha (to'liq sanalgan) bu jarayon davom etadi

#### Murakkablik Tahlili:

- Eng yaxshi holat:  $O(n \log n)$
- O'rtacha holat:  $O(n \log n)$
- Eng yomon holat:  $O(n^2)$

Shell Sort algoritmi klassik saralash algoritmlariga nisbatan yaxshiroq natijalar ko'rsatadi. Uning asosiy afzalligi - uzun ro'yxatlarni saralashda yuqori samaradorlikka erishishidir. Shu bilan birga, uning murakkabligi ham o'rtacha hisoblanadi.

Shell Sort ko'p hollarda tezkor va samarali saralash algoritmi sifatida qo'llaniladi. Uning ishlash jarayoni oddiy bo'lsa-da, yaxshi natijalar ko'rsatishiga imkon beradi.

Shell saralash (Shell sort) algoritmi, qo'shimcha gaplarni o'zgartirib chiqarib, pastroq saralashning qattiq versiyasini taqdim etadi. Bu algoritim 1959-yilda Donald Shell tomonidan taklif qilingan va boshqaruvdagi pastroq saralash algoritmlariga nisbatan

bir necha qadamli bo'lib, qidiruvni yuzaga chiqarish va saralash jarayonini tezlashtirish uchun ishlatiladi.

Shell saralashning ishlash algoritmi quyidagicha:

1. Saralash uchun qadam uzatiladi (odd sequence).
2. Ro'yxat boshida barcha odd-uzunlikdagi interval orqali elementlarni almashish (gap saralash).
3. Har bir odd uzunlikdagi intervalda saralash amalga oshiriladi, odd uzunlik kamayguncha yana gap saralash bajariladi.
4. Ushbu jarayon boshqacha odd uzunlikdagi interval bilan takrorlanadi.
5. Odd uzunlik 1 ga yetganda, barcha elementlar chiziqli saralash jarayonida saralash uchun qaytib, bu jarayon boshqariladi.

Quyidagi Kotlin kodida Shell saralash algoritmi ko'rsatilgan:

```
```kotlin
fun shellSort(arr: IntArray) {
    val n = arr.size
    var gap = n / 2
    while (gap > 0) {
        for (i in gap until n) {
            val key = arr[i]
            var j = i
            while (j >= gap && arr[j - gap] > key) {
                arr[j] = arr[j - gap]
                j -= gap
            }
        }
    }
}
```

```
        arr[j] = key
    }
    gap /= 2
}
}
```

```
fun main() {
    val arr = intArrayOf(64, 34, 25, 12, 22, 11, 90)
    shellSort(arr)
    println("Sorted array:")
    for (i in arr.indices) {
        print("${arr[i]} ")
    }
}
...

```

Ushbu Kotlin kodida Shell saralash algoritmi ko'rsatilgan. Algoritm odd-uzunlikdagi gap larini qabul qiladi va ro'yxatni ushbu interval bo'yicha saralash jarayonini amalga oshiradi. Natijada, ro'yxat chiziqli ravishda saralandi.

S176. Daraxtsimon ma'lumotlar tuzilmasi va ularning qo'llanilishi.

Daraxtsimon Ma'lumotlar Tuzilmasi va Qo'llanilishi

Daraxtsimon ma'lumotlar tuzilmasi - bu ma'lumotlarni *иерархик*, yog'onsimon tarzda saqlash va qidirish uchun mo'ljallangan tuzilmalardir. Ular barcha element va aloqalarni o'zida ifodalaydi.

Asosiy Daraxtsimon Tuzilmalar:

1. Ikki sonli daraxt (Binary Tree) :

- Har bir tugun ikkita farzand (o'ng va chap) tuguni bo'ladi.
- Keng qo'llaniladi: qidirish, saqlash, o'chirish, saralash.

2. Avl daraxt (AVL Tree) :

- Ikki sonli daraxtning balandligi doim muvozanatlangan bo'ladi.
- Qidiruv, saqlash, o'chirish operatsiyalari samarali.

3. B-daraxt (B-Tree) :

- Ko'p farzandli daraxt.
- Katta hajmdagi ma'lumotlar bilan ishlashda samarali.
- Ko'p qo'llaniladi: fayllar tizimida, ma'lumotlar bazasida.

4. Heap daraxt (Heap Tree) :

- Ikki sonli daraxtning maxsus turi.
- Har bir tugunning qiymati farzandlaridan katta/kichik bo'ladi.
- Saralash, eng katta/kichik qiymatni topishda qo'llaniladi.

5. Trie (Prefix Tree) :

- Matnlar ustida operatsiyalar bajarish uchun mo'ljallangan.
- So'zlar, leksikografik tartibda joylashtiriladi.
- Prefiks qidirish, so'z kiritish/o'chirish uchun ishlatiladi.

Daraxtsimon Tuzilmalarning Qo'llanilishi:

- Ro'yxatlarni saqlash va qidirish: Ikki sonli daraxt, AVL daraxt.
- Fayllar tizimi va ma'lumotlar bazasi: B-daraxt.
- Eng katta/kichik qiymatni topish: Heap daraxt.
- Matnlar ustida operatsiyalar: Trie (Prefix Tree).
- Graflarni ifodalash: Ikki sonli daraxt, B-daraxt.

Daraxtsimon tuzilmalar ma'lumotlarni saqlash, qidirish va boshqa amallarni bajarish uchun keng qo'llaniladi. Har bir tuzilma o'ziga xos xususiyatlarga ega bo'lib, turli muammolarni yechishda qo'l keladi. Ularning tanlovi ko'pincha hal qilinayotgan muammoning xususiyatlariga bog'liq bo'ladi.

Daraxtsimon ma'lumotlar tuzilmasi (DSMS) ma'lumotlarni daraxt shaklida saqlash va ularni boshqarish uchun maxsus algoritmlar va ma'lumot strukturalarini taqdim etadi. DSMS, ma'lumotlarni yaxshi tashkil etish, tezlik va samarali qidirish, kiritish va o'chirish, statistik tahlil, va boshqalar kabi jarayonlarda ishlovchi bo'lish uchun qulayliklar yaratadi.

DSMS keng qo'llaniladi, chunki u o'zgaruvchan ma'lumotlarni daraxt shaklida saqlash va ularga tez, samarali vaqtda kirish imkonini ta'minlaydi. Ushbu tuzilma alohida topilganlar, ma'lumotlar qidirish, filtratsiya, aggregatsiya, va statistik tahlil kabi amallar uchun samarali algoritmlarni o'z ichiga oladi.

Daraxtsimon ma'lumotlar tuzilmasining qo'llanishiga misollar quyidagilar bo'lishi mumkin:

1. Fayllar tizimi : Fayllar tizimi har bir faylni daraxt shaklida saqlash uchun ishlatiladi. Har bir fayl nomi daraxtdagi boshqa ma'lumotlarga o'xshash "bola"lar sifatida bo'lib, ularning ustiga yozuvlar (malumotlar) saqlanadi.

2. Mijozlar bazasi : Biznes sohasida, mijozlar bazasi mijozlar haqida ma'lumotlarni boshqarish uchun o'zgaruvchan qo'llaniladi. Bu tuzilma mijozlarni daraxt shaklida saqlaydi, masalan, har bir mijozni boshqa mijozlar va ularning buyurtmalari bilan bog'langan "bola"lar bilan bog'langan boshqa mijozlarga o'xshash bo'lib saqlaydi.

3. Telefon kontaktlari : Telefon kontaktlarini boshqarishning optimal usuli daraxt shaklida saqlash bo'lishi mumkin. Har bir kontakt nomi daraxtdagi boshqa kontaktlarga bog'langan "bola"lar sifatida bo'lib, ularning tavsifi (raqami, elektron pochta manzili, ish joyi, kabi) esa ularning ustiga yozuvlar (malumotlar) sifatida saqlanadi.

DSMS ga qo'llanish, ma'lumotlarni tuzish va ularga kirishni osonlashtiradi, va tashqi tizimlar bilan integratsiya qilish uchun yaxshi platforma taqdim etadi.

S177. Binar qidirish algoritmi va uning ishlash algoritmi.

Binar Qidirish Algoritmi va Ishlash Tartibi

Binar qidirish (Binary Search) - bu tartiblangan ro'yxatda qiymatni tezkor topish uchun ishlatiladi. U ikkilamchi bo'linish (divide-and-conquer) prinsipiga asoslanadi.

Binar Qidirish Algoritmi Ishlash Tartibi:

1. Ro'yxatning o'rta qiymatini toping.
2. Agar qidirilayotgan qiymat o'rta qiymatga teng bo'lsa, qiymat topilgan.
3. Agar qidirilayotgan qiymat o'rta qiymatdan kichik bo'lsa, chap yarimro'yxatda qidirish davom etadi.
4. Agar qidirilayotgan qiymat o'rta qiymatdan katta bo'lsa, o'ng yarimro'yxatda qidirish davom etadi.
5. Agar qidirilayotgan qiymat ro'yxatda yo'q bo'lsa, qiymat topilmadi.

Binar Qidirish Algoritmi Pseudo-kodi:

```
```python
```

```
def binary_search(arr, target):
```

```
 left = 0
```

```
 right = len(arr) - 1
```

```
 while left <= right:
```

```
 mid = (left + right) // 2
```

```
 if arr[mid] == target:
```

```
 return mid
```

```
 elif arr[mid] < target:
```

```
 left = mid + 1
```

```
 else:
```

```
 right = mid - 1
```

```
return -1 # target not found
```

```
...
```

### Binar Qidirish Algoritmi Murakkablik Tahlili:

- Eng yaxshi holat:  $O(1)$  - qidirilayotgan element o'rta qiymatga teng bo'lsa
- O'rtacha holat:  $O(\log n)$  -  $n$  ta elementli tartiblangan ro'yxatda
- Eng yomon holat:  $O(\log n)$  - qidirilayotgan element ro'yxatda bo'lmasa

Binar qidirish algoritmi samarali ishlashi uchun ro'yxat tartiblangan bo'lishi shart. Algoritm barcha elementlarni bir-bir ko'rib chiqmaydi, balki ro'yxatni ikki qismga bo'lib, tegishli yarimro'yxatni tanlaydi. Bu uning samaradorligini oshiradi.

Binar qidirish algoritmi keng qo'llaniladi, chunki u o'zining tezkorligi va oson implementatsiyasi bilan ajralib turadi. U ma'lumotlar bazalari, lug'atlar, indekslar va boshqa ko'plab sohalarda qo'llaniladi.

Binar qidiruv (Binary Search) algoritmi, to'rtburchak ma'lumot tuzilmalarida barcha ma'lumotlarni daraxt shaklida saqlash uchun ishlatiladi. Bu algoritmda barcha tugunlar daraxtda joylashgan va ma'lumot qidirilmoqchi bo'lgan ma'lumotlarning birinchi tugunidan boshlab qidiriladi. Qidiruv jarayoni yarimma'lumot (binary) qidiruv usuli bo'yicha bajariladi.

Binar qidiruv algoritmining ishlash jarayoni quyidagicha:

1. Boshlang'ich massiv yoki ro'yxatda bosh, o'rtadagi va oxirgi indekslar aniqlanadi.
2. Qidirilmoqchi bo'lgan qiymat (misol uchun, "x") vaqtinchalik bosh, o'rtadagi va oxirgi indekslar orasida yerlashgan indeks aniqlanadi.
3. Qidiruv jarayoni davom ettiriladi: "x" qiymat bosh indeksdan katta bo'lsa, o'rtadagi va oxirgi indekslar orasidagi qismini qidirish davom ettiriladi. Aks holda,



qidirilmoqchi bo'lgan qiymat bosh indeksdan kichik bo'lsa, bosh va o'rtadagi indekslar orasidagi qismini qidirish davom ettiriladi.

4. "x" qiymat topilganda yoki indekslar qisqarib qolganida, qidiruv jarayoni yakunlanadi va topilgan indeks qaytariladi. Agar "x" qiymat topilmagan bo'lsa, -1 qiymati qaytariladi.

Quyidagi Kotlin kodida Binar qidiruv algoritmi ko'rsatilgan:

```
```kotlin
```

```
fun binarySearch(arr: IntArray, x: Int): Int {  
    var left = 0  
    var right = arr.size - 1  
    while (left <= right) {  
        val mid = left + (right - left) / 2  
        // x qiymati masala indeksidagi qiymatga teng bo'lsa, indeksi qaytarish  
        if (arr[mid] == x) return mid  
        // Agar x kichik bo'lsa, masala indeksini o'ng tomondan o'zgartirish  
        if (arr[mid] < x) left = mid + 1  
        // Aks holda, masala indeksini chap tomondan o'zgartirish  
        else right = mid - 1  
    }  
    // x qiymati topilmagan  
    return -1  
}
```

```
fun main() {  
    val arr = intArrayOf(2, 3, 4, 10, 40)  
    val x = 10
```

```
val result = binarySearch(arr, x)
if (result != -1) {
    println("$x qiymati $result indeksda topildi.")
} else {
    println("$x qiymati topilmadi.")
}
}
```

Ushbu Kotlin kodida Binar qidiruv algoritmi ko'rsatilgan. Kod natijasida, qidirilmoqchi bo'lgan qiymat topilganda, uni indeksini qaytaradi; aks holda, topilmagan deb -1 qiymatini qaytaradi.

S178. Bog'langan ro'yhatlar va ularning qo'llanilishi

Bog'langan Ro'yxatlar va Ularning Qo'llanilishi

Bog'langan ro'yxat (Linked List) - bu yordamchi tugunlar (node) yordamida ketma-ket joylashtirilgan ma'lumotlar tuzilmasidir. Har bir tugun o'zidan keyingi tugungacha bo'lgan yagona aloqaga ega bo'ladi.

Bog'langan Ro'yxatlarning Asosiy Turlari:

1. Birjinli Bog'langan Ro'yxat (Singly Linked List) :
 - Har bir tugun keyingi tugungacha bo'lgan aloqaga ega.
 - Odatda ro'yxatning boshi va oxirini saqlaydi.
2. Ikki tomonlama Bog'langan Ro'yxat (Doubly Linked List) :
 - Har bir tugun oldingi va keyingi tugungacha bo'lgan aloqaga ega.
 - Ro'yxatda ilgari va keyinga qadam tashlash imkonini beradi.
3. Halqa Bog'langan Ro'yxat (Circular Linked List) :
 - Oxirgi tugun boshlanish tuguniga bog'langan.
 - Ro'yxatning boshi va oxirini ajratish mumkin emas.

Bog'langan Ro'yxatlarning Qo'llanilishi:

1. Dinamik Xotira Boshqaruvi :
 - Bog'langan ro'yxatlar xotiradan samarali foydalanishga imkon beradi.
 - Elementlar odam xohishiga qarab qo'shilishi/o'chirilishi mumkin.

2. Implementatsiya va O'zgarishga Moyillik :

- Bog'langan ro'yxatlar murakkab tuzilmalarni ifodalash uchun qo'l keladi.
- Yangi elementlarni qo'shish/o'chirish oson.

3. Yordamchi Tuzilmalar :

- Dasturchining yordamchisi sifatida ishlatiladi (steks, navbat, deque).
- Qidiruv va saralash algoritmlarida qo'llaniladi.

4. Graflarni Ifodalash :

- Graflar tugunlar va ular o'rtasidagi aloqalar yordamida ifodalanishi mumkin.
- Bog'langan ro'yxatlar graflarni ifodalash uchun qo'l keladi.

Bog'langan ro'yxatlar ma'lumotlarni saqlash, qidirish, kirish va boshqa amallarni bajarish uchun qo'llaniladi. Ular samarali xotira boshqaruvi, murakkab tuzilmalarni ifodalash, graflarni tasvirlash kabi masalalarni hal qilish uchun ishlatiladi.

Bog'langan (linked) ro'yxatlar, har bir elementning o'zida keyingi elementni ko'rsatadigan bog'langan tugunlari (o'zgaruvchilari) orqali o'rganilgan ma'lumot tuzilmasidir. Bu tuzilma odatda dinamik ma'lumotlar strukturasi sifatida ishlatiladi, chunki uning o'lchami dinamik ravishda o'zgarishi mumkin.

Bog'langan ro'yxatlar quyidagi maqsadlar uchun yaxshi qo'llaniladi:

1. Kiritish va o'chirish : Elementlarni qo'shish va o'chirish tez va oddiy amallar bo'lib, bog'langan ro'yxatlar uchun juda qulay bo'lgan qo'llanish shakli.

2. Qidiruv : Bog'langan ro'yxatlar ustida qidiruv amaliyotlari (masalan, boshidan boshlab qidirish, elementlar indeksi bo'yicha qidirish) ham tez va samarali bo'lishi mumkin.

3. Birlashgan ro'yxatlar : Bog'langan ro'yxatlar asosan boshqaruvchisiz ro'yxatlardir, shuning uchun ularni birlashtirish, qisqa vaqt ichida barcha elementlarni birlashtirish mumkin.

4. Izoh : Bog'langan ro'yxatlar ustida elementlarga izoh qo'shish (masalan, ma'lumotlar yordamida) ham oson.

5. Yig'indilash : Bog'langan ro'yxatlarning yig'indilishi (masalan, barcha elementlarni birlashtirish) ham osonroq amalga oshiriladi.

Kod ko'rishni osonlashtirish uchun, bog'langan ro'yxatlarning Kotlin tilda oddiy qismi quyidagicha:

```
```kotlin
```

```
class Node(var data: Int) {
 var next: Node? = null
}
```

```
class LinkedList {
 var head: Node? = null
```

```
 fun insert(data: Int) {
 val newNode = Node(data)
 if (head == null) {
 head = newNode
 } else {
 var temp = head
 while (temp?.next != null) {
```

```

 temp = temp.next
 }
 temp?.next = newNode
}
}

```

```

fun display() {
 var current = head
 while (current != null) {
 print("${current.data} -> ")
 current = current.next
 }
 println("null")
}
}

```

```

fun main() {
 val linkedList = LinkedList()
 linkedList.insert(1)
 linkedList.insert(2)
 linkedList.insert(3)
 linkedList.insert(4)
 linkedList.display() // Output: 1 -> 2 -> 3 -> 4 -> null
}
...

```

Ushbu Kotlin kodida, bog'langan ro'yxatlar ko'rsatilgan. ``Node`` sinfi bog'langan tugunni ifodalaydi, va ``LinkedList`` sinfi bog'langan ro'yxatni boshqaradi. ``insert`` funksiya yangi element qo'shadi va ``display`` funksiya ro'yxatni ekranga chiqaradi.

### **S179. Radix saralash (Radix sort) haqida tushuncha va uni ishlash algoritmi.**

Radix saralash (Radix sort) algoritmi, qo'shimcha qisqartirish (radix) qo'llanib, har bir raqamni bitta vaqtda saralaydigan bir qidiruv algoritmidir. Bu algoritm har bir raqamni birinchi boshlang'ich raqamdan oxirgi boshlang'ich raqamgacha saralaydi. Agar elementlar sonlar bo'lsa, bu sonlar bo'yicha saralash qiladi.

Radix saralashning ishlash jarayoni quyidagicha:

1. Har bir elementning eng katta raqamini topish uchun ma'lumotlarning sonlarini o'zlashtirish uchun bir raqamdagi qo'shimcha tartibni belgilovchi algoritm (misol uchun 0-9).
2. Eng kam raqamdan boshlab saralash jarayoni boshlanadi.
3. Har bir raqam bo'yicha barcha elementlar boshqalar, o'rtalik va oxirgi tartibda saralaydi.
4. Keyin, bir raqamga boshlanadigan barcha elementlar uchun bu jarayon takrorlanadi.
5. Eng katta raqamga yetganda, saralash amalga oshiriladi va natijaviy tartib qaytariladi.

Radix saralash algoritmi odatda kichik sonli ma'lumotlar saralash uchun juda samarali bo'lib, u shu sababli katta sonli ma'lumotlar kuchli bilan bir qatorda ishlay oladi. Ammo, agar elementlar sonlar emas, balki matnlar yoki boshqa ko'rinishdagi ma'lumotlar bo'lsa, bu algoritmni ishlatish mavjud emas.

Quyidagi Kotlin kodida Radix saralash algoritmi ko'rsatilgan:

```
```kotlin
fun radixSort(arr: IntArray) {
    val max = arr.maxOrNull() ?: return
    var exp = 1
```



```

val n = arr.size
val output = IntArray(n)
while (max / exp > 0) {
    val count = IntArray(10) { 0 }
    for (i in arr.indices) {
        count[arr[i] / exp % 10]++
    }
    for (i in 1 until 10) {
        count[i] += count[i - 1]
    }
    for (i in arr.size - 1 downTo 0) {
        output[count[arr[i] / exp % 10] - 1] = arr[i]
        count[arr[i] / exp % 10]--
    }
    for (i in arr.indices) {
        arr[i] = output[i]
    }
    exp *= 10
}
}

```

```

fun main() {
    val arr = intArrayOf(170, 45, 75, 90, 802, 24, 2, 66)
    radixSort(arr)
    println("Sorted array:")
    for (element in arr) {
        print("$element ")
    }
}

```

```
}  
}  
...
```

Ushbu Kotlin kodida Radix saralash algoritmi ko'rsatilgan. Kod natijasida, barcha elementlar raqamlar bo'yicha saralandi.

Radix Saralash (Radix Sort) va Uning Ishlash Algoritmi

Radix Sort - bu raqamlar yoki harflardan iborat elementlarni saralash uchun ishlatiladi. U o'rnatilgan sonli tizim (positional number system) asosida ishlaydi. Bu algoritm o'zgarmas paytda (linear time) ishlaydi, bu esa uni juda tezkor qiladi.

Radix Sort Algoritmi Ishlash Tartibi:

1. Elementlarning eng katta raqam yoki harf soni aniqlanadi.
2. Elementlar eng kichik raqam yoki harfdan boshlab saralanadi.
3. Elementlar har bir raqam yoki harf uchun yangi ro'yxatlarga (bucket) tarqatiladi.
4. Elementlar har bir bucket ichida saralanadi.
5. Natijadagi yangi ro'yxat yana saralanishi kerak, to eng katta raqam yoki harf saralanguncha.

Radix Sort Pseudo-kodi:

```
```python  
def radix_sort(arr):
 # Elementlarning eng katta raqam yoki harf soni
 max_digit = max(arr)
```

```

Har bir raqam yoki harf uchun saralash
exp = 1
while max_digit / exp > 0:
 counting_sort(arr, exp)
 exp *= 10

def counting_sort(arr, exp):
 n = len(arr)
 # Output array elements
 output = [0] * n
 # Initialize count array as 0
 count = [0] * 10

 # Store count of occurrences in count[]
 for i in range(n):
 index = arr[i] // exp
 count[index % 10] += 1

 # Change count[i] so that count[i] now contains actual
 # position of this digit in output array
 for i in range(1, 10):
 count[i] += count[i - 1]

 i = n - 1
 while i >= 0:
 index = arr[i] // exp
 output[count[index % 10] - 1] = arr[i]

```

```
count[index % 10] -= 1
```

```
i -= 1
```

```
for i in range(n):
```

```
 arr[i] = output[i]
```

```
...
```

### Radix Sort Murakkablik Tahlili:

- Murakkablik:  $O(kn)$ ,  $k$  - elementlarning eng katta raqam yoki harf soni,  $n$  - elementlar soni
- Radix Sort ikkilamchi saralash algoritmidan (Counting Sort) foydalanadi, shuning uchun uning umumiy murakkabligi Counting Sort murakkabligiga teng.
- Counting Sort  $O(n+k)$  murakkabligi bilan ishlaydi,  $k$  - maksimal element qiymati.

Radix Sort qimmatli, tezkor va oson implementatsiya qilinuvchi algoritm hisoblanadi. U katta to'plamli ma'lumotlarni saralashda juda samarali. Asosan, raqam yoki harf tizimidan foydalangan holda tartiblash kerak bo'lganda ishlatiladi.

## S180. Stack ma'lumotlar tuzilmasi va uning qo'llanilishi.

Stack ma'lumotlar tuzilmasi, ma'lumotlarni tuzish va undan foydalanishning odatiy usuli hisoblanadi. U holda, ma'lumotlar tizimi ma'lumotlarni yig'ish uchun odatiy bo'shlig'i mavjud bo'lgan o'zgaruvchan ma'lumotlar tizimini ifodalaydi. Bunday bo'shlig'ingizdan o'zgaruvchan uzunligi yo'q, balki siz unga ma'lumotlarni qo'shishingiz, ularni o'qishingiz, o'chirishingiz va undan foydalanishingiz mumkin.

Stack tizimi, bo'shlig'iga ma'lumotlarni qo'shish (push) va ularni olish (pop) amaliyotlarini o'z ichiga oladi. Odatda, bu amaliyotlar "Last In, First Out" (LIFO) tartibida amalga oshiriladi, yani eng oxirgi qo'shilgan ma'lumot eng birinchi olinadi.

Stack ma'lumotlar tuzilmasining asosiy operatsiyalari quyidagilardir:

1. push(item) : Stackga yangi ma'lumotni qo'shish.
2. pop() : Eng so'nggi qo'shilgan ma'lumotni olish va undan uni o'chirish.
3. peek() : Eng so'nggi qo'shilgan ma'lumotni olish, lekin undan uni o'chirmaslik.
4. isEmpty() : Stack bo'shlig'ini tekshirish, agar bo'sh bo'lsa true, aks holda false qaytaradi.
5. size() : Stackda qancha element borligini qaytaradi.

Stack tizimi odatda kutubxonalar (array) yoki bog'liq ro'yxatlar (linked lists) orqali amalga oshiriladi. Kutubxona odatda sodda operatsiyalar uchun yaxshi bo'lsa, bog'liq ro'yxatlar keng ishlab chiqish imkoniyatini ta'minlayadi.

Quyidagi Kotlin kodida Stack ma'lumotlar tuzilmasi ko'rsatilgan:

```
```kotlin
```

```
class Stack {
```

```
    private var items: MutableList<Int> = mutableListOf()
```

```
fun push(item: Int) {  
    items.add(item)  
}
```

```
fun pop(): Int? {  
    if (isEmpty()) return null  
    return items.removeAt(items.size - 1)  
}
```

```
fun peek(): Int? {  
    if (isEmpty()) return null  
    return items[items.size - 1]  
}
```

```
fun isEmpty(): Boolean {  
    return items.isEmpty()  
}
```

```
fun size(): Int {  
    return items.size  
}  
}
```

```
fun main() {  
    val stack = Stack()  
    stack.push(1)
```

```

stack.push(2)
stack.push(3)

println("Stack size: ${stack.size()}") // Output: Stack size: 3
println("Top element: ${stack.peek()}") // Output: Top element: 3

println("Pop element: ${stack.pop()}") // Output: Pop element: 3
println("Stack size after pop: ${stack.size()}") // Output: Stack size after pop: 2
}
...

```

Ushbu Kotlin kodida Stack ma'lumotlar tuzilmasi ko'rsatilgan. ``push``, ``pop``, ``peek``, ``isEmpty``, va ``size`` funksiyalar orqali Stack tizimi boshqariladi.

Stack Ma'lumotlar Tuzilmasi va Uning Qo'llanilishi

Stack - bu ma'lumotlar tuzilmasi bo'lib, ma'lumotlarni ichki qismi bilan ishlashga imkon beradi. Stackda elementlar oxirgi kiritilgan elementdan boshlab olinadi (LIFO - Last-In-First-Out).

Stack Asosiy Amallar:

1. `Push(element)` : Stackning ustiga yangi element qo'shish.
2. `Pop()` : Stackning ustidagi elementni o'chirish va qaytarish.
3. `Peek()` : Stackning ustidagi elementni qaytarish, lekin uni o'chirmaydi.
4. `isEmpty()` : Stackning bo'sh-emasligini tekshirish.
5. `isFull()` : Stackning to'la-emasligini tekshirish.

Stack Qo'llanilishi:

1. Ifoda baholash (Expression Evaluation) :

- Arifmetik ifodalarni baholashda stack qo'llaniladi.
- Operandlar stackga, operatorlar esa ifodani baholash uchun ishlatiladi.

2. Backtracking Algoritmmlar :

- Backtracking algoritmmlarda stack takroriy chaqiruvlarni saqlash uchun ishlatiladi.
- Masalan, N-queen, Sudoku, Maze Solving.

3. Chaqiruvlar Stacki (Call Stack) :

- Dasturlash tillarida funktsiyalar chaqiruvlarini saqlash uchun stack qo'llaniladi.
- Funktsiyadan chiqish vaqtida stackdan oxirgi element olib tashlanadi.

4. Brauzer Tarixini Boshqarish :

- Brauzer oynasi orasida navigatsiya qilish uchun stack ishlatiladi.
- Oldinga/orqaga qaytish amallari stackda saqlanadi.

5. Undo/Redo Amallarini Amalga Oshirish :

- Amalni bekor qilish (undo) va qayta amalga oshirish (redo) uchun stack qo'llaniladi.
- Har bir amal stackga pushlanadi, so'ngra ular orasida undo/redo amallarini bajarish mumkin.

Stack ma'lumotlar tuzilmasi, operatsiyalari oson va tushunarlilik tufayli keng qo'llaniladi. U ifoda baholash, algoritm amalga oshirish, xotira boshqaruvi kabi masalalarni yechishda ishlatiladi.

S181. Algoritmnlarni ifodalashning usullari.

Algoritmnlarni Ifodalashning Usullari

Algoritmnlar - muayyan muammoni yechish uchun ishlatiladi. Algoritmnlarni aniqlash va ifodalash uchun turli usullar mavjud:

1. Psevdokod (Pseudocode):

- Algoritmni mavhum va zamonaviy tilsiz yozish usuli.
- Oson o'qilishi, tushunarli bo'lishi va bajarilishi mumkin.
- Odatda, har bir qadamning bajarilishi uchun ta'rif yoki tushuntirish qo'shiladi.

2. Blok Diagrammalar (Flowcharts):

- Algoritm bosqichlarini grafik ko'rinishda ifodalash.
- Turli shakllardagi bloklar (masalan, uchburchak, to'rtburchak) orqali algoritm qadamlari ko'rsatiladi.
- Bloklar orasidagi o'zaro bog'liqlik oqimlar orqali ifodalanadi.

3. Tabiiy Til (Natural Language):

- Algoritmni yozma ravishda, kundalik nutq orqali ifodalash.
- Batafsil tushuntirish va izohlar berilishi mumkin.
- Qisqa va aniq bo'lishi, ammo ba'zan noaniq va noimkoni bo'lishi mumkin.

4. Tavsifiy Dasturlash Tillari (High-Level Programming Languages):

- Algoritmnlarni dasturlash tillari yordamida yozish.
- Masalan, C, Python, Java, C++ kabi tillar.
- Algoritmni aniq va bir xil tarzda ifodalash imkonini beradi.
- Algoritmni bajarish uchun kompyuter tomonidan bajariladigan kod yaratadi.

5. Matematika/Mantiq Notatsiyalari:

- Algoritmnlarni mantiqiy va matematik formulalar orqali ifodalash.
- Masalan, xalqaro algoritmlar tilida (pseudocode) yozish, Bul algebra, rekursiv tenglamalar.
- Aniq va formal, ammo ba'zan murakkab bo'ladi.

Algoritmnlarning turli xil ifodalash usullari mavjud. Ulardan qaysi biri qo'llanilishi muammo va kontekstga bog'liq bo'ladi. Har bir usulning o'z afzalliklari va kamchiliklari mavjud. Oson o'qilish, tushunarlilik, anqlik va rasmiylik muhim omillar hisoblanadi.

Algoritmnlarni ifodalashning usullari bir algoritmdagi operatsiyalarni, strukturalarni va mavjud qiymatlarni oson va aniq shaklda tushuntirish uchun foydalaniladi. Algoritmnlarni ifodalashda quyidagi usullar odatda ishlatiladi:

1. Matematik ifodalash : Algoritmdagi har bir amalga oshiriladigan amallar matematik ifodalash orqali ifodalanganda, ularning o'sish tartibi, qanday holatda o'tkazilishi, va natijaviy qiymatlar kengroq o'rganiladi. Masalan, algebraik formulalar, mantiqiy ifodalar va matematik jarayonlarga asoslangan tushunchalar bilan algoritmnlarni tushuntirish.
2. Natijaviy tasvirlar : Algoritmnlar umuman maqsad va natijalariga qaratiladi, shuning uchun ularning natijaviy tasvirlarini berish foydali bo'ladi. Bu usulda algoritmdagi har bir qadam va jarayon bir ta'rif va natijaviy o'zgaruvchilar bilan ifodalangan.
3. Algoritmik tushunchalar : Algoritmdagi har bir qadamni alohida qo'llanib, uning algoritmik tushunchalar bilan ifodalash, masalan, qidiruv jarayonlari, o'zgaruvchilar va shartlar, rekursiv chaqirishlar, va boshqalar.
4. Uml-diagrammalari : UML (Unified Modeling Language) algoritmik jarayonlarni ifodalash uchun ko'p qo'llaniladi. UML-diagrammalari orqali, algoritmdagi har bir qadam va mantiqiy bog'lanishlarni ifodalash, jarayonning qanday holatda amalga oshirilishi va algoritmik strukturalarni tushuntirish mumkin.

5. Kodnamalar : Algoritmdagi har bir qadamni amalga oshirish uchun to'g'ri, aniq va qulay kodnamalar yaratish. Bu usulda, algoritmning har bir qadamini real dasturlash tillarida (masalan, Python, Java, Kotlin) yoki matematik xususiyatlar bilan ifodalash keng tarqalgan.

6. Yangi tushunchalar yoki formulalar : Algoritmalar umuman yangi tushunchalar, formulalar yoki matematik modelni ko'rsatishi mumkin. Yangi tushunchalar, algoritmning asosiy qatlamini va amal qilishini tushuntirishda yordam beradi.

7. Qimmatbaho tahlillar : Algoritmdagi amallarni, tasavvur qilingan barcha holatlarni, natijalar va xususiyatlarni tushuntirish uchun qimmatbaho tahlillar keng foydalaniladi. Bu tahlillar algoritmning samaradorligini, xato qilish imkoniyatini va qanchalik qo'zg'atilishi kerakligini aniqlashga yordam beradi.

Bu usullar algoritmni oson va aniqligi bilan tushuntirishda yordam beradi, shuning uchun algoritmni tushuntirish va qo'llashda ularni birlashuvchi usullardan foydalanish maqsadga muvofiq bo'ladi.

S182. Vaznga ega va aylanasi yo'q bo'lgan yo'naltirilgan graflar uchun eng qisqa yo'lni topish algoritmi.

Vaznga ega va aylanasi yo'q bo'lgan yo'naltirilgan graflar uchun eng qisqa yo'lni topish algoritmi Bunday graflar uchun eng mashhur algoritm - Dijkstra Algoritmidir . Dijkstra Algoritmi

Dijkstra algoritmi eng qisqa yo'lni topishdek muammoni hal qilishda foydalaniladi. U vaznga ega va aylanasi yo'q bo'lgan yo'naltirilgan graflar uchun ishlaydi.

Algoritmning asosiy bosqichlari quyidagicha:

1. Boshlanish nuqtasidan har bir tugunning eng qisqa masofasini saqlash uchun massiv yarating.
2. Hamma tugunlarni "ochildi" deb belgilang va boshlanish nuqtasining masofasini 0 qiling, boshqa barcha tugunlarning masofasini cheksiz qiling.
3. Ochildi deb belgilangan tugunlardan eng kam masofadagi tugungacha bo'lgan masofani toping.
4. Ushbu tugungacha bo'lgan yo'lni qayd eting va uni "yopildi" deb belgilang.
5. Agar yakuniy nuqta "yopildi" deb belgilangan bo'lsa, algoritm to'xtaydi. Aks holda, 3-qadam qaytariladi.

Algoritmning ishlashi:

1. Boshlang'ich nuqta A, barcha tugunlar "ochildi" va A ning masofasi 0.
2. A va B orasidagi yo'l uzunligi 2, B ning masofasi 2 bo'ladi.
3. A va C orasidagi yo'l uzunligi 1, C ning masofasi 1 bo'ladi.
4. C va D orasidagi yo'l uzunligi 1, D ning masofasi 2 bo'ladi.
5. D va E orasidagi yo'l uzunligi 2, E ning masofasi 4 bo'ladi.
6. Eng kam masofadagi E tuguni "yopildi" deb belgilanadi.
7. Eng qisqa yo'l: A -> C -> D -> E

Dijkstra algoritmi grafdagi eng qisqa yo'lni tezda va samarali topishga imkon beradi. U murakkab algoritmlar orasida eng mashhurlaridan biridir.

Vaznli (weighted) va aylanmagan (unweighted) yo'naltirilgan graflarda eng qisqa yo'lni topish algoritmi uchun quyidagi usulni amalga oshirish mumkin:

1. Qadamli qidiruv (Breadth-First Search - BFS) :

BFS algoritmi, vaznli va aylanmagan graflar uchun eng qisqa yo'l topishda juda samarali bo'ladi. U grafdagi barcha tug'unlarni o'zi shaxsiy o'zgartiruvchisiga qo'shib, undan boshlab barcha qolgan tug'unlarni aniqlab olib, yo'naltirilgan yo'l topishga imkon beradi. Vaznli yo'naltirilgan graflarda, BFS algoritmi har bir qadamda boshlang'ich tug'unlar bilan bog'liq bo'lgan yopqinlik miqdorini saqlaydi. Aylanmagan yo'naltirilgan graflarda, miqdor yo'qligi saqlanmaydi.

Bu algoritm quyidagi tartibda ishlaydi:

- Boshlang'ich tug'unlarni qabul qilish.
- Boshlang'ich tug'unlarga qo'shimcha o'zgaruvchilarni qo'shish va ularni bir o'zgaruvchiga o'zlashtirish.
- Har bir tug'un o'zgaruvchilarini tekshirish va ularga tegishli bo'lmagan tug'unlarni qo'shish.
- Har bir qadamda o'zgaruvchilarni saqlaydigan qatorning barcha tug'unlarini tekshirish.
- Qolgan tug'unlarni aniqlash va ularga yana qo'shimcha o'zgaruvchilar qo'shish.

Ushbu usuldan foydalanish bilan, grafdagi eng qisqa yo'l topish mumkin.

Misol sifatida, quyidagi Kotlin kodida BFS algoritmi ko'rsatilgan:

```
```kotlin
```

```
fun shortestPathBFS(graph: Map<Int, List<Int>>, start: Int, end: Int): List<Int>? {
 val queue = ArrayDeque<List<Int>>()
 val visited = mutableSetOf<Int>()
 queue.add(listOf(start))
 visited.add(start)
```

```

while (queue.isNotEmpty()) {
 val path = queue.removeFirst()
 val lastNode = path.last()
 if (lastNode == end) {
 return path
 }

 for (neighbor in graph[lastNode] ?: emptyList()) {
 if (neighbor !in visited) {
 visited.add(neighbor)
 val newPath = path.toMutableList()
 newPath.add(neighbor)
 queue.add(newPath)
 }
 }
}
return null
}

```

```

fun main() {
 val graph = mapOf(
 1 to listOf(2, 3),
 2 to listOf(4),
 3 to listOf(5),
 4 to listOf(6),
 5 to listOf(6),

```

```
 6 to emptyList()
)

 val shortestPath = shortestPathBFS(graph, 1, 6)
 println("Shortest path: $shortestPath") // Output: Shortest path: [1, 2, 4, 6]
}
...

```

Ushbu Kotlin kodida, berilgan yo'naltirilgan grafda eng qisqa yo'l topish uchun BFS algoritmi ko'rsatilgan. Kod natijasida, boshlang'ich tug'unidan boshlab barcha tug'unlarni qidirish va eng qisqa yo'lni topish uchun ko'rsatilgan.

### S183. Counting saralash (Counting sort) haqida tushuncha va uni ishlash algoritmi.

#### Counting Sort (Sanash Saralash) Algoritmi

Counting Sort - bu qiymatlarning sonini sanash va ularni saralash asosida ishlatiladigan algoritm. U faqat butun sonli qiymatlar bilan ishlashi mumkin.

Counting Sort Algoritmi quyidagi bosqichlardan iborat:

1. Maksimal qiymatni topish : Berilgan massivdagi eng katta qiymatni topish.
2. Frequency Array yaratish : Massivdagi har bir qiymat uchun ularning soni saqlanadigan frequency massivi yaratish. Frekvensiya massivining uzunligi maksimal qiymatdan 1 ga katta bo'ladi.
3. Frequency Array to'ldirish : Berilgan massivni boshdan oxirigacha qayta chiqib, har bir qiymatning frekvensiya massivdagi indeksini 1 ga oshirish.
4. Frekvensiya massivini yig'indi massivga aylantirish : Frekvensiya massivini yig'indi massivga aylantirish. Bunda, har bir indeks qiymatiga oldindagi indeks qiymatlarini qo'shiladi.
5. Saralangan massivni yaratish : Boshlanishidan oxirigacha berilgan massivni qayta chiqib, har bir qiymat uchun yig'indi massivdan indeks olib, saralangan massivga yozish.

Misol:

Berilgan massiv: [1, 4, 1, 2, 4, 3]

1. Maksimal qiymat: 4
2. Frekvensiya massivi: [0, 2, 1, 1, 2]
3. Yig'indi massivi: [0, 2, 3, 4, 6]
4. Saralangan massiv: [1, 1, 2, 3, 4, 4]

Counting Sort algoritmi  $O(n+k)$  murakkablikka ega, bu  $n$  - massiv uzunligi va  $k$  - maksimal qiymat. U integer qiymatlar uchun juda samarali ishlaydi. Qisqa vaqt ichida katta massivlarni saralash uchun ishlatiladigan eng samarali algoritmlardan biri hisoblanadi.

Counting saralash (Counting sort) algoritmi, har bir elementning qiymatini sanab, uning har bir qiymatiga teng bo'lgan elementlar sonini hisoblaydi. Keyin, sanalgan ma'lumotlar asosida boshqa ro'yxatni yaratadi, jadvallarni to'plab olib va asosiy



ro'yxatni saralash uchun qayta yaratadi. Bu algoritmi odatda uchta asosiy qadamdan iborat bo'ladi:

1. Sanash : Elementlarning miqdorini sanash, ya'ni har bir element qiymatiga teng bo'lgan elementlar sonini hisoblash.
2. Jadval yaratish : Elementlar qiymatlarini saralash uchun jadvallarni yaratish. Bu jadvallar odatda sanalgan element qiymatlari orasidagi maksimum va minimum qiymatlarni o'zlashtirish uchun ishlatiladi.
3. Ro'yxatni saralash : Asosiy ro'yxatni o'rganish uchun yaratilgan jadvallar orqali asosiy ro'yxatni saralash.

Counting saralash algoritmi odatda qat'iy sonlar uchun (masalan, integer yoki katta sonlar) ishlatiladi va har bir elementning qiymati ustida o'zgaruvchilarni o'chirish va saqlash uchun katta xotirani talab qilmaydi. Bunday, agar elementlar keng aralashgan bo'lsa yoki barcha elementlar bir-biriga teng bo'lsa, u yuzaga chiqqan qiymatlar orqali juda tez amalga oshiriladi.

Quyidagi Kotlin kodida Counting saralash algoritmi ko'rsatilgan:

```
```kotlin
```

```
fun countingSort(arr: IntArray) {  
    val max = arr.maxOrNull() ?: return  
    val min = arr.minOrNull() ?: return  
  
    val countArray = IntArray(max - min + 1)  
    for (element in arr) {  
        countArray[element - min]++  
    }  
  
    var index = 0  
    for (i in min..max) {  
        while (countArray[i - min] > 0) {  
            arr[index++] = i  
        }  
    }  
}
```

```

        countArray[i - min]--
    }
}
}

fun main() {
    val arr = intArrayOf(4, 2, 2, 8, 3, 3, 1)
    countingSort(arr)
    println("Sorted array:")
    for (element in arr) {
        print("$element ")
    }
}

```

Ushbu Kotlin kodida Counting saralash algoritmi ko'rsatilgan. Kod natijasida, barcha elementlar qiymatlariga ko'ra saralandi.

S184. Algoritm mukammalligi va Big O

Algoritm mukammalligi va Big O (O'lchov) yordamida algoritmning davlatida, mukammalligida va ishlash tezligida qanday o'zgarishlar kuzatiladi.

1. Algoritm mukammalligi :

Algoritmning mukammalligi uning qaysi darajada to'g'ri natijalarni topishi va har bir kirish uchun to'g'ri natijalarni chiqarishi mumkinligini aks ettiradi. Algoritmndagi har bir qadam qanday to'g'ri va to'g'ri amalga oshirilsa, algoritm mukammal deb hisoblanadi. Agar algoritm bir necha tartibda to'g'ri natijalarni topishi mumkin bo'lsa, unda ham mukammal deb hisoblanadi.

2. Big O (O'lchov) :

Big O notatsiyasi (O'lchov) algoritmning so'rovlarni ishga tushirishning har bir qadamida sarflangan vaqt va xotira miqdorini ifodalovchi usuldir. O'lchovning asosiy maqsadi algoritmning ishlash davrini o'rganish va uni boshqa algoritmlar bilan taqqoslashdir. O'lchovning quyidagi shakllari mavjud:

- $O(1)$: Soddadan yoki sababdan, misol uchun massiv elementiga murojat qilish
- $O(\log n)$: Kichikroq qismli qidiruvlar, misol uchun binarni qidirish algoritmi
- $O(n)$: Massivni to'la, yaqin orasidan yoki boshidan boshlab qidirish
- $O(n \log n)$: Quick sort yoki merge sort kabi tizimlash algoritmlari
- $O(n^2)$: Yuqori darajadagi xususiy algoritm, misol uchun Insertion sort
- $O(2^n)$: Oson algoritmlardan xolis qiyin algoritm, misol uchun Towers of Hanoi
- $O(n!)$: Factorial darajasidagi algoritm, misol uchun permutation

Big O notatsiyasi bizga algoritmning so'rovlarni qanchalik tez vaqtda ishga tushirishi haqida ma'lumot beradi, shuning uchun bizga yordam beradi qanday algoritmni qo'llashni tanlash. Yukoridagi shakllar bizga berilgan algoritmlarni taqqoslash va ularning amalga oshirilishi qanday o'zgarishlarga olib kelishi mumkinligi haqida ma'lumot beradi.

Algoritm Mukammalligi va Big O Notatsiyasi

Algoritm mukammalligi - bu algoritm ishlashining kompleksligini ifodalovchi miqdordir. Algoritm mukammalligi ko'pincha Big O notatsiyasi yordamida ifodalanadi.

Big O Notatsiyasi - bu algoritm ishlash vaqtini ifodalovchi funksiya. U algoritm ishlashining asimptotik yoki eng yomon holatini ko'rsatadi.

Big O notatsiyasi quyidagicha belgilanadi: $O(f(n))$, bu yerda $f(n)$ - algoritm ishlash vaqtini ifodalovchi funksiya.

Eng ko'p uchraydigan Big O notatsiyalari:

1. $O(1)$ - Doimiy Murakkablik : Algoritm qadamlari soni o'zgarmas bo'lib, input o'lchamlari bilan bog'liq emas. Masalan, massivning birinchi elementi bilan ishlash.
2. $O(\log n)$ - Logarifmik Murakkablik : Algoritm qadamlari $\log(n)$ ga proporsional. Masalan, binary search algoritmi.
3. $O(n)$ - Chiziqli Murakkablik : Algoritm qadamlari n ga proporsional. Masalan, massivni bir marta aylanib chiqish.
4. $O(n \log n)$ - Loglinear Murakkablik : Algoritm qadamlari $n \cdot \log(n)$ ga proporsional. Masalan, Quicksort va Mergesort kabi sorting algoritmlari.
5. $O(n^2)$ - Kvadratik Murakkablik : Algoritm qadamlari n^2 ga proporsional. Masalan, ikki xona massivni ko'paytirish.
6. $O(2^n)$ - Eksponensial Murakkablik : Algoritm qadamlari 2^n ga proporsional. Masalan, rekursiya yordamida faktorial hisoblash.

Algoritm mukammalligi tahlili algoritm samaradorligini baholash va solishtirish uchun muhim ahamiyatga ega. Bunda Big O notatsiyasi algoritm ishlash vaqtini asimptotik chegarasini ifodalagani uchun keng qo'llaniladi.

S185. Massiv tushunchasi va uning turlari.

Massiv tushunchasi, biror turdagi ma'lumotlar to'plami hisoblanadi, bu ma'lumotlar to'plami odatda bir-biriga mos kelishuvda bo'lgan elementlardan iborat. Massivlar, o'zlarining indekslariga mos kelishuvda bo'lgan ma'lumotlarni saqlash uchun foydalaniladi. Bu, biror turdagi ma'lumotlarni bir yerda bo'lgan qatorlar ko'rinishida saqlashni tashkil etadi.

Turli turdagi massivlar mavjud bo'lishi mumkin, ulardan ba'zilari quyidagilardir:

1. Bir xil ma'lumotlarni o'z ichiga oluvchi massivlar : Bu tur massivlar faqatgina biror turdagi ma'lumotlarni o'z ichiga oladi, masalan, int, float, boolean, char, va boshqalar.
2. O'zgaruvchan miqdori massivlar : Bu massivlar ma'lumotlar to'plami, kichikroq bir massivga bo'lgan kirishlar yordamida o'zgartirib borish mumkinligini ta'minlaydi.
3. Ko'p o'lchamdagi massivlar : Bu massivlar ikki yoki undan ko'p o'lchamdagi ma'lumotlar to'plamlarini saqlaydi, masalan, ikki o'lchamdagi massivlar (matrislar), uch o'lchamdagi massivlar (tensorlar) va h.k.

Massivlar dasturlashda juda mashhur va kerakli hisoblanadi, chunki ular ma'lumotlar to'plamini yashirish va unga tez murojat qilishni osonlashtiradi. Ma'lumotlar o'zgaruvchan bo'lishi mumkin va ularga mos ravishda murojat qilish mumkin. Bu esa algoritmlarni amalga oshirish va dasturlashni sodda qiladi.

Quyidagi Kotlin misolida, turli turdagi massivlarning tashrif buyurish ko'rinishlari ko'rsatilgan:

```
```kotlin
fun main() {
 // Bir xil ma'lumotlarni o'z ichiga oluvchi massivlar
```

```

val intArray = intArrayOf(1, 2, 3, 4, 5)
val doubleArray = doubleArrayOf(1.5, 2.5, 3.5)
val booleanArray = booleanArrayOf(true, false, true)

// O'zgaruvchan miqdori massivlar
var dynamicIntArray = mutableListOf<Int>()
dynamicIntArray.add(1)
dynamicIntArray.add(2)
dynamicIntArray.add(3)

// Ko'p o'lchamdagi massivlar (matris)
val matrix = arrayOf(
 intArrayOf(1, 2, 3),
 intArrayOf(4, 5, 6),
 intArrayOf(7, 8, 9)
)
}
...

```

Ushbu Kotlin kodida turli turdagi massivlarning ko'rinishlari ko'rsatilgan.

Massiv (Array) Tushunchasi va Turlari

Massiv - bu bir xil ma'lumotlar turidan iborat elementlar to'plami. Massivlar ma'lumotlarni saqlash va ulardan foydalanish uchun keng qo'llaniladi.

Massivlar quyidagi turlarga bo'linadi:

1. Bir o'lchamli massiv (1D Array) : Bu eng oddiy va ko'p qo'llaniladigan massiv turi. Bir o'lchamli massivda elementlar bir qator tartibida joylashgan bo'ladi.

Misol: `[10, 20, 30, 40, 50]`

2. Ko'p o'lchamli massiv (Multidimensional Array) : Bu bir nechta o'lchamga ega bo'lgan massivlar. Eng ko'p tarqalgan ko'p o'lchamli massiv - ikki o'lchamli massivdir.

Misol: `[[1, 2, 3], [4, 5, 6], [7, 8, 9]]`

3. Dinamik massiv (Dynamic Array) : Bu o'z o'lchamini o'zgartirib turadigan massivlar. Ma'lumotlarni saqlash uchun dinamik xotira ajratish talab qilinadi.

Misol: `ArrayList` va `Vector` kabi Java da qo'llaniladigan dinamik massivlar.

4. Assosiativ massiv (Associative Array) : Ushbu massivda elementlar indeks sifatida faqat butun sonlar emas, balki boshqa ma'lumot turlari (masalan, satrlar) ham bo'lishi mumkin.

Misol: `HashMap` va `Hashtable` kabi Java da qo'llaniladigan assosiativ massivlar.

Massivlar ma'lumotlarni saqlash va ularga tez kirish imkonini beradi. Ular obyektlar, raqamlar, satrlar va boshqa ma'lumot turlarini saqlashda qo'l keladi. Massivlarning afzalliklari - tez kirish va sodda indekslash, kamchiliklari - o'lchamini o'zgartira olmaslik va yuqori xotira talab qilish.



### S186. Rasmda keltilgan grafni Ford – Belmann algoritmi asosida eng qisqa yo'lni topish masalasi misolida yechib tushuntiring.

Javob: Algoritm tarixi uchta mustaqil matematiklar bilan bog'liq: Lester Ford, Richard Bellman va Edward Moore. Ford va Bellman algoritmini 1956 va 1958 yillarda nashr etishdi, Moore esa 1957 yilda taqdim qilgan. Va ba'zan uni Bellman-Ford-Moore algoritmi deb ham atashadi. Usul ba'zi vektorli-marshrutlash protokollarida, masalan, RIPda (Routing Information Protocol) qo'llaniladi. Deykstra algoritmi singari, Bellman-Ford algoritmi ham vaznga ega bo'lgan graflarda bitta tugundan qolgan barcha tugunlarga bo'lgan eng qisqa masofani aniqlashda ishlatiladi. Bu algoritm manfiy vaznga ega bo'lgan graflar bilan ishlashda ham qo'llanilishi mumkin (istisno holatlar ham mavjud).

s tugundan qolgan barcha tugunlargacha bo'lgan qisqa masofani Bellman-Ford algoritmidan foydalanib topish dinamik dasturlashtirish usulini qo'llash demakdir, ya'ni uni qism masalalarga ajratib, ularni yechimi orqali umumiy asosiy masalani hal qilishdir. Bunda qism masala bo'lib, bitta alohida qaralayotgan tugundan boshqasigacha eng qisqa yo'lni aniqlash masalasi hisoblanadi. Algoritm natijalarini saqlash uchun  $d[]$  bir o'lchovli massiv qabul qilamiz. Uning har bir  $i$ -elementida  $s$  gundan  $i$ -elementgacha qisqa masofa qiymatini saqlanadi (agar mavjud bo'lsa). Dastlab,  $d[]$  massiv elementlariga shartli sheksiz katta qiymat berib chiqiladi,  $d[s]$  ga 0 o'zlashtiriladi.

$G = \{V, E\}$ ,  $n = |V|$ ,  $m = |E|$  graf berilgan bo'lsin. Qo'shni tugunlarni  $v$  va  $u$  deb,  $(v, u)$  orasidagi qirrani  $w$  deb belgilab olamiz. Boshqacha aytganda,  $v$  tugundan chiquvchi va  $u$  tugunga kiruvchi qirra vazni  $w$  ga teng. U holda, Bellman-Ford algoritmining muhim qismi quyidagicha ko'rinishga ega bo'ladi:

- - $I = 1$  dan  $n - 1$  gacha bajaramiz
  - $j = 1$  dan  $m$  gacha bajaramiz
  - agar  $d[v] + w(v, u) < d[u]$  bo'lsa,  $u$  holda
  - $d[u] < d[v] + w(v, u)$

Har bir  $n$ -qadamda  $d[]$  massiv elementlari qiymatlarini yashilashga harakat qilinadi: agar  $w(v, u)$  qirra vazni va  $d[v]$  element qiymati yig'indisi  $d[u]$  qiymaridan kichik bo'lsa,  $u$  holda bu qiymat  $d[u]$  ga o'zlashtiriladi.

```

#include "stdafx.h"
#include
#define inf 100000
using namespace std;
struct Edges{
int u, v, w;
};
const int Vmax=1000;
const int Emax=Vmax*(Vmax-1)/2;
int i, j, n, e, start;
Edges edge[Emax];
int d[Vmax];
//Bellman-Ford algoritmi
void bellman_ford(int n, int s)
{
int i, j;
for (i=0; i
d[s]=0;

for (i=0; i<n-1; i++)< p="">
for (j=0; j
if (d[edge[j].v]+edge[j].w
d[edge[j].u]=d[edge[j].v]+edge[j].w;
for (i=0; i
cout<<endl<<start<<"->"<<i+1<<"="<<"not";
else cout<<endl<<start<<"->"<<i+1<<"="<
}
//asosiy funksiya
void main()
{
int w;
cout<<"tugunlar soni > "; cin>>n;
e=0;
for (i=0; i
for (j=0; j
{
cout<<"Bec "<<i+1<<"->"<<j+1<<"> "; cin>>w;
if (w!=0)
{</j+1<<"></i+1<<"-></i+1<<"="<</endl<<start<<"-
></i+1<<"="<<"not";></endl<<start<<"-></n-1; i++)<>

```

```

{
edge[e].v=i;

edge[e].u=j;
edge[e].w=w;
e++; } }
cout<<"boshlang'ich tugun > "; cin>>start;
cout<<"qisqa masofalar ro'yhati:";
bellman_ford(n, start-1);
system("pause>>void"); }

```

Bu yerda graf soddalashtirilgan qirralar ro'yhati ko'rinishida ifodalangan va foydalanuvchi tomonidan vazn matrisasi kiritiladi. Bellman-Ford algoritmining asosiy qismi  $m \cdot (n-1)$  marta bajariladi, tashqi sikl  $n-1$  [marta takrorlanadi](#), ichki sikl esa  $m$  marta.  $N$ -iterasiyadan voz kechish esa maqsadga muvofiqdir, chunki algoritm  $n-1$  ta iterasiyada ham o'z vazifasini to'liq amalga oshira oladi. Ammo tashqi siklni  $n$  marta amalga oshirish grafda manfiy sikl mavjudligini aniqlashga imkon beradi.

## S187. Piramidali saralash haqida tushuncha va uni ishlash algoritmi

Javob: Piramidali saralash algoritmi

Piramidali saralash algoritmining asosida binar daraxtning piramida deb ataluvchi maxsus turidan foydalanish yotadi. Bunday binar daraxt tugunlarining qiymati eng yaqin avlodlari qiymatidan doimo katta bo'ladi. Saralash jarayoni piramida qurilishidan boshlanadi. Bunda ro'yxatning aksimal elementi daraxtning eng yuqori tugunida joylashadi. So'ngra ushbu element ro'yxatning eng oxirgi navbatiga joylashtiriladi. Elementi olingan piramida esa qaytadan quriladi. Natijada daraxt ildizida kattalik bo'yicha ikkinchi o'rinda turadigan element joylashadi va uni ro'yxatning oxiridan bitta oldingi o'ringa o'tkaziladi. Protsedura barcha elementlar ro'yxatdagi o'z o'rinlarini egallagunlaricha davom etadi. Bu jarayonga mos algoritm quyidagi ko'rinishga ega:

- 
- *piramida qurish*
- *for  $i=l$  to  $N$  do*
- *piramida ildizini ro'yxatga ko'chirish*
- *piramidani qayta qurish*
- *end for*

Ushbu algoritmdagi piramida qurish va uni qayta shakllantirish jarayonlarini ko'rib o'tamiz. Bu jarayonlar algoritm effektivligiga ta'sir ko'rsatadi. Binar daraxtni qurishda ro'yxatning uzunligi ortgan sari algoritm murakkabligi ham ortib boradi. Piramida qurishda quyidagi mulohazalardan kelib chiqish mumkin: Ro'yxatning  $i$ -elementi eng yaqin avlodlarining  $2i$  va  $2i + 1$  pozitsiyalardan yozamiz. Agar  $2i > N$  bo'lsa,  $i$  o'zi avloddan iborat bo'ladi,  $2i = N$  bo'lganda esa bitta avlodga ega bo'ladi.

### S188. Ma'lumotlar tuzilmasi klassifikatsiyasi.

Javob: Ma'lumot - bu biror bir ob'ekt, jarayon, hodisa yoki voqelikni ifodalab (tasniflab) beruvchi belgi yoki belgilar majmuasidir.

Ma'lumotlar tuzilmasi quyidagicha klassifikatsiyalanadi:

1. Tuzilmadagi ma'lumotlarning o'zaro bog'liqligi bo'yicha;
2. Vaqt bo'yicha yoki dastur bajarilishi davomida tuzilmaning o'zgaruvchanligi bo'yicha;
3. Tuzilmaning tartiblanganligi bo'yicha;
4. Tuzilma tarkibi bo'yicha.

1. Tuzilmadagi ma'lumotlarning o'zaro bog'liqligi bo'yicha:

1. agar tuzilmalar elementlari o'zaro kuchsiz bog'langan bo'lsa, bunday tuzilmalar bog'lanmagan tuzilma deyiladi
  - vektor, massiv, satr, stek
2. agar tuzilmada elementlar o'zaro bog'langan bo'lsa, bunday tuzilma bog'langan tuzilma deyiladi
  - bog'langan ro'yxatlar

2. Vaqt bo'yicha yoki dastur bajarilishi davomida tuzilmaning o'zgaruvchanligi bo'yicha:

1. dastur bajarilishi yakunlanganga qadar tuzilmada o'zgarish bo'lmasa bunday tuzilmalar statik tuzilmalar deyiladi:
  - yozuv, massiv, satr, vector;
2. dastur bajarilishi davomida tuzilmada qisman o'zgarish bo'lsa, bunday tuzilmalar yarimstatik tuzilmalar deyiladi:
  - stek, dek, navbat;
3. dastur bajarilishi davomida tuzilmada to'liq o'zgarishlar bo'lsa, bunday tuzilmalar dinamik tuzilmalar deyiladi:
  - bog'lamli ro'yxat, daraxt, graf
3. Vaqt bo'yicha yoki dastur bajarilishi davomida tuzilmaning o'zgaruvchanligi bo'yicha:

dastur bajarilishi yakunlanganga qadar tuzilmada o'zgarish bo'lmasa bunday tuzilmalar statik tuzilmalar deyiladi: u yozuv, massiv, satr, vector;

dastur bajarilishi davomida tuzilmada qisman o'zgarish bo'lsa, bunday tuzilmalar yarimstatik tuzilmalar deyiladi:

stek, dek, navbat;

dastur bajarilishi davomida tuzilmada to'liq o'zgarishlar bo'lsa, bunday tuzilmalar dinamik tuzilmalar deyiladi:

bog'lamli ro'yxat, daraxt, graf

#### 4. Tuzilma tarkibi bo'yicha:

Tuzilmalar tarkibi bo'yicha oddiy va integrallashgan (tarkibli - murakkab) turlarga ajratiladi.

Oddiy tuzilma tarkibidagi elementlar bitdan boshqa qismga ajralmaydigan tuzilma hisoblanadi.

Fizik tuzilmalar nuqtai nazaridan qaraganda oddiy tuzilma uchun uning o'lchami va komp'yuter xotirasiga joylashish usuli aniq.

Mantiqiy tuzilma nuqtai nazaridan qaraganda esa, oddiy tuzilma bo'linmas birlik hisoblanadi.

Integrallashgan (murakkab) tuzilmalar boshqa oddiy yoki murakkab tuzilmalardan tashkil topgan bo'ladi.

### **S189. “Ma'lumotlar tuzilmasi va algoritmlar” fanining maqsad va vazifalari**

Javob:

Ma'lumotlar tuzilmasi (MT) (ing.: data structure) - bu tuzilmani tashkil qiluvchi elementlar (ma'lumotlar) va ular orasidagi bog'liqlikni ifodalovchi munosabatlar majmuasi hisoblanadi.

Maqsadi (ing.: purpose) - turli dasturlash tizimlarida loyihalash usullari, ma'lumotlar tuzilmasini ishlab chiqish hamda algoritmlar bo'yicha nazariy va amaliy bilimlar berish.

Vazifasi (ing.: objective) – talabalarni turli xil ma'lumotlar tuzilmalari bilan tanishtirish, yangi tuzilmalarni ishlab chiqish va o'quv jarayonlariga tadbqiq etish usullari o'rgatishdan iboratdir.

Fanni o'qitishdan maqsad - talabalarga dasturlashda qo'llaniladigan ma'lumotlar tuzilmalari, ularning spetsifikatsiyasi va amalga oshirilishi bo'yicha bilimlarning nazariy asoslarini, ma'lumotlarni qayta ishlash algoritmлари va bu algoritmларni tahlil qilish, algoritmлар va ma'lumotlar strukturalarining o'zaro bog'liqligini o'rgatish hamda ularni amaliyotda tadbiq etish ko'nikmasini hosil qilishdan iborat. O'quvchilar bu fan orqali algoritmik fikrlash va muammolarni yechish uchun zarur algoritmларni o'rganishadi. Bu, muammolarni hal qilishda va dasturlashda kerak bo'lgan qadamlarni tuzish, qanday algoritmлар ishlatilishi kerakligini tushunish kabi mavzularni o'z ichiga oladi. Ma'lumotlar tuzilmasi va algoritmлар fanida o'quvchilar dasturlash tillarini Python, Java, C++, JavaScript o'rganadilar. Ushbu dasturlash tillari o'quvchilarga dasturlashda yaxshi tajribani taqdim etadi va ularga dasturlash sohasida ilmiy yondashuv beradi. Ma'lumotlar tuzilmasi va algoritmлар fanida o'quvchilar o'z fikrlarini bayon etish, muammolar uchun yaratuvchilikni rivojlantirish va boshqa dasturlash mutaxassislarining dasturlarini baholash va muhokama qilish imkoniyatlarini o'rganishadi.

## S190. Chiziqli ma'lumotlar tuzilmasi. Konteynerlar. Iteratirlar.

Javob: Ma'lumotlar strukturasi muhim xususiyati uning elementlarining tartiblanishidir. Ushbu xususiyatga asoslanib, strukturalarni *chiziqli* va *chiziqli bo'lmagan* tuzilmalarga bo'lish mumkin.

Chiziqli tuzilmalarga massiv, stek, navbat, jadval kiradi.

Massiv - bu ketma-ket ma'lumotlar namoyishi yordamida amalga oshiriladigan qat'iy o'lchamdagi chiziqli ma'lumotlar strukturasi. Har bir massiv elementi bir yoki bir nechta indekslar bilan aniqlanadi. Indeks - bu butun son bo'lib, uning qiymati massivdagi mos keladigan elementning o'rnini belgilaydi va ushbu elementga kirish uchun ishlatiladi. Alohida massiv elementlari o'zgarishi mumkin, lekin massiv elementlarining umumiy soni o'zgarishsiz qoladi. (Massiv uchun o'chirish va qo'shish amallari bajarilmaydi, faqat o'zgartirishlar kiritiladi). Indekslar soniga qarab bir o'lchovli va ko'p o'lchovli massivlar farqlanadi.

Stack - elementlarni kiritish va chiqarib tashlash imkonini beruvchi o'zgaruvchan o'lchamdagi chiziqli tuzilma. Kiritish va chiqarib tashlash uchun elementlarga kirish faqat strukturaning bir chetidan - yuqoridan mumkin (ma'lumotlar "oxirgi kir, birinchi chiqadi" tamoyili bo'yicha qayta ishlanadi). Stackni saqlash uchun ketma-ket yoki ulangan tasvirdan foydalanish mumkin.

Navbat - o'zgaruvchan o'lchamdagi chiziqli ma'lumotlar strukturasi. Elementlarni navbatdan faqat bir chetidan - navbat boshidan olib tashlash mumkin. Elementlarni kiritish faqat navbat oxirida mumkin. Ma'lumotlar birinchi kiruvchi, birinchi chiquvchi tamoyili asosida qayta ishlanadi. Elementlarga kirish va tugatish ko'rsatkichlari yordamida amalga oshiriladi. Navbatni amalga oshirish uchun ketma-ket va ulangan tasvirlardan foydalanish mumkin.

Jadval - bu chiziqli ma'lumotlar strukturasi bo'lib, uning har bir elementi ma'lum bir kalit qiymati bilan tavsiflanadi va uning elementlariga kirish kalit orqali amalga oshiriladi. Kalit struktura elementlarini kiritish va chiqarib tashlash imkonini beradi. Jadvalni amalga oshirish uchun ketma-ket yoki ulangan tasvirdan foydalanish mumkin. Odatda jadvallar qandaydir printsip bo'yicha tartiblanadi (masalan, o'sish/kamayuvchi kalit qiymatlari).

Konteyner - bu STL klassi bo'lib, u ba'zi ma'lumotlar strukturasi funksionalligini, ya'ni bir nechta elementlarning omborini amalga oshiradi. Turli konteynerlarga misollar: vektor, stek, navbat, deque, string, set, map va boshqalar.

Turli konteynerlarda elementga kirishning turli usullari mavjud. Misol uchun, vektor va deque "tasodifiy kirish" deb ataladigan narsani ta'minlaydi, bu sizga indeks bo'yicha kirish orqali konteynerning istalgan elementi bilan ishlash imkonini beradi, stek va navbat esa konteynerning faqat eng tashqi elementlariga kirishga imkon beradi.



Konteynerlar elementlariga kirish uchun iterator tushunchasi mavjud. Iterator - bu indeks bo'yicha elementga kirish g'oyasini umumlashtirish va C tilidagi ko'rsatkichlarni umumlashtirish. Siz iteratorlarni aqlli ko'rsatkichlar deb hisoblashingiz mumkin.

Iteratorlar - konteyner bo'ylab harakatlanish uchun mo'ljallangan maxsus ob'ektlar. Iteratorlar sizga konteynerning joriy elementiga kirish va qo'shni elementlarga o'tish imkonini beradi. Konteynerning boshlang'ich elementiga ishora qiluvchi iterator start funksiyasi tomonidan qaytariladi.

### **S191. Sheyker saralash haqida tushuncha va uni ishlash algoritmi.**

Javob: Shakerni saralash ikki tomonlama pufakchali saralashdir. Bunda algoritm massivni avval chapdan o'ngga qayta ishlaydi, shu tariqa eng katta elementni massiv oxirigacha, keyin esa o'ngdan chapga, eng kichik elementni massiv boshiga o'tkazadi.

Pufakchani saralash usulini tahlil qilib, ikkita narsani ta'kidlash mumkin:

- agar massivning bir qismi bo'ylab harakatlanayotganda hech qanday almashtirish sodir bo'lmasa, u holda massivning bu qismi allaqachon tartiblangan va shuning uchun ko'rib chiqishdan chiqarib tashlash mumkin.
- Massiv oxiridan boshiga o'tishda minimal element birinchi holatga "suzadi", maksimal element esa faqat bitta pozitsiyani o'ngga siljitadi.

Ushbu ikki g'oya pufakchani saralash usulini o'zgartirishga olib keladi.

- Oxirgi almashtirishdan massivning oxirigacha (boshi) tartiblangan elementlar mavjud. Ushbu haqiqatni hisobga olgan holda, skanerlash massivning oxirigacha (boshiga) emas, balki ma'lum bir pozitsiyaga qadar amalga oshiriladi. Massivning tartiblangan qismining chegaralari har bir iteratsiyada 1 pozitsiyaga siljiydi.
- Massiv o'ngdan chapga va chapdan o'ngga navbatma-navbat skanerlanadi.
- Massiv barcha elementlar o'sish (kamayish) tartibda bo'lgunga qadar skanerdan o'tkaziladi.
- Massiv elementlarini ko'rish soni uning elementlarini tartiblash momenti bilan belgilanadi.

### **S192. Ma'lumotlarni xeshlash algoritmlari.**

Javob: Xeshlash yoki xeshlash(ing. xeshlash) - ixtiyoriy uzunlikdagi kirish ma'lumotlari massivini ma'lum bir algoritm yordamida amalga oshiriladigan belgilangan uzunlikdagi (chiqish) bit qatoriga aylantirish. Algoritmni amalga oshiradigan va o'zgartirishni amalga oshiradigan funktsiya deyiladi.

Hashing quyidagi hollarda qo'llaniladi:

- assotsiativ massivlarni qurishda;
- bir qator ma'lumotlar to'plamida dublikatlarni qidirishda;
- ma'lumotlar to'plami uchun noyob identifikatorlarni qurishda;
- ma'lumotlardan (signaldan) nazorat summalarini hisoblashda ma'lumotlarni saqlash va (yoki) uzatish paytida yuzaga keladigan xatolarni (tasodifiy yoki qasddan kiritilgan) keyinchalik aniqlash uchun;
- parollarni saqlashda himoya tizimlaridaxesh-kod ko'rinishida (xesh-kod yordamida parolni tiklash uchun ishlatiladigan xesh funktsiyasiga teskari funktsiya talab qilinadi);
- elektron imzoni ishlab chiqishda (amalda ko'pincha xabarning o'zi emas, balki uning "xesh tasviri" imzolanadi);
- va boshq.

Bir nechta tez-tez ishlatiladigan hash funktsiyalari mavjud. Ularning barchasi matematiklar va dasturchilar tomonidan ishlab chiqilgan. Keyinchalik o'rganish natijasida ularning ba'zilarida kamchiliklar borligi aniqlandi, ammo ularning barchasi kriptografik bo'lmagan ilovalar uchun maqbul deb topildi.

#### **MD5**

MD5 xesh funktsiyasi 128 bitli xesh qiymatini hosil qiladi. U dastlab kriptografiyada foydalanish uchun ishlab chiqilgan, ammo vaqt o'tishi bilan unda zaifliklar aniqlangan va bu endi bu maqsad uchun mos emas. Biroq, u hali ham ma'lumotlar bazasini qismlarga ajratish va fayllarni uzatishni tekshirish uchun nazorat summalarini hisoblash uchun ishlatiladi.

#### **SHA-1**

SHA "Secure Hash Algoritm" degan ma'noni anglatadi. SHA-1 algoritmning birinchi versiyasi bo'lib, keyinchalik uni SHA-2 ta'qib qilgan.

MD5 128 bitli xeshni hosil qilsa, SHA-1 160 bitli (20 bayt) hosil qiladi. Agar siz bu raqamni o'n oltilik formatda ifodalasangiz, u 40 belgidan iborat butun sonidir. MD5 singari, bu algoritm kriptografik ilovalar uchun ishlab chiqilgan, ammo tez orada unda ham zaifliklar topildi. Bugungi kunda u MD5 ga nisbatan hujumlarga nisbatan chidamliroq hisoblanadi.

#### **SHA-2**

Algoritmning ikkinchi versiyasi SHA-2 juda ko'p o'zgarishlarga ega. Ehtimol, eng ko'p ishlatiladigan SHA-256 bo'lib, Milliy standartlar va texnologiyalar instituti (NIST) MD5 va SHA-1 o'rniga foydalanishni tavsiya qiladi.

SHA-256 algoritmi 256 bitli xesh qiymatini qaytaradi, bu 64 belgidan iborat o'n oltilik qiymatdir. Garchi bu eng ideal variant bo'lmasa-da, hozirgi tadqiqotlar shuni ko'rsatadiki, bu algoritm xavfsizlikda MD5 va SHA-1 dan sezilarli darajada ustundir.

Agar biz ushbu algoritmni ishlash nuqtai nazaridan ko'rib chiqsak, uning yordamida xeshni hisoblash MD5 yoki SHA-1 dan foydalanishga qaraganda 20-30% sekinroq bo'ladi.

### SHA-3

Ushbu xeshlash algoritmi 2015 yil oxirida ishlab chiqilgan va hozirgacha keng qo'llanilmaydi. Ushbu algoritm o'zidan oldingi SHA-2 tomonidan qo'llanilgan algoritm bilan bog'liq emas.

SHA3-256 algoritmi avvalgi SHA-256 algoritmiga teng qo'llanilishi mumkin bo'lgan algoritm bo'lib, birinchisini hisoblash ikkinchisiga qaraganda bir oz ko'proq vaqt talab etadi.

### **S193. Ma'lumotlar tuzilmasi tushunchasi va ma'lumotni tasvirlash.**

Javob: *Ma'lumotlar tuzilmasi — bu ma'lumotlarni samarali o'qish va o'zgartirish imkonini beruvchi, ma'lumotlarni saqlash va boshqarishning bir formatga solingan shaklidir.*

Soddaroq qilib aytganda, ma'lumotlar tuzilmasi — bu ma'lumotlarning ma'lum bir strukturaga solingan, ular o'rtasida ma'lum bir bog'lanishlar yaratilgan va ular ustida ma'lum amallar bajaruvchi funksiyalardan tashkil topgan guruh. Eng sodda ma'lumotlar tuzilmasiga misol qilib **massiv (array)**ni ko'rsatishimiz mumkin.

Quyida keltirilgan ma'lumotlar tuzilmalari dasturlashda eng ko'p qo'llaniladigan tuzilmalardir. Ularga:

1. *Massiv (Array)*
2. *Bog'langan ro'yhat (Linked List)*
3. *Navbat (Queue)*
4. *Stek (Stack)*
5. *Hash jadvallar (Hash tables)*
6. *Daraxtlar (Trees)*
7. *Graflar (Graph)*

*Ma'lumotlarni vizualizatsiya qilishning qanday usullari mavjud?*

*Grafiklar va diagrammalar*

*Grafiklar va diagrammalar ma'lumotlarni vizuallashtirishning eng keng tarqalgan usullaridir.*

*Issiqlik xaritalari*

*Geografik xaritalar*

*Daraxtlar va ierarxiyalar*

*Tarmoqlar va grafiklar*

*Infografika*

*Jadvallar va matritsalar*

## S194. Graflarda Breadth first search (BFS) algoritmi.

Javob: BFS yoki Breadth First Search - kenglikdan birinchi bo'lib grafiklarni o'tkazish algoritmi. Grafik - bu cho'qqilarni bog'laydigan "cho'qqilar" va "qirralar" tuzilishi. Siz chekkalar bo'ylab bir cho'qqidan ikkinchisiga o'tishingiz mumkin va BFS bu darajani daraja bo'yicha bajaradi: avval u boshlang'ich nuqtaga eng yaqin bo'lgan barcha cho'qqilar bo'ylab ketadi, keyin esa chuqurroq boradi.

Bu shunday ko'rinadi: algoritm oldindan tanlangan cho'qqidan boshlanadi va birinchi "tashrif" va bu tepalikning barcha qo'shnilarini belgilaydi. Keyin u tashrif buyurilgan cho'qqilarning qo'shnilariga o'tadi, so'ngra xuddi shu printsip bo'yicha. Tarqalishining to'lqinga o'xshash xususiyatiga ko'ra, algoritm to'lqin algoritmi deb ham ataladi. BFS ikki mashhur o'tish algoritmlaridan biridir. Ikkinchisi DFS deb ataladi va birinchi chuqurlikdan o'tishni o'z ichiga oladi: birinchidan, algoritm grafikning chetlarini "chuqur" kesib o'tadi.

BFS algoritmi qanday ishlaydi?

Algoritm oddiy va intuitivdir. U ko'rilmagan cho'qqilar qolmaguncha grafikning cho'qqilarini kesib o'tadi va maqsad cho'qqigacha bo'lgan eng qisqa yo'lni hisoblab chiqadi. Uning ishini aniqroq ko'rsatish uchun algoritmni bosqichma-bosqich taqdim qilaylik.

Ishning boshlanishi. Siz har qanday cho'qqini boshlang'ich sifatida tanlashingiz mumkin. Algoritmning boshida barcha tepaliklar ko'rilmagan deb belgilanadi - ular "oq" deb nomlanadi. Algoritm qiladigan birinchi narsa - tashrif buyurilgan boshlang'ich cho'qqini belgilash ("kengaytirilgan" yoki "kulrang" atamaları ham qo'llaniladi). Agar u maqsad bo'lsa, algoritm shu erda tugaydi. Ammo ko'pincha bunday emas.

Qo'shnilarni qidiring. Algoritm boshlang'ich cho'qqining qaysi qo'shnilari borligini tekshiradi. Ular algoritm tomonidan topilgan tartibda "harakat navbati" ga qo'shiladi va "kulrang" sifatida ham belgilanadi. Bu boshlang'ich tepada "oq" qo'shnilar qolmaguncha davom etadi.

Keyingi cho'qqiga o'ting. Algoritm boshlang'ich cho'qqining barcha qo'shnilaridan o'tganda, uni butunlay chetlab o'tilgan deb belgilaydi. Bunday tepaliklar "qora" deb ham ataladi: algoritm ularga qaytmaydi. Keyin u "kulrang" cho'qqilardan biriga - boshlang'ichning qo'shnilariga o'tadi. Algoritm navbatdagi birinchi cho'qqini tanlaydi. Keyin harakatlar takrorlanadi: "qora" dan tashqari, tepalikning "qo'shnilari" navbatga qo'shiladi.

Ushbu cho'qqidan o'tganda, o'tish xuddi shu printsipga ko'ra takrorlanadi - navbatdagi birinchi cho'qqi. Bunday holda, u boshlang'ich cho'qqining ikkinchi qo'shnisi bo'ladi -

biz ular birinchi navbatda navbatga qo'shilganligini eslaymiz. Va faqat navbatdagi boshlang'ich cho'qqining qo'shnilari tugagach, algoritm tepaliklarning keyingi "darajasi" bo'ylab harakatlanadi. Shunday qilib, kenglikdan birinchi o'tishga erishiladi.

Algoritmning oxiri. Agar navbat bo'sh bo'lsa, bu boshqa "oq" va "kulrang" cho'qqilar qolmaganligini anglatadi. Algoritm tugaydi. Agar maqsad cho'qqisiga erishilmasa, bu boshlang'ich nuqtadan unga kirish imkoni yo'qligini anglatadi.

Agar algoritm butun grafikni bosib o'tmaguncha maqsad cho'qqisiga erishilsa, bu uning tugallanganligini ham anglatishi mumkin. Algoritm to'xtaydi, chunki vazifa bajariladi: maqsad cho'qqisiga eng qisqa yo'l topiladi.



## S195. Saralashning asosiy maqsadi. Saralash algoritmlari guruhi. Turg'un va noturg'un saralash

Javob: **Saralash** - tartiblash (Sorting Algorithms) deb, berilgan obyektlar ketma-ketligini ma'lum mantiqiy tartibda qayta joylashtirish jarayoniga aytiladi. Saralash bir necha ko'rsatkichlarga bog'liq bo'lishi mumkin. Misol uchun maktab jurnalida o'quvchilar familiyasi alifbo tartibiga ko'ra saralangan bo'ladi.

Masalan bizga sonlar qatori berilgan: 8, 23, 0, -50, 100 Bu qatorni kichigidan kattasiga qarab yoki kattasidan kichigiga qarab saralashimiz mumkin. Bu saralashni amalga oshirish jarayoni **Saralash algoritmi** deyiladi. Saralash jarayoni taqqoslashga asoslangan jarayon hisoblanadi. Yuqoridagi sonli qatorni kichigidan kattasiga qarab tartiblaganimizda -50, 0, 8, 23, 100 ko'rinishiga keladi. Biz buni qanday amalga oshirdik. Bunda har xil usuldan foydalanish mumkin va mana shu algoritm turlaridir Biz algoritmlardan bittasidan foydalanib yuqoridagi sonli qatorni tartiblaymiz. Avval, sonli qatordan eng kichigini topamiz va uni ro'yxatning boshiga qo'yamiz. Har bir sonni boshqasi bilan solishtirib chiqamiz. Agar son o'zidan keyingi sondan kichik bo'lsa, son shu joyida qoladi, agar katta bo'lsa sonlarning o'rnini almashtiramiz.

Saralash asosan ro'yxat, massiv elementlarida amalga oshiriladi. Masalan sizning sinfingizda 5 ta o'quvchi bor. Ularni familiyasini alifbo tartibida saralash mumkin.

Sonlar berilishi: 23, 54, 3, 22, 1, 45;

Eng kattasini boshiga o'tkazamiz: 23, 3, 22, 1, 45, 54;(54 soni har bir son bilan solishtirilib eng katta ekani aniqlandi, 45 esa o'z o'rnida turipti) Shu tartibni davom ettiramiz: 3, 22, 1, 23, 45, 54;(23 undan keyinda turuvchi eng katta son) Yuqoridagi amalni yana davom ettiramiz: 3, 1, 22, 23, 45, 54;(22 esa davomchi) Oxirgi marta almashtirishimiz quyidagi natijani beradi: 1, 3, 22, 23, 45, 54;(1 eng kichigi) Saralangan tartib quyidagi holatga keldi: 1, 3, 22, 23, 45, 54;

Saralash algoritmi turlari

- Bubble sort
- Selection sort
- Insertion sort
- Quick sort
- Merge sort

Turg'un (barqaror) saralash - saralash sodir bo'ladigan bir xil kalitlarga ega bo'lgan tartiblangan elementlarning nisbiy tartibini o'zgartirmaydigan saralash.

Noturg'un saralash - bu tasodifiy almashtirish turi (juda tasodifiy emas, albatta).

## S196. Graflarda qisqa yo'l(shortest path)ni aniqlash tushunchasi va uning xususiyatlari.

Javob: Graflardagi eng qisqa yo'l tushunchasi grafdagi ikkita cho'qqi orasidagi eng kam xarajat (yoki uzunlik) yo'lini topishga ishora qiladi. Bu yo'l chekka og'irliklarining minimal yig'indisi (agar grafik tortilgan bo'lsa) yoki qirralarning minimal soni (agar grafik tortilmagan bo'lsa) bilan belgilanishi mumkin.

Eng qisqa yo'l muammosining uchta asosiy turi mavjud:

- SSSP (yagona manbali eng qisqa yo'l): bir cho'qqidan boshqasiga eng qisqa yo'llarni toping
- APSP (barcha juftlik eng qisqa yo'l): barcha cho'qqilardan hammaga eng qisqa yo'llarni toping
- P2P (nuqtadan nuqtaga): bir cho'qqidan ikkinchisiga eng qisqa yo'lni toping

Agar SSSP muammosi uchun har bir cho'qqi uchun manbadan ushbu cho'qqigacha bo'lgan eng qisqa yo'lni belgilasak va tanlangan eng qisqa yo'lda oxirgi chekkani dan ga belgilasak, u holda qirralarning kichik to'plami **eng qisqa yo'llar daraxtini** hosil qiladi - bu struktura to'g'ridan-to'g'ri hamma narsani saqlamasdan har bir alohida yo'lni osongina olishingiz mumkin.

Eng qisqa yo'l muammosi minimal xarajatlar oqimi muammosining alohida holati bo'lib, u o'z navbatida chiziqli dasturlash muammosining alohida holatidir. Uzoq vaqt davomida umumiy chiziqli dasturlash masalalarini yechishning asosiy usuli simpleks usuli bo'lib, eng qisqa yo'lni topish juda mashaqqatli va yuqorida sanab o'tilgan barcha usullar undan yaxshiroq bo'lgan masalalar uchun samarali bo'lmagan. Biroq, birinchi bo'lib 70-80-yillarda Nesterov va Nemirovskiy ichki nuqta usulini (IPM) ishlab chiqdilar, bu amaliyotda tez va polinom murakkabligi bahosiga ega edi. Keyinchalik bu usul nafaqat chiziqli dasturlash masalalarini, balki umumiyroq masalalarni ham hal qilish uchun asosiy usul bo'ldi. Yaqinda undan oqimli vazifalar uchun va bilvosita APSP uchun bir qator yaxshilanishlar olingan.

Graflardagi eng qisqa yo'lning ba'zi xususiyatlari:

Optimallik: Eng qisqa yo'l optimaldir, ya'ni uning narxi (yoki uzunligi) ikki cho'qqi orasidagi barcha mumkin bo'lgan yo'llar orasida minimaldir.

Aniqlik: eng qisqa yo'lning og'irligi grafdagi berilgan juft cho'qqilar uchun aniq va yagonadir.

Substruktiv xususiyat: A va B cho'qqilari orasidagi eng qisqa yo'lni ko'rib chiqing. Agar bu yo'l C cho'qqisidan o'tsa, u holda A dan C gacha va C dan B gacha bo'lgan eng qisqa

yo'l ham eng qisqa yo'ldir. Bu xususiyat eng qisqa yo'l muammosini hal qilish uchun dinamik dasturlashdan foydalanishga imkon beradi.

Salbiy vaznli tsikllar yo'q: manfiy og'irliklarga ega bo'lgan grafiklarda chekka og'irliklarining yig'indisi manfiy bo'lgan tsikllarga ega bo'lish mumkin. Bunday grafiklarda eng qisqa yo'lni to'g'ri aniqlash har doim ham mumkin emas, chunki salbiy og'irlikka ega bo'lgan yo'lni ushbu halqadan bir necha marta o'tish orqali "yaxshilash" mumkin. Dijkstra algoritmi va Bellman-Ford algoritmi kabi ko'plab eng qisqa yo'l algoritmlari manfiy davrli grafiklarni ishlay olmaydi, Floyd-Uorshell algoritmi esa bunday grafiklarda eng qisqa yo'llarni topishga qodir, ammo salbiy sikllar to'g'ri ishlov berilmaydi.

## S197. Rekursiya tushunchasi va uning qo'llanilishi

Javob: **Rekursiya** — Funksiya o'ziga o'zi to'g'ridan-to'g'ri yoki qandaydir vosita orqali murojaat qilish jarayoniga rekursiya deyiladi va bunday funksiya rekursiv funksiya deb ataladi<sup>[1]</sup>. Rekursiv funksiya o'zini — o'zi chaqirgani uchun dasturchilar orasida quyi oldin rekursiya nimagligini tushunish kerak — Stephen Hawking<sup>[2]</sup>. Rekursiya [funktional dasturlashning](#) asosiy elementlaridan hisoblanadi. Rekursiya deyarli hamma joyda ishlatiladi. Ba'zi masalalarning [iterativ yechimi](#) juda ham uzun bo'lib ketishi mumkin. Rekursiya esa kodni bir necha barobar qisqartirib berishi mumkin. Aksariyat [tuzilmalar](#) va [algoritmnlarni](#) rekursiyasiz tasavvur qilib bo'lmaydi. [Tree](#), [Graph](#), [Heap](#), [Quick Sort](#), [Merge Sort](#), ... Bu [ro'yhatni](#) juda uzoq davom ettirish mumkin. Ayniqsa, murakkab tuzilmalar bo'lgan Tree va Graphlarda rekursiya har qadamda uchraydi.

### Kamchiligi

- Rekursiya har doim xotiradan qo'shimcha joy talab qiladi.
- Rekursiv yechimda xato qilish ehtimoli yuqori, chunki rekursiya juda ham chalg'ituvchi.
- Rekursiv yechimni xatosini topish qiyin.
- Murakkab algoritmni hisoblash qiyin.

**Fibonachchi ketma ketligining  $n$  — hadini rekursiya qism dastur orqali hisoblovchi dastur**

```
#include <iostream.h>
int fib(int);
int main()
{
 int n;
 cout << "n="; cin >> n;
 cout << fib(n) << endl;
 return 0;
}
int fib(int k)
{
 if (k == 0 || k == 1) return 1;
 else return fib(k - 1) + fib(k - 2);
}
```

## S198. Vaznli va vaznsiz graflar uchun algoritmlar

Javob: Vaznli grafik: Ushbu turdagi grafiklarda har bir chekka yoki tepaga og'irlik deb ataladigan raqamli qiymat beriladi. Og'irlangan grafiklar nafaqat ulanishlar mavjudligi, balki ularning ahamiyati yoki narxi ham muhim bo'lgan turli vaziyatlarni modellashtirish uchun ishlatiladi.

vaznsiz grafik: Bu grafik bo'lib, uning qirralari va cho'qqilari hech qanday raqamli qiymatlarga ega emas, faqat cho'qqilar orasidagi bog'lanishning mavjudligi yoki yo'qligi. O'lchovsiz grafiklar faqat munosabatlarning mavjudligi yoki yo'qligi muhim bo'lgan hollarda qo'llaniladi, ularning ahamiyatlilik darajasi emas.

Tarkibi va talablaridagi farqlar tufayli vaznli va tortilmagan grafiklar uchun turli xil algoritmlar mavjud.

Vaznli grafiklar:

1. Dijkstra algoritmi: Bu algoritm vaznli grafikda bir cho'qqidan qolgan barcha nuqtalarga eng qisqa yo'llarni topish uchun ishlatiladi. Bu manfiy og'irliklarsiz grafiklar uchun samarali.
2. Bellman-Ford algoritmi: Bu algoritm eng qisqa yo'llarni ham topadi, lekin chekkalarida manfiy og'irliklarni o'z ichiga olgan grafiklar bilan ishlashi mumkin. U bitta istisnosiz barcha grafiklarda ishlaydi - agar grafikda salbiy og'irlik sikli mavjud bo'lsa, u holda bu algoritm uni aniqlay oladi.
3. A\* (Yulduz) Algoritmi: Bu algoritm ikki cho'qqi orasidagi eng qisqa yo'lni topish uchun qo'llaniladi, bunda maqsadga erishishga yordam beruvchi evristik funksiya berilgan. U ko'pincha xaritada yo'nalishlarni topish kabi ilovalarda qo'llaniladi.

Vaznsiz grafiklar:

1. Breadth-First Search (BFS): Bu algoritm berilgan boshlang'ich cho'qqidan boshlab grafikni o'rganadi va joriy darajadagi barcha qo'shni cho'qqilar bo'ylab tarqaladi, so'ngra keyingi bosqichga o'tadi. Undan vaznsiz grafiklardagi eng qisqa yo'llarni topish uchun foydalanish mumkin.
2. Depth-First Search (DFS): Bu algoritm grafik tuzilishiga imkon qadar tezroq kirib borish orqali grafikni o'rganadi. Bundan tashqari, vaznsiz grafiklardagi yo'llarni topish uchun ham foydalanish mumkin, lekin eng qisqa yo'llarni topish shart emas.

Algoritmni tanlash muayyan muammoga, grafikning xususiyatlariga (vaznli yoki tortilmagan, salbiy og'irliklarning mavjudligi va boshqalar) va ishlash talablariga

bog'liq.

### S199. Rekursiv algoritmlar, ularning tahlili. Rekursiyaga doir misollar.

Javob: Rekursiya – o'z-o'zi orqali aniqlanuvchi ob'ekt hisoblanadi. Matematikada rekursiya yordamida bir qancha cheksiz to'plamlarni aniqlash mumkin, masalan, natural sonlar to'plami.

- Rekursiv funktsiya ta'rifi bir yoki bir nechtaga ega asosiy holatlar, funktsiya natijani beradigan kirish (lar) ni anglatadi ahamiyatsiz (takrorlanmasdan), va bitta yoki bir nechtagi rekursiv holatlar, dastur takrorlanadigan kirish (lar) ni anglatadi (o'zini o'zi chaqiradi). Masalan, faktorial funktsiyani tenglamalar bilan rekursiv ravishda aniqlash mumkin  $0! = 1$  va hamma uchun  $n > 0$ ,  $n! = n(n-1)!$ . Ikkala tenglama ham o'z-o'zidan to'liq ta'rifni tashkil etmaydi; birinchisi - asosiy, ikkinchisi - rekursiv holat. Asosiy ish rekursiya zanjirini uzganligi sababli, ba'zan uni "tugatuvchi ish" ham deyishadi.
- Rekursiv holatlarning ishi murakkab yozuvlarni oddiylariga ajratish sifatida qaralishi mumkin. To'g'ri ishlab chiqilgan rekursiv funktsiyasida, har bir rekursiv chaqiriq bilan, kiritish muammosi shunday soddalashtirilishi kerakki, oxir-oqibat asosiy holatga erishish kerak. (Oddiy sharoitlarda tugatish uchun mo'ljallanmagan funktsiyalar - masalan, ba'zilari tizim va server jarayonlari - bu bundan mustasno.) Asosiy ishni yozishga beparvolik qilish yoki uni noto'g'ri tekshirish, sabab bo'lishi mumkin cheksiz pastadir.

Masala. Fibonachchi sonini hisoblash uchun Fib funktsiyasini tuzing. Bunda quyidagilar berilgan:

$f_0=0$ ,  $f_1=1$ .

$i>1$  uchun  $f_i=f_{i-1}+f_{i-2}$ .

Rekursiyani qo'llanilgan funktsiya quyidagicha bo'ladi:

```
int Fib (int n)
```

```
{
```

```
if (n==0) return 0;
```

```
if (n==1) return 1;
```

```
return Fib(n-1)+Fib(n-2);
```

```
}
```

```
def fibonachchi(n):
 if n <= 1:
 return n
 else:
 return fibonachchi(n - 1) + fibonachchi(n - 2)

Test qilish
for i in range(6):
 print(fibonachchi(i)) # Natija: 0, 1, 1, 2, 3, 5
```



## S200. "Algoritm" atamasi: kelib chiqishi va rivojlanishi

Javob: Algoritm – (lotincha *algoritmus* – “ibtido” ma’nosini anglatadi; bu so’zning etimologiyasi buyuk alloma Al-Xorazmiy ismi bilan bog’liq. U Yevropada buzilgan shaklda “Algorithmi” tarzida ishlatilgan) ma’lum bir tipga oid hamma masalalarni yechishda ishlatiladigan jarayonlar tizimining muayyan tartibda bajarilishi haqidagi aniq qoida yoki qoidalar tizimi. 20 O’rta asrlarda sanoqning o’nli sistemasi bo’yicha to’rt arifmetik amal bajariladigan qoida algoritm deb atalgan. Bu qoidalarni fanga IX asrda o’zbek matematigi Muhammad al-Xorazmiy kiritgan. Uning “Arifmetika” asarining birinchi so’zi “Qola al-Xorazmiy” so’zlari lotin tilida “Dixit Algorithmi” (“Dediki alXorazmiy”) deb tarjima qilingan. Bu yerda al-Xorazmiy so’zi “Algorithmi” deb yozilgan. Shundan so’ng algoritm tushunchasi fanda e’tirof etilib, uning qo’llanilish doirasi kengaydi. XX asrda algoritmlar kibernetikaning nazariy va mantiqiy asosi sifatida tan olingan.

Kompyuter texnologiyalari taraqqiyoti bilan bog’liq holda odam bilan mashina orasida o’ziga xos “ko’prik” vazifasini bajaruvchi sun’iy tillar yaratildi, ular programmalashtirish jarayonida qo’llaniladi. Sun’iy tillar buyruqlarning tizimli algoritmiga tayanib yaratilgani uchun ular algoritmik tillar (programmalashtirish tillari) deb ham ataladi. Dastlabki shunday til 1958 yilda Parijda yaratilgan va “Algol” deb atalgan (Uning to’liq nomi inglizcha – “Algorithmic language”). Hozir jahonda 500 dan ortiq algoritmik tillar mavjud.

Algoritmlar asosan quyidagi shakllarda yoziladi: a) so’z ko’rinishida – tabiiy til yordamida; b) grafik ko’rinishida – tasvir, chizma va grafikli simvollar yordamida; c) psevdokod ko’rinishida – algoritmlarning yarim formallashtirilgan tavsifi bo’lib, u programmalashtirishning dastlabki bosqichlarida qo’llaniladi. Bunda programmalashtirish tillari unsurlaridan ham, tabiiy tildagi so’z va birikmalardan ham foydalaniladi; d) programma ko’rinishida – programmalashtirish tillaridagi belgilar yordamida.

### Xususiyatlari

- Aniqlik (deterministik) — har bir vaqtda algoritmning keyingi qadami uning mazkur holati bilan to’liq aniqlanadi.
- Tushunarliklik — **algoritm** faqat uni bajaruvchisi uchun tushunarli buyruqlardan iborat bo’lishi shart.
- Yakunlanishi — **algoritm** albatta chekli qadamdan so’ng yakunlashishi lozim.

### Turlari

- **Chiziqli algoritm** — barcha ko‘rsatmalari hech qanday shartsiz, faqat ketma-ket bajariladigan jarayonlar.
- **Tarmoqlanuvchi algoritm** — hisoblash jarayoni qandaydir berilgan shartning bajarilishiga qarab turli tarmoqlar bo‘yicha davom ettiriladigan va hisoblash jarayonida har bir tarmoq faqat bir marta bajariladigan jarayonlar.
- **Takrorlanuvchi algoritm** — biron bir shart tekshirilishi yoki qandaydir parametrlarning har xil qiymatlari asosida algoritmda takrorlanish yuz beradigan jarayonlar.
- **Aralash (kombinatsiyalashgan) algoritm** — tarkibida bir necha turdagi algoritmlar qatnashgan algoritmdir.

## S201 Algoritm tushunchasi. Algoritm xususiyatlari. Algoritm ijrochisi.

### Algoritm tushunchasi:

Algoritm - bu muayyan muammoni yechish uchun aniq ketma-ket harakatlar to'plami, ya'ni buyruklar ketma-ketligi. Algoritmilar turli muammolarni (matematik, tashkiliy, loyihalash va h.k.) yechish uchun qo'llaniladi.

### Algoritm xususiyatlari:

1. **Aniqlik:** Algoritm har bir qadami aniq (bir ma'noli) bo'lishi kerak. Algoritmida har qanday nobut qolmasligi lozim.
2. **Uzluksizlik:** Algoritmida barcha harakatlar mantiqiy ketma-ketlikda bajarilishi shart.
3. **Cheklangan uzunlik:** Algoritm cheklangan sonli qadamlardan iborat bo'lishi kerak.
4. **Natijalilik:** Algoritm muammoni maqbul vaqt ichida, to'g'ri yechimni topishi lozim.
5. **Universal:** Algoritm turli xil muammolar uchun qo'llanilishi mumkin.

### Algoritm ijrochisi:

Algoritm ijrochisi - bu algoritmdagi harakatlar ketma-ketligini bajaruvchi qurilma yoki dastur. Algoritm ijrochisi quyidagi xususiyatlarga ega bo'lishi kerak:

1. Aniq ko'rsatmalarni qabul qilish va ularni bajarish qobiliyati.
2. Harakatlarni ketma-ketlikda bajarish qobiliyati.
3. Takroriy harakatlarni bajarish qobiliyati.
4. Tashqi ma'lumotlarni qabul qilish va qayta ishlash qobiliyati.

Algoritm ijrochisi turli xil bo'lishi mumkin: kompyuter, mobil qurilma, robotlar, odamlar va h.k. Amaliyotda algoritmilar asosan kompyuter dasturlari tomonidan bajariladi.

**Algoritm**, amaliyotlarni bajarish uchun aniqlangan tartib va qadamli amalni belgilovchi qoidalardir. Algoritm kompyuter dasturlarini yaratishda keng qo'llaniladi. Uni bajarish uchun muayyan tartibda harakat qilish uchun aniqlangan qadamli amallar ketma-ketligi sifatida ta'riflash mumkin. Algoritmning boshqa tushunchalari va xususiyatlari quyidagilardir:

1. **Qadamli amal:** Algoritmni bajarishda bir necha qadamlar bo'lishi kerak. Bu qadamlar o'zaro bog'liq va tartiblangan bo'lishi lozim.
2. **Aniq:** Algoritm bajarilishi kerak bo'lgan vazifani to'liq, aniq va o'zlashtirilgan shaklda bajarishi lozim.
3. **Yuqori darajada ta'kidlash:** Algoritm bajarilishi lozim bo'lgan vazifani to'g'ri va to'la bajarilishini ta'minlaydi.
4. **Tasvirlash:** Algoritm tushunchasini o'rganish va amaliyotda tushuntirishni osonlashtiradigan murakkab tizimlarni yaratish uchun yordam beradi.

Algoritm ijrochisi, algoritmni amalda bajarishga murojaat etadi. Bu xususiyatlar o'zgaruvchan bo'lib, odatda kompyuter dasturlarini yaratishda va bajarganda dasturlarni o'zlashtirishda foydalaniladi. Algoritmalar tezroq va samarali dasturlar yaratish uchun zarur bo'lgan asosiy qismi tashkil etadi. Algoritmalar odatda matematik, hisob-kitob, grafikalar, qidiruv va boshqa sohalarda yaxshi ishlaydi.

Algoritm, amaliyotlarni bajarish uchun aniqlangan tartib va qadamli amalni belgilovchi qoidalardir. Algoritmalar murakkab dasturlarni yaratishda yoki ma'lumotlarni tahlil qilishda keng qo'llaniladi.

Algoritmning xususiyatlari quyidagilardir:

1. **Aniq:** Algoritm bajarilishi kerak bo'lgan vazifani to'liq, aniq va o'zlashtirilgan shaklda bajarishi lozim.
2. **Yuqori darajada ta'kidlash:** Algoritm bajarilishi kerak bo'lgan vazifani to'g'ri va to'la bajarilishini ta'minlaydi.
3. **Qadamli amal:** Algoritmni bajarishda bir necha qadamlar bo'lishi kerak. Bu qadamlar o'zaro bog'liq va tartiblangan bo'lishi lozim.
4. **Aniq muayyanlik:** Algoritmni bajarish uchun har bir qadamning aniq va bajarilishi kerak.
5. **Yuqori samaradorlik:** Algoritm yuzaga kelsa, uning bajarilishi kerak bo'lgan vazifani etkazish uchun yuqori darajada samarali bo'lishi lozim.

Algoritmalar boshqaruvchi, amaliyotlarni bajarish uchun qo'llaniladigan instruksiyalarni tashkil etadi. Bu instruksiyalar tartib va qadamli amalni ta'minlaydi.

Algoritm ijrochisi, algoritmni amalda bajarishga murojaat etadi. Uni muayyan va aniqlangan tartibda amalga oshirishi va to'g'ri natijalarni olish uchun javobgar bo'lib hisoblanadi. Algoritmalar dasturlash tillarida, kompyuterlarning masofaviy tizimlarida va boshqa sohalarda keng qo'llaniladi. Ular murakkab dasturlarni yaratish va samarali ishlashlar uchun yordam beradi.

Algoritm, amaliyotlarni bajarish uchun belgilangan qadamli amallar ketma-ketligidir. Algoritmning asosiy xususiyatlari quyidagilardir:

1. **Aniqlik:** Algoritm bajarilishi kerak bo'lgan vazifani aniq va to'liq bajarishi lozim.
2. **Qadamli amal:** Algoritmni bajarishda qadamlar ketma-ketligi bo'lishi kerak. Har qadam bir necha muayyan amallarni bajarishi lozim.
3. **Yuqori darajada ta'kidlash:** Algoritm bajarilishi kerak bo'lgan vazifani to'g'ri va to'la bajarilishini ta'minlaydi.
4. **O'zlashtirilganlik:** Algoritmning amal qilish jarayoni o'zlashtirilgan va tanlangan vazifani bajarishda foydalanish mumkin bo'lishi lozim.

Algoritmlar odatda kompyuter dasturlarini yaratishda, hisob-kitobda, ma'lumotlar tahlil qilishda va boshqa sohalarda foydalaniladi.

Algoritm ijrochisi, belgilangan algoritmlarni amalda bajaradi. U aniq, to'g'ri va samarali amalga oshirish uchun javobgar bo'ladi. Algoritmlar dasturlash tillarida, kompyuter dasturlarini yaratishda va ma'lumotlar analizida keng qo'llaniladi.

Algoritm, amaliyotlarni bajarish uchun belgilangan tartib va qadamli amalni aks ettiruvchi qoidalar jamlanmasidir. Algoritmning asosiy xususiyatlari quyidagilardir:

1. **Aniqlik:** Algoritm, bajarishi kerak bo'lgan vazifani to'liq, aniq va o'zlashtirilgan shaklda bajarishi lozim.
2. **Yuqori darajada ta'kidlash:** Algoritm bajarishi kerak bo'lgan vazifani to'g'ri va to'la bajarilishini ta'minlaydi.
3. **Qadamli amal:** Algoritmni bajarishda bir necha qadamlar bo'lishi kerak. Bu qadamlar o'zaro bog'liq va tartiblangan bo'lishi lozim.
4. **Aniq muayyanlik:** Algoritmni bajarish uchun har bir qadamning aniq va bajarilishi kerak.

Algoritm ijrochisi, algoritmni amalda bajarishga murojaat etadi. U algoritmda belgilangan tartib va qadamlarni amalda bajaradi. Algoritmni to'liq bajarish, to'g'ri natijalarni olish va yuqori samarali amalga oshirishga javobgar bo'lib hisoblanadi. Algoritmlar kompyuter dasturlarini yaratishda, ma'lumotlar tahlilida, grafik ishlarda va boshqa sohalarda keng qo'llaniladi.

Algoritm - bu amaliyotlarni bajarish uchun belgilangan tartib va qadamli amalni aks ettiruvchi qoidalaridir. Algoritmning xususiyatlari quyidagilardir:

1. **Aniqlik:** Algoritm, bajarishi kerak bo'lgan vazifani to'liq, aniq va o'zlashtirilgan shaklda bajarishi lozim.
2. **Yuqori darajada ta'kidlash:** Algoritm bajarishi kerak bo'lgan vazifani to'g'ri va to'la bajarilishini ta'minlaydi.
3. **Qadamli amal:** Algoritmni bajarishda bir necha qadamlar bo'lishi kerak. Bu qadamlar o'zaro bog'liq va tartiblangan bo'lishi lozim.
4. **Aniq muayyanlik:** Algoritmni bajarish uchun har bir qadamning aniq va bajarilishi kerak.

Algoritm ijrochisi, algoritmni amalda bajarishga murojaat etadi. U algoritmda belgilangan tartib va qadamlarni amalda bajaradi. Algoritmni to'liq bajarish, to'g'ri natijalarni olish va yuqori samarali amalga oshirishga javobgar bo'lib hisoblanadi. Algoritmlar kompyuter dasturlarini yaratishda, ma'lumotlar tahlilida, grafik ishlarda va boshqa sohalarda keng qo'llaniladi.



## S202 Algoritmning turlari va ularni tavsiflash usullari.

Algoritmning asosiy turlari quyidagilar:

1. **Chiziqli algoritmlar:**

- Algoritmning qadamlari aniq ketma-ketlikda bajariladi.
- Har bir qadamda faqat bitta harakatni bajaradi.
- Qadamlar ketma-ketligi o'zgarmaydi.

2. **Tarmoqlangan algoritmlar:**

- Algoritm qadamlarida shartli operatsiyalar mavjud.
- Algoritm jarayonida turli variantlardan biri tanlanadi.
- Algoritm tarmoqlari shart operatorlari asosida belgilanadi.

3. **Takroriy algoritmlar:**

- Algoritm qadamlaridan ba'zilar qayta-qayta bajariladi.
- Takroriy harakatlar shart operatorlari yoki takrorlash operatorlari yordamida amalga oshiriladi.
- Takroriy algoritmlar tezkor va samarali yechimlarni ta'minlaydi.

4. **Rekursiv algoritmlar:**

- Algoritm o'z-o'zini chaqiruvchi funksiyalarga ega.
- Rekursiya jarayonida algoritmning bir qismi qayta-qayta bajariladi.
- Rekursiv algoritmlar ma'lum bir holatda to'xtatilishi kerak.

Algoritmning tavsiflash usullari:

1. **Lingvistik usul:**

- Algoritm odatdagi til yordamida, so'zlar va jumlar yordamida tavsiflanadi.
- Oddiy va tushunarli bo'ladi, ammo aniqlik darajasi past.

2. **Blok-sxemalar:**

- Algoritm grafik tasvirlash usuli orqali ifodalanadi.
- Algoritm qadamlari to'rtburchaklar, shartli operatsiyalar romblarda tasvirlanadi.
- Blok-sxemalar aniqlik va tushunarlik bilan ajralib turadi.

3. **Pseudokod:**

- Algoritmni dasturlash tillariga o'xshash shartli yozuv orqali tavsiflanadi.
- Algoritm qadamlari qisqa va tushunarli yozuvlar orqali ifodalanadi.
- Dasturlash tillariga o'tish uchun qulay vosita hisoblanadi.

Har bir usulning o'z afzalliklari va kamchiliklari mavjud. Algoritmni tavsiflash usuli muammoning murakkabligi va maqsadga qarab tanlanadi.

Algoritm turli xil vazifalarni bajarish uchun foydalaniladi va ularni turli usullarda tavsiflash mumkin. Quyidagi algoritmning turli turlari va ularni tavsiflash usullari mavjud:

### 1. Asosiy Algoritm Turlari:

### **a. Qidiruv Algoritmлари:**

Bu algoritmлар berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi. Ular quyidagi usullarda tavsiflanadi:

- **Diskret qidiruv algoritmлари:** Diskret elementlarni (masalan, massiv yoki ro'yxat) qidirish uchun ishlatiladi.
- **Teskari qidiruv algoritmлари:** Ma'lumotlar to'plamidan kerakli ma'lumotni topish uchun ishlatiladi.

### **b. Sortirovka Algoritmлари:**

Bu algoritmлар ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi. Ular quyidagi usullarda tavsiflanadi:

- **Comparative Sortirovka:** Muqayise bilan tartiblash algoritmлари, muqayise operatsiyalari asosida ishlaydi.
- **Non-comparative Sortirovka:** Bu algoritmлар o'zaro almashtirish operatsiyalari orqali tartiblash uchun ishlatiladi.

### **c. Qidiruv va Sortirovka Qimmatbaho Algoritmлари:**

Bu algoritmлар ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

### **d. Ko'rsatkich Algoritmлари:**

Bu algoritmлар ko'rsatkichlarni o'rganish va ulardan foydalanish uchun ishlatiladi.

## **2. Xususiy Algoritmлар Turlari:**

### **a. Graf Algoritmлари:**

Bu algoritmлар grafik strukturasi ustida amal qilish va u bilan bog'liq vazifalarni hal qilish uchun ishlatiladi.

- **Qisqa Yo'l Topish:** Grafiklarda ikki uchun eng qisqa yo'li topish.
- **Graph Coloring:** Grafikning ranglangan turlarini aniqlash.
- **Minimum Spanning Tree (MST):** Grafikda eng kam ro'yhatni topish.

### **b. Shablon Algoritmлари:**

Bu algoritmлар ma'lumotlar ustida ishlaydigan shablon va asosiy vazifalarni bajarish uchun ishlatiladi. Masalan, rekursiv algoritm, dinamik dastur, va boshqalar.



### ***c. Matematikiy Algoritmalar:***

Bu algoritmalar sonlar va hisob kitob usullari, masalan, hisob-kitob, integratsiya, differensiallar va boshqalar bilan bog'liq vazifalarni bajarishda ishlatiladi.

#### **Tavsiflash Usullari:**

##### ***a. Matematik Tavsiflash:***

Bu usulda algoritmalar matematiklik ko'rikda tavsiflanadi, ya'ni ko'rib chiqiladi, va algoritmining ishlatilishi tavsiflanadi.

##### ***b. Ishlovlarni Tavsiflash:***

Bu usulda algoritmalar ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi. Bu, harakatlanuvchilar va ma'lumotlarning o'zgarishlarini tavsiflash uchun qo'llaniladi.

##### ***c. Grafik Tavsiflash:***

Bu usulda algoritmalar grafik shaklida tavsiflanadi, ya'ni, algoritmalar grafik usullari orqali ko'rsatiladi. Bu usul murakkab algoritmalarini tushuntirishda yordam beradi.

##### ***d. Kod Tavsiflash:***

Bu usulda algoritmalar dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi. Bu usul algoritmalarini amalda olib chiqishga imkon beradi.

### **Algoritmalarining Turlari va Tavsiflash Usullari**

Algoritmalar turli xil vazifalarni bajarish uchun foydalaniladi va ularni turli usullarda tavsiflash mumkin. Quyidagi algoritmalarining turli turlari va ularni tavsiflash usullari mavjud:

#### **1. Asosiy Algoritmalar Turlari:**

##### **a. Qidiruv Algoritmari:**

Bu algoritmalar berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi. Ular quyidagi usullarda tavsiflanadi:

- Diskret qidiruv algoritmari
- Teskari qidiruv algoritmari

## **b. Sortirovka Algoritmлари:**

Bu algoritmlar ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi. Ular quyidagi usullarda tavsiflanadi:

- Comparative Sortirovka
- Non-comparative Sortirovka

## **c. Qidiruv va Sortirovka Qimmatbaho Algoritmлари:**

Bu algoritmlar ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

## **d. Ko'rsatkich Algoritmлари:**

Bu algoritmlar ko'rsatkichlarni o'rganish va ulardan foydalanish uchun ishlatiladi.

## **2. Xususiy Algoritmлар Turlari:**

### **a. Graf Algoritmлари:**

Bu algoritmlar grafik strukturasi ustida amal qilish va u bilan bog'liq vazifalarni hal qilish uchun ishlatiladi.

- Qisqa Yo'l Topish
- Graph Coloring
- Minimum Spanning Tree (MST)

### **b. Shablon Algoritmлари:**

Bu algoritmlar ma'lumotlar ustida ishlaydigan shablon va asosiy vazifalarni bajarish uchun ishlatiladi.

### **c. Matematikiy Algoritmлар:**

Bu algoritmlar sonlar va hisob-kitob usullari, masalan, hisob-kitob, integratsiya, diferensiallar va boshqalar bilan bog'liq vazifalarni bajarishda ishlatiladi.

## **Tavsiflash Usullari:**

### **a. Matematik Tavsiflash:**

Bu usulda algoritmlar matematiklik ko'rikda tavsiflanadi, ya'ni ko'rib chiqiladi, va algoritmning ishlatilishi tavsiflanadi.

## **b. Ishlovlarni Tavsiflash:**

Bu usulda algoritmlar ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi.

## **c. Grafik Tavsiflash:**

Bu usulda algoritmlar grafik shaklida tavsiflanadi, ya'ni, algoritmlar grafik usullari orqali ko'rsatiladi.

## **d. Kod Tavsiflash:**

Bu usulda algoritmlar dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi.

## **Algoritm Turlari va Tavsiflash Usullari**

Algoritm turli xil vazifalarni bajarish uchun foydalaniladi va ularni turli usullarda tavsiflash mumkin. Quyidagi algoritmlarning turli turlari va ularni tavsiflash usullari mavjud:

### **1. Asosiy Algoritm Turlari:**

#### **a. Qidiruv Algoritmari:**

Bu algoritmlar berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi. Ular quyidagi usullarda tavsiflanadi:

- Diskret qidiruv algoritmlari
- Teskari qidiruv algoritmlari

#### **b. Tartiblash Algoritmari:**

Bu algoritmlar ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi. Ular quyidagi usullarda tavsiflanadi:

- Muqayisali tartiblash
- Tartiblashsiz tartiblash

#### **c. Qidiruv va Tartiblash Algoritmari:**

Bu algoritmlar ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

#### **d. Xususiyl Algoritmlar:**

Bu turdagi algoritmlar ma'lum bir muammoga yoki kontekstga mo'ljallangan. Masalan, matematik algoritmlar, graf algoritmlar, optimallashtirish algoritmlari va boshqalar.

### **2. Tavsiflash Usullari:**

#### **a. Matematik Tavsiflash:**

Bu usulda algoritmlar matematiklik ko'rikda tavsiflanadi, ya'ni ko'rib chiqiladi, va algoritmning ishlatilishi tavsiflanadi.

#### **b. Ishlovlarni Tavsiflash:**

Bu usulda algoritmlar ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi.

#### **c. Grafik Tavsiflash:**

Bu usulda algoritmlar grafik shaklida tavsiflanadi, ya'ni, algoritmlar grafik usullari orqali ko'rsatiladi.

#### **d. Kod Tavsiflash:**

Bu usulda algoritmlar dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi. Bu usul algoritmlarni amalda olib chiqishga imkon beradi.

Algoritm, amaliyotlarni bajarish uchun belgilangan qadamli amalni aks ettiruvchi qoidalaridir. Algoritmning turli turlari mavjud va ularni turli usullarda tavsiflash mumkin.

### **1. Algoritmning Turlari:**

#### **a. Qidiruv Algoritmlari:**

Bu algoritmlar berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi.

- Diskret qidiruv algoritmlari
- Teskari qidiruv algoritmlari

#### **b. Tartiblash Algoritmlari:**

Bu algoritmlar ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi.

- Muqayisali tartiblash
- Tartiblashsiz tartiblash

### **c. Qidiruv va Tartiblash Algoritmnlari:**

Bu algoritmnlar ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

### **d. Xususiyl Algoritmnlar:**

Bu turdagi algoritmnlar ma'lum bir muammoga yoki kontekstga mo'ljallangan. Misol uchun, matematik algoritmnlar, graf algoritmnlar, optimallashtirish algoritmnlari va boshqalar.

## **2. Algoritmning Tavsiflash Usullari:**

### **a. Matematik Tavsiflash:**

Bu usulda algoritmnlar matematiklik ko'rikda tavsiflanadi va algoritmning ishlatilishi tavsiflanadi.

### **b. Ishlovlarni Tavsiflash:**

Bu usulda algoritmnlar ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi.

### **c. Grafik Tavsiflash:**

Bu usulda algoritmnlar grafik shaklida tavsiflanadi, ya'ni, algoritmnlar grafik usullari orqali ko'rsatiladi.

### **d. Kod Tavsiflash:**

Bu usulda algoritmnlar dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi. Bu usul algoritmnlarni amalda olib chiqishga imkon beradi.

Algoritm tushunchasi amaliyotlarni bajarish uchun belgilangan tartib va qadamli amalni aks ettiruvchi qoidalar jamlanmasidir. Algoritmning turli turlari mavjud va ularni turli usullarda tavsiflash mumkin.

## **1. Algoritmning Turlari:**

### **a. Qidiruv Algoritmnlari:**

Bu algoritmlar berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi.

- Diskret qidiruv algoritmlari
- Teskari qidiruv algoritmlari

#### **b. Tartiblash Algoritmlari:**

Bu algoritmlar ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi.

- Muqayisali tartiblash
- Tartiblashsiz tartiblash

#### **c. Qidiruv va Tartiblash Algoritmlari:**

Bu algoritmlar ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

#### **d. Xususiy Algoritmlar:**

Bu turdagi algoritmlar ma'lum bir muammoga yoki kontekstga mo'ljallangan. Masalan, matematik algoritmlar, graf algoritmlar, optimallashtirish algoritmlari va boshqalar.

### **2. Algoritmning Tavsiflash Usullari:**

#### **a. Matematik Tavsiflash:**

Bu usulda algoritmlar matematiklik ko'rikda tavsiflanadi va algoritmning ishlatilishi tavsiflanadi.

#### **b. Ishlovlarni Tavsiflash:**

Bu usulda algoritmlar ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi.

#### **c. Grafik Tavsiflash:**

Bu usulda algoritmlar grafik shaklida tavsiflanadi, ya'ni, algoritmlar grafik usullari orqali ko'rsatiladi.

#### **d. Kod Tavsiflash:**

Bu usulda algoritmlar dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi. Bu usul algoritmlarni amalda olib chiqishga imkon beradi.

## **Algoritmlarning Turlari va Tavsiflash Usullari**

Algoritmlar turli xil vazifalarni bajarish uchun foydalaniladi va ularni turli usullarda tavsiflash mumkin. Quyidagi algoritmlarning turli turlari va ularni tavsiflash usullari mavjud:

### **1. Asosiy Algoritmlar Turlari:**

#### **a. Qidiruv Algoritmlari:**

Bu algoritmlar berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi. Ular quyidagi usullarda tavsiflanadi:

- Diskret qidiruv algoritmlari
- Teskari qidiruv algoritmlari

#### **b. Tartiblash Algoritmlari:**

Bu algoritmlar ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi. Ular quyidagi usullarda tavsiflanadi:

- Muqayisali tartiblash
- Tartiblashsiz tartiblash

#### **c. Qidiruv va Tartiblash Algoritmlari:**

Bu algoritmlar ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

#### **d. Xususiy Algoritmlar:**

Bu turdagi algoritmlar ma'lum bir muammoga yoki kontekstga mo'ljallangan. Masalan, matematik algoritmlar, graf algoritmlar, optimallashtirish algoritmlari va boshqalar.

### **2. Algoritmlarning Tavsiflash Usullari:**

#### **a. Matematik Tavsiflash:**

Bu usulda algoritmlar matematiklik ko'rikda tavsiflanadi, ya'ni ko'rib chiqiladi, va algoritmning ishlatilishi tavsiflanadi.

#### **b. Ishlovlarni Tavsiflash:**

Bu usulda algoritmlar ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi.

### **c. Grafik Tavsiflash:**

Bu usulda algoritmlar grafik shaklida tavsiflanadi, ya'ni, algoritmlar grafik usullari orqali ko'rsatiladi.

### **d. Kod Tavsiflash:**

Bu usulda algoritmlar dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi. Bu usul algoritmlarni amalda olib chiqishga imkon beradi.

Algoritm tushunchasi amaliyotlarni bajarish uchun belgilangan tartib va qadamli amalni aks ettiruvchi qoidalar jamlanmasidir. Algoritmning turli turlari mavjud va ularni turli usullarda tavsiflash mumkin.

## **1. Algoritmning Turlari:**

### **a. Qidiruv Algoritmlari:**

Bu algoritmlar berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi.

- Diskret qidiruv algoritmlari
- Teskari qidiruv algoritmlari

### **b. Tartiblash Algoritmlari:**

Bu algoritmlar ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi.

- Muqayisali tartiblash
- Tartiblashsiz tartiblash

### **c. Qidiruv va Tartiblash Algoritmlari:**

Bu algoritmlar ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

### **d. Xususiy Algoritmlar:**

Bu turdagi algoritmlar ma'lum bir muammoga yoki kontekstga mo'ljallangan. Misol uchun, matematik algoritmlar, graf algoritmlar, optimallashtirish algoritmlari va boshqalar.



## **2. Algoritmning Tavsiflash Usullari:**

### **a. Matematik Tavsiflash:**

Bu usulda algoritmlar matematiklik ko'rikda tavsiflanadi va algoritmning ishlatilishi tavsiflanadi.

### **b. Ishlovlarni Tavsiflash:**

Bu usulda algoritmlar ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi.

### **c. Grafik Tavsiflash:**

Bu usulda algoritmlar grafik shaklida tavsiflanadi, ya'ni, algoritmlar grafik usullari orqali ko'rsatiladi.

### **d. Kod Tavsiflash:**

Bu usulda algoritmlar dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi. Bu usul algoritmlarni amalda olib chiqishga imkon beradi.

Algoritm, amaliyotlarni bajarish uchun belgilangan tartib va qadamli amalni aks ettiruvchi qoidalar jamlanmasidir. Algoritmning turli turlari mavjud va ularni turli usullarda tavsiflash mumkin.

## **1. Algoritmning Turlari:**

### **a. Qidiruv Algoritmnlari:**

Bu algoritmlar berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi.

- Diskret qidiruv algoritmlari
- Teskari qidiruv algoritmlari

### **b. Tartiblash Algoritmnlari:**

Bu algoritmlar ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi.

- Muqayisali tartiblash
- Tartiblashsiz tartiblash

### **c. Qidiruv va Tartiblash Algoritmnlari:**

Bu algoritmlar ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

#### **d. Xususiy Algoritmlar:**

Bu turdagi algoritmlar ma'lum bir muammoga yoki kontekstga mo'ljallangan. Masalan, matematik algoritmlar, graf algoritmlar, optimallashtirish algoritmlari va boshqalar.

### **2. Algoritmning Tavsiflash Usullari:**

#### **a. Matematik Tavsiflash:**

Bu usulda algoritmlar matematiklik ko'rikda tavsiflanadi va algoritmning ishlatilishi tavsiflanadi.

#### **b. Ishlovlarni Tavsiflash:**

Bu usulda algoritmlar ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi.

#### **c. Grafik Tavsiflash:**

Bu usulda algoritmlar grafik shaklida tavsiflanadi, ya'ni, algoritmlar grafik usullari orqali ko'rsatiladi.

#### **d. Kod Tavsiflash:**

Bu usulda algoritmlar dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi. Bu usul algoritmlarni amalda olib chiqishga imkon beradi.

Algoritm, amaliyotlarni bajarish uchun belgilangan tartib va qadamli amalni aks ettiruvchi qoidalar jamlanmasidir. Algoritmning turli turlari mavjud va ularni turli usullarda tavsiflash mumkin.

### **1. Algoritmning Turlari:**

#### **a. Qidiruv Algoritmlari:**

Bu algoritmlar berilgan ma'lumotlar o'rtasidan kerakli ma'lumotni topishda ishlatiladi.

- Diskret qidiruv algoritmlari
- Teskari qidiruv algoritmlari

## **b. Tartiblash Algoritmлари:**

Bu algoritmлар ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi.

- Muqayisali tartiblash
- Tartiblashsiz tartiblash

## **c. Qidiruv va Tartiblash Algoritmлари:**

Bu algoritmлар ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.

## **d. Xususiyl Algoritmлар:**

Bu turdagi algoritmлар ma'lum bir muammoga yoki kontekstga mo'ljallangan. Misol uchun, matematik algoritmлар, graf algoritmлар, optimallashtirish algoritmлари va boshqalar.

## **2. Algoritmning Tavsiflash Usullari:**

### **a. Matematik Tavsiflash:**

Bu usulda algoritmлар matematiklik ko'rikda tavsiflanadi va algoritmning ishlatilishi tavsiflanadi.

### **b. Ishlovlarni Tavsiflash:**

Bu usulda algoritmлар ma'lumotlar ustida amal qilishni ta'minlaydi va shu natijada o'zgarishlar yuzaga keladi.

### **c. Grafik Tavsiflash:**

Bu usulda algoritmлар grafik shaklida tavsiflanadi, ya'ni, algoritmлар grafik usullari orqali ko'rsatiladi.

### **d. Kod Tavsiflash:**

Bu usulda algoritmлар dasturlash tili (masalan, Python, Java, C++, vb.) kod yordamida tavsiflanadi. Bu usul algoritmларni amalda olib chiqishga imkon beradi.

Анализ приостановлен

3.5

ChatGPT

I see you need a summary of the algorithm types and description methods. Here's a concise version:

---

## **Algoritmlarning Turlari va Tavsiflash Usullari**

**Algoritmlar turli vazifalarni bajarish uchun belgilangan tartib va qadamli amalni aks ettiruvchi qoidalardir.** Algoritmlarning turli turlari mavjud va ularni quyidagi usullarda tavsiflash mumkin:

### **Algoritm Turlari:**

- **Qidiruv Algoritmlari:** Ma'lumotlar o'rtasidan kerakli ma'lumotni topish uchun ishlatiladi. Ular diskret va teskari qidiruv shakllarida bo'lishi mumkin.
- **Tartiblash Algoritmlari:** Ma'lumotlarni o'zlarini tartiblash uchun ishlatiladi. Bu algoritmlar muqoyisali tartiblash va tartiblashsiz tartiblash shakllarida bo'lishi mumkin.
- **Qidiruv va Tartiblash Algoritmlari:** Ma'lumotlar ustida amal qilish va ularni qidirish yoki tartiblashda ishlatiladi.
- **Xususiy Algoritmlar:** Ma'lum bir muammoga yoki kontekstga mo'ljallangan algoritmlar. Misol uchun, matematik algoritmlar, graf algoritmlar, optimallashtirish algoritmlari va boshqalar.

### **Algoritmning Tavsiflash Usullari:**

- **Matematik Tavsiflash:** Algoritmlarni matematiklik ko'rikda tavsiflash va ishlatilishi.
- **Ishlovlarni Tavsiflash:** Ma'lumotlar ustida amal qilishni ta'minlash va o'zgarishlarni yuzaga chiqarish.
- **Grafik Tavsiflash:** Algoritmlarni grafik shaklida tavsiflash, ya'ni, algoritmlar grafik usullari orqali ko'rsatish.
- **Kod Tavsiflash:** Algoritmlarni dasturlash tili kod yordamida tavsiflash. Bu usul algoritmlarni amalda olib chiqishga imkon beradi

## S203 Algoritmnlarni ifodalash shakllari.

Algoritmnlarni ifodalash uchun quyidagi asosiy shakllarga tayaniladi:

1. **Tabiiy (yozma) til:**

- Algoritmni oddiy so'zlar, jumlar va paragraflar yordamida bayon qilish.
- Tushunish oson, ammo aniqlik va rasmiylik darajasi past.
- Masalan: "Birinch qadamda yangi matn faylini oching. Keyin uni tahrirlashni boshlang. Tahrir qilishni tugatgach, faylni saqlang."

2. **Blok-sxemalar:**

- Algoritm qadamlarini grafik tasvirlash orqali ifodalash.
- To'rtburchaklar - operatsiyalar, romblar - shartlar, oqimlar - ularga o'tish.
- Aniq va ko'rgazmali, ammo murakkab alqoritmlar uchun qo'llanish qiyinroq.

3. **Psevdokod:**

- Algoritmni dasturlash tillari sintaksisiga yaqin bo'lgan shartli yozuv orqali ifodalash.
- Muhim operatsiyalar, shartlar, takrorlar va boshqa tuzilmalarni qisqa va aniq yozish.
- Algoritm tushunish oson, dasturlash tillariga o'tish uchun qulay.

4. **Formal tillar:**

- Algoritmni matematik va mantiqiy formulalar, shartlar, xususiyatlar orqali ifodalash.
- Masalan, Xarakteristik funktsiya yordamida ifodalangan algoritm:  
 $F(x) = 1$ , agar  $x$  toq bo'lsa  
 $F(x) = 0$ , agar  $x$  juft bo'lsa

Algoritmni ifodalash shakli muammoning xususiyatlari, algoritmning murakkabligi va ifodalash maqsadlariga bog'liq ravishda tanlanadi. Odam va kompyuter uchun tushunarli bo'lishi muhimdir.

Algoritm ifodalashda turli shakllar mavjud bo'lishi mumkin. Quyidagi shakllar odatda amalga oshiriladi:

1. **Matematik ifodalar:** Algoritmnlarni matematik formulalar yordamida ifodalash mumkin. Misol uchun, algoritmni formalizatsiyalash uchun o'zgaruvchilar, funktsiyalar, va mantiqiy ifodalar ishlatiladi.
2. **O'zgaruvchilar va operatorlar:** Algoritmnlar o'zgaruvchilar va operatorlar yordamida ifodalash mumkin. Bu ko'rsatmalarni beruvchi, if-else operatorlari, looplar va boshqa amallar o'z ichiga oladi.
3. **Pseudocode:** Bu shaklda, algoritm boshqa dastur tillariga bog'liq qilib yoziladi, lekin aniqlik bilan dastur tilida yozilmaydi. Bu, oddiy ko'rsatmalar bilan alqoritmlarni ta'riflash uchun o'girilgan shakl.

4. **Flowchart (suhbatdiagramma):** Bu algoritmning qadamli tavsifini ko'rsatish uchun grafiyani ishlatadi. Qadamli jarayonlar va shartlar ifodalangan. Har bir qadam va shart bir grafik element sifatida ifodalangan.
5. **Structured English:** Bu algoritmning natural tilda yozilgan varianti. Bu, oddiy ingliz tilida algoritmni ta'riflash usuli. Dastlabki hujjat ma'lumotlar to'plamidan boshlanadi va dasturni amalga oshirish uchun qadam qadamga tavsiflangan.
6. **Programing Language Code:** Algoritmilar ko'p dastur tillarida yozilishi mumkin. Ko'p dastur tillarida yozilgan kod, algoritmnlarni bajarish uchun foydalaniladi.

Ushbu shakllar bir nechta algoritm turlari uchun o'zgartirib turilishi mumkin. Algoritmilar, xususan dastur ishlab chiqish jarayonida, bir nechta ifodalash shakllari yordamida o'rganiladi va tushiriladi.

## S204 Algoritmnlarni vaqt va hajm bo'yicha baholash. Ko'phadlar qiymatlarini hisoblashda Gorner sxemasi

Algoritmnlarni baholash uchun asosan ikki xil mezon ishlatiladi:

### 1. Vaqt muddati (time complexity):

- Algoritm ishlashining vaqt jihatidan samaradorligini o'lchovchi mezon.
- Algoritm ishlashi uchun ketadigan vaqtni aniqlaydi.
- Asosan, bajarilishi kerak bo'lgan amallar soni orqali o'lchanadi.
- Vaqt murakkabligi algoritmning kiritilgan ma'lumotlar hajmiga bog'liq ravishda o'zgarishi mumkin.
- Vaqt murakkabligi quyidagi asosiy belgilar bilan ifodalanadi:
  - $O(1)$  - doimiy vaqt (ma'lumotlar hajmidan qat'iy nazar)
  - $O(\log n)$  - logarifmik vaqt
  - $O(n)$  - chiziqli vaqt
  - $O(n \log n)$  - chiziqli logarifmik vaqt
  - $O(n^2)$  - kvadratik vaqt
  - $O(2^n)$  - eksponentsiyal vaqt

### 2. Xotira hajmi (space complexity):

- Algoritm ishlashi uchun kerak bo'lgan xotira hajmini aniqlaydi.
- Algoritm ishlashi uchun vaqtinchalik va doimiy xotira sarf-xarajatlarini o'lchaydi.
- Xotira murakkabligi ham kiritilgan ma'lumotlar hajmiga bog'liq bo'ladi.
- Xotira murakkabligi quyidagi asosiy belgilar bilan ifodalanadi:
  - $O(1)$  - doimiy xotira
  - $O(n)$  - chiziqli xotira
  - $O(n^2)$  - kvadratik xotira

### Gorner sxemasi:

Ko'phadlar qiymatini tez hisoblash uchun Gorner sxemasi qo'llaniladi. Bu sxema o'zgaruvchilarga qiymatlar berib, ko'phadning qiymatini hisoblab beradi.

Gorner sxemasi, bir nechta algoritmni hisoblash uchun yaxshi va effektiv bir usuldir. Agar ularning biri sifatida algoritmni ko'rsatgan bo'lsangiz, Gorner sxemasi yordamida algoritmnlarni vaqt va hajm bo'yicha baholash osonlik bilan amalga oshirilishi mumkin.

Gorner sxemasi (Gorner's Scheme) ushbu usul bilan ko'plab darajadagi polinomlarni hisoblash mumkin. U, hisoblashni "kuchini darajalarga bo'lish" operatsiyasi orqali o'zlashtiradi. Bu qo'llanma, algoritmni bajarish vaqtini va hajmini baholashda yaxshi natijalarga ega bo'lishga yordam beradi.

Algoritmni baholashda, qo'shish va ko'paytirish operatsiyalari barcha vaqt va hajmni hisoblashda muhimdir. Agar bizga har bir operatsiyani o'zining o'zlashtirilgan vaqti

va hajmi bo'yicha berilgan bo'lsa, algoritmnining jami vaqt va hajmini topish oson bo'ladi. Va bizga berilgan, qo'shish operatsiyasi uchun 1 vaqtda va 1 birlik hajm sarflanadi, ko'paytirish operatsiyasi uchun 1 vaqtda va 1 birlik hajm sarflanadi. Endi biz bu polinomni Horner sxemasi yordamida hisoblashni ko'ramiz. Yuqoridagi formulada ko'rsatilgan bo'lib, biz yana ham barcha ko'paytirish operatsiyalarini yozmaganmiz. Faqatgina qo'shish operatsiyalari qolgan.

Endi, operatsiyalarni hisoblashni boshlaymiz:

1. 1 vaqtda va 1 birlik hajm uchun 4xni hisoblaymiz.
2. 1 vaqtda va 1 birlik hajm uchun -5 ni qo'shamiz.
3. 1 vaqtda va 1 birlik hajm uchun 2 ni qo'shamiz.
4. 1 vaqtda va 1 birlik hajm uchun 3 ni qo'shamiz.

Jami vaqt: 4 vaqtda. Jami hajm: 4 birlikda.

Shuningdek, bizda boshqa parametrlar mavjud bo'lsa, ularni ham muhokama qilishimiz mumkin. Bu, algoritmlar hisoblashida juda foydali bo'ladi, chunki ularni taqqoslash va optimallashtirishga yordam beradi.



## S205 Chiziqli algoritmlar. Sikllar. Integrallarni taqribiy hisoblash usullari, samaradorligi.

Chiziqli algoritmlar, bir funksiyaning integralini taqribiy hisoblash uchun qo'llaniladigan usullardan biridir. Bundan tashqari, sikllar yordamida integralni hisoblash usullari ham samaradorligini o'lchashda muhimdir. Mana shu chiziqli integrallarni taqribiy hisoblash usullari va ularning samaradorligini:

### 1. Chiziqli algoritmlar:

Chiziqli algoritmlar, bir funksiyaning integralini  $x$  kenglik bo'yicha chiziqli to'g'ri chiziqlar yordamida taqribiy hisoblash usullaridir. Ular quyidagi ko'rsatkichlar orqali taqribiy integralni topish uchun ishlatiladi:

- To'g'ri chiziqlar metodi
- Simmetrik chiziqlar metodi
- Simpson chiziqlar metodi

Bu usullar, funksiyaning integralini taqribiy hisoblashda mos keladigan  $x$  kenglik bo'yicha chiziqli to'g'ri chiziqlar va ularning kesishgan joylari bo'yicha qiymatlarni hisoblash asosida ishlaydi.

### 2. Sikllar:

Sikllar (loops) algoritm yaratishda yoki dastur qilishda turli turdagi vazifalarni takrorlash uchun ishlatiladi. Sikllar samaradorligini o'lchashda asosiy faktorlar quyidagilardir:

- **Ishonchli vaqt samaradorligi:** Sikllar biror ishonchli darajada ishga tushsa, ular samaradorligini oshirishi mumkin. Ammo, aks holda, sikllarning samaradorligi past bo'lishi mumkin.
- **Hajm samaradorligi:** Sikllar qanchalik ko'p xotira sarflaydi, shuningdek, ularning ishga tushishi uchun qanchalik vaqt sarflaydi?
- **Kod qiyinchiliklari:** Bazi sikllar kodni qiyinlashtirishi mumkin, shuning uchun, ularning samaradorligi past bo'lishi mumkin.

Yuqorida keltirilgan samaradorlik o'lchamlari, sikllarning to'g'ri yoki yomon samaradorligini aniqlashda yordam beradi. Agar sikl uzun va kengroq bo'lsa, u bilan ishlashning boshqa usullari muhim bo'ladi.

### 3. Integrallarni taqribiy hisoblash usullari samaradorligi:

Integralni taqribiy hisoblash usullari quyidagi muhim samaradorlik kriteriyalari bo'yicha baholash mumkin:

- **To'g'ri chiziqlar metodi:** Bu usul juda oddiy va oson, lekin darajadagi  $x$  kengligi bo'yicha to'g'ri chiziqlar sonini tanlashda foydalanish lozim. Agar  $x$  kengligi soni juda ko'p bo'lsa, usul kam samarador bo'ladi.
- **Simmetrik chiziqlar metodi:** Bu usul odatda to'g'ri chiziqlar metodi bilan bir xil samaradorlikka ega bo'lsa ham, uning o'rta qiymatlarni qo'llash yordamida integrallarni hisoblashda bir oz navbatda ishlatiladi.
- **Simpson chiziqlar metodi:** Bu usulga ikki chetli funksiya bo'lishi kerak. Ushbu usul, odatda, integrallarni hisoblashda eng samarali usul hisoblanadi, lekin agar funksiya toq bo'lsa, uning taqribiy integrallarini topishda xatolik yuzaga kelishi mumkin.

Integralni taqribiy hisoblash usullari samaradorlikni o'lchashda, yuqorida keltirilgan muhim qatorlar bo'yicha baholash mumkin. Ushbu usullar sodda funksiyalarda yaxshi natijalarni bermasligi mumkin, lekin murakkab funksiyalar uchun yaxshi taqribiy natijalar olishda yordam beradi.

#### 1. Chiziqli algoritmlar:

- Chiziqli algoritmlar - vaqt murakkabligi  $O(n)$  bo'lgan algoritmlar.
- Bunday algoritmlar ma'lumotlar to'plamini bir marta ketma-ket qarab chiqadi.
- Masalan, massivdagi barcha elementlarni yig'ish, massivdagi eng katta/eng kichik elementni topish, ma'lumotlarni saralash va h.k.
- Chiziqli algoritmlar samarali va tez ishlaydi, xotirani ko'p iste'mol qilmaydi.

#### 2. Sikllar:

- Sikllar - algoritmda takrorlanuvchi operatsiyalar uchun ishlatiladi.

#### 3.

- Sikllar orqali algoritmlar yanada qisqa va aniq ifodalanadi.
- Sikllar uchun vaqt murakkabligi esa  $O(n)$  bo'ladi.

#### 4. Integrallarni taqribiy hisoblash usullari:

- Integral - funksiyaning ma'lum oraliqda olingan yig'indisi.
- Integrallarni hisoblash uchun quyidagi asosiy usullar mavjud:

##### 1. Trapetsiya usuli:

1. Integrallangan funksiyaning to'g'ri chiziqlar bilan taxminiy ifodalash.
2. Vaqt murakkabligi  $O(n)$ , xotira murakkabligi  $O(1)$ .

##### 2. To'rtburchak usuli:

1. Integrallangan funksiyaning to'g'ri to'rtburchaklar bilan taxminiy ifodalash.

2. Vaqt murakkabligi  $O(n)$ , xotira murakkabligi  $O(1)$ .

3. **Simpson usuli:**

1. Integrallangan funksiyani ikkinchi darajali parabola bilan taxminiy ifodalash.
  2. Daqiqroq natija beradi, ammo vaqt murakkabligi  $O(n)$ , xotira murakkabligi  $O(1)$ .
- Bu usullar yordamida integral qiymati taqribiy aniqlanadi. Natijaning aniqligini oshirish uchun interval kichiklashtiriladi.
  - Samaradorligi: Trapetsiya va To'rtburchak usullari tezkor, Simpson usuli esa aniqroq.

Umumiy xulosa qilib aytganda, chiziqli algoritmlar va sikllar algoritmlarni samarali va tez bajariladigan qilib yozish uchun muhim tushunchalardir. Integral hisoblash usullari esa taxminiy natijalar olish imkonini beradi.

## S206 Matrisalarni ko'paytirish. Determinantni hisoblash. Algoritm ijrochisi.

Matrisalar bilan ishlashning bir necha muhim jarayonlari bor, masalan, matrisalarni ko'paytirish, determinantni hisoblash va algoritm ijrochisi. Biz ularni qisqa muddatda ko'rib chiqamiz:

### 1. Matrisalarni ko'paytirish:

Matrisalarni ko'paytirish - bu ikki yoki undan ko'p matrisani bitta matrisaga ko'paytirishdir. Agar bir matrisa AAA bo'lsa, ikkinchi matris BBB bo'lsa, ularni ko'paytirish tartibi quyidagicha bo'ladi:

$$C=A \cdot B \text{ } C=A \cdot B$$

Ko'paytirilayotgan ikki matrisning ko'lay qilib ko'paytirilishi uchun, ko'paytirish amaliyoti quyidagi shaklda ifodalangan bo'ladi:

$$c_{ij} = \sum_{k=1}^n a_{ik} \cdot b_{kj} \text{ } c_{ij} = \sum_{k=1}^n a_{ik} \cdot b_{kj}$$

Bu formulada,  $c_{ij}$  - C matrisasining i-qi qatorining j-qi ustunidagi element,  $a_{ik}$  - A matrisasining i-qi qatorining k-qi ustunidagi element, va  $b_{kj}$  - B matrisasining k-qi qatorining j-qi ustunidagi elementni ifodalaydi.

### 2. Determinantni hisoblash:

Matrisaning determinantni hisoblash - bu matrisaning bu axborot kengligi bo'yicha quyidagi shaklda hisoblanadi:

$$\det(A) = \sum_{\sigma} \operatorname{sgn}(\sigma) \cdot a_{1\sigma(1)} \cdot a_{2\sigma(2)} \cdot \dots \cdot a_{n\sigma(n)} \text{ } \det(A) = \sum_{\sigma} \operatorname{sgn}(\sigma) \cdot a_{1\sigma(1)} \cdot a_{2\sigma(2)} \cdot \dots \cdot a_{n\sigma(n)}$$

Bu yerda,  $\sigma$  - n ta raqam elementlarining tarkibiga ega bo'lgan barcha permutatsiyalar,  $\operatorname{sgn}(\sigma)$  - berilgan permutatsiyaning og'irligi (barcha transpozitsiyalar soni),  $a_{i\sigma(i)}$  - berilgan permutatsiyada i-qi qatorning  $\sigma(i)$ -chi ustunidagi element.

### 3. Algoritm ijrochisi:

Matrisalarni ko'paytirish va determinantni hisoblash uchun algoritm ijrochilari quyidagilar:

- **Matrisalarni ko'paytirish uchun algoritm:**
  - Eng asosiy algoritm, matrisalarni ko'paytirishda ikki matrisning elementlarini ko'paytirish formulasi yordamida ishlatiladi.
  - Bu algoritm  $O(n^3)$  vaqt talab qiladi,  $n$  - matrisa o'lchamini ifodalaydi.
- **Determinantni hisoblash uchun algoritm:**
  - Determinant hisoblash uchun ko'plab usullar mavjud, ammo asosiy algoritmlar Gauss eliminatsiyasi va ko'paytirish formulasi.
  - Gauss eliminatsiyasi - bu algoritm, matrisani yechimlar va ustunlar bo'yicha birlashtiradi va determinantni hisoblaydi. U  $O(n^3)$  vaqt bilan ishlaydi,  $n$  - matrisa o'lchamini ifodalaydi.
  - Ko'paytirish formulasi - bu usul determinantni elementlar va yordam hisoblab topadi. U ham  $O(n^3)$  vaqt talab qiladi.

Matrisalarni ko'paytirish va determinantni hisoblash uchun algoritm ijrochilari samaradorligini o'lchashda muhimdir, chunki matrisalar bilan ishlash kompyuterlarning kuchli tomonlaridan biri hisoblanadi. Bu algoritm ijrochilar samaradorlikni o'lchashda va o'rganilayotgan matematik modellarini amalga oshirishda muhim rollarga ega.

### Matrisalarni ko'paytirish:

- Matrisa ko'paytirish - ikki matrisa ko'paytmasini hisoblash jarayoni.
  - Agar A matritsasi  $m \times n$  o'lchamli, B matritsasi  $n \times p$  o'lchamli bo'lsa, ularning ko'paytmasi C matritsasi  $m \times p$  o'lchamli bo'ladi.
1. **Determinantni hisoblash:**
- Determinant - kvadrat matritsaning skalyar qiymati.
  - $2 \times 2$  matritsaning determinanti:  
Copy  

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$
  - Larger matritsa uchun determinantni hisoblash:
    - Rekursiv algoritm:  
css  
Copy  

$$DET(A) =$$

```

IF A is 1x1 THEN
 return A[1,1]
ELSE
 sum = 0
 FOR i = 1 TO n
 B = A without i-th row and 1-st column
 sum = sum + (-1)^(i+1) * A[i,1] * DET(B)

```

```
END FOR
return sum
END IF
```

- Vaqt murakkabligi  $O(n!)$ , xotira murakkabligi  $O(n^2)$ .
- Determinantni hisoblash uchun boshqa optimalroq algoritmlar ham mavjud.

## 2. Algoritm ijrochisi:

- Algoritmning kodni yoki psevdokodini qabul qiladi.
- Algoritm bo'yicha berilgan ma'lumotlar (kirish qiymatlari)ni qabul qiladi.
- Algoritm bo'yicha hisoblash jarayonini bajaradi.
- Algoritm natijasini (chiqish qiymatlarini) qaytaradi.
- Algoritm ijrochisi kompyuter yoki boshqa raqamli qurilma bo'lishi mumkin.

Umumiy xulosalar:

- Matrisa ko'paytirish va determinant hisoblash murakkab amallar bo'lib, o'z-o'zidan optimallashtirish qiyin.
- Algoritm ijrochisi algoritmning to'g'ri bajarilishini ta'minlaydi va natijalarni qaytaradi.

## S207 Tarmoqlanuvchi algoritmlar. Algebraik va transendent tenglamalarni taqribiy yechish usullari

### Tarmoqlanuvchi algoritmlar:

- Tarmoqlanuvchi algoritmlar - bir amalni bajarish uchun tarmoqning bir nechta qadamli tartibini ishlatadigan algoritmlar.
- Har bir qadamda kelgan natija oldingi qadamlarda olingan natijalarga bog'liq bo'ladi.
- Tarmoqlanuvchi algoritmlar samarali va tez ishlaydigan algoritmlardir.
- Masalan, binar qidiruv, tarmoqli saralash, rekursiv algoritmlar tarmoqlanuvchi algoritmlarga misoldir.

### 2. Algebraik tenglamalarni taqribiy yechish usullari:

- Algebraik tenglamalarni taqribiy yechish usullari - xususiy tenglama yechimi uchun taxminiy qiymatlarni topish usullari.
- Ba'zi usullar:
  - **Birinchi darajadagi kesmalar:**
    - Tenglamani birinchi darajadagi kesmalar yordamida yechish.
    - Ba'zi algoritmik usullar bilan kesmalar topiladi va yechimlarni aniqlash uchun uning qiymatlari qo'llaniladi.
  - **Newton-Raphson usuli:**
    - Tayangan tomonidan ishlatiluvchi yuqori darajali tenglamalar uchun.
    - Tayangan qarshilik sirti va tenglamning bir nuqtadagi qarashlanish tezligi asosida yangi taxminiy yechimlar topiladi.
  - **Regula falsi usuli:**
    - Tenglamani kesmalar yordamida yechish algoritmi.
    - Kesmalar orasida tenglamani kesish nuqtasi topiladi.
  - **Iteratsion usullar:**
    - Tenglamani yangi qiyinlashgan tenglamaga o'tkazib borish.
    - Ba'zi algoritmik usullar, misol uchun iteratsion formulalar, qo'llaniladi.

### 3. Transendent tenglamalarni taqribiy yechish usullari:

- Transendent tenglamalarni taqribiy yechish usullari - transendent funksiyalarning yechimlarini aniqlash uchun taxminiy qiymatlarni topish usullari.
- Ba'zi usullar:
  - **Bolajak usul:**

- Tenglamani biror bo'lagichga bolib, tenglamning o'ziga qaraganda oddiy funksiyalarga aylantirib yechish.
- Bolajak usulning mos varianti shartli o'zgaruvchilar usuli.
- **Iteratsion usullar:**
  - Tenglamni yangi qiyinlashgan transcendent funksiyaga o'tkazib borish.
  - Ba'zi iteratsion formulalar, misol uchun Newton-Raphson usuli, qo'llaniladi.

Bu usullar transcendent va algebraik tenglamalarni taqribiy yechishda yordam beradi, lekin natijalar taxminiy bo'lishi mumkin. Yechimlarning aniqligini oshirish uchun qiymatlar diapazoni kengaytiriladi yoki boshqa usullar qo'llaniladi.

Tarmoqlanuvchi algoritmlar, ma'lumotlar oqimida axborotlar o'tkazishning qulay va samarali usullarini ifodalaydi. Bu algoritmlar, ba'zi ma'lumotlarni qo'llab-quvvatlash, filtratsiya qilish, qayta ishlash va boshqa maqsadlarni amalga oshirish uchun ishlatiladi. Algebraik va transcendent tenglamalarni taqribiy yechish usullari esa qo'shimcha ma'lumotlarni olish, kengaytirish yoki darrov yechimlarni topishda yordam beradi. Quyidagi, ularni qisqa muddatda ko'rib chiqamiz:

### 1. Tarmoqlanuvchi algoritmlar:

Tarmoqlanuvchi algoritmlar ma'lumotlarni tarmoqlar orqali o'zaro almashish uchun ishlatiladi. Ular quyidagi asosiy tarmoqlashuv usullarini qamrab oladi:

- **Arifmetik tarmoqlanish:** Ushbu usulda arifmetik amallar (qo'shish, ayirish, ko'paytirish, bo'lish) o'tkaziladi.
- **Logik tarmoqlanish:** Bu usulda boole algebra amallari (AND, OR, NOT, XOR) o'tkaziladi.
- **Turli xil funksiyalar:** Ma'lumotlar tarmoqlar orqali turli xil funksiyalarga murojaat qilish uchun yordam beriladi.
- **Tarmoq kirish-uchish:** Ushbu usulda tarmoqlar orqali ma'lumot oqimida to'liq kirish-uchish yordamida ma'lumot almashish amalga oshiriladi.

### 2. Algebraik va transcendent tenglamalarni taqribiy yechish usullari:

Algebraik va transcendent tenglamalarni taqribiy yechish uchun quyidagi asosiy usullar mavjud:

- **Arifmetik yechim usullari:** Ko'p darajali tenglamalarni yechish uchun, ma'lumotlar ustida odatda ko'p qatlamli arifmetik operatsiyalar (qo'shish, ayirish, ko'paytirish, bo'lish) amalga oshiriladi.



- **Nishonchilik usullari:** Bu usul algebraik yechimlarni aniqlash uchun iste'mol qilinadi. Bu usulda, ma'lumotlar ustida odatda soniy tahlillar yordamida yechimlar aniqlanadi.
- **Ko'paytirish va ichma-ich yechimlar:** Ushbu usulda, algebraik va transendent tenglamalarni ko'paytirish va ichma-ich yechimlar yordamida yechish amalga oshiriladi. Ko'paytirish usulida, matrisa algoritm va determinanti yordamida yechimlar topiladi. Ichma-ich yechimlar esa faqat to'g'ri yoki mos keladigan yechimni topish uchun yordam beradi.
- **Hulosa usullari:** Ushbu usulda, tenglamalar uchun hulosa usullari (Newton-Raphson, sekant, bisection, Muller, Brent va boshqalar) qo'llaniladi. Ular, bir nechta ishtirok etadigan usullardir, va har biri o'zining maqsad va mahsulotlariga ega.

Algebraik va transendent tenglamalarni taqribiy yechish usullari, kompyuter dasturlash va matematika sohasida ahamiyatga ega bo'lgan muhim texnologiyalardir. Ular ma'lumotlar analizi, axborotlar oqimlari, ilmiy tadqiqotlar, statistika, injinerlik va boshqa sohalarda samarali yechimlar topishda yordam beradi.

## S208 Algoritmlar samaradorligini baholash. Iteratsion sikllar

Algoritmning samaradorligini baholash, uning ishlash tezligi, xotira sarflanishi va tarmoqlanishlilik kabi ko'rsatuvlarga asoslanadi. Iteratsion sikllar esa algoritmning samaradorligini oshirish uchun juda foydali bo'lishi mumkin. Quyidagi ko'rsatuvlar iteratsion sikllar samaradorligini baholashda muhim bo'lgan asosiy qatorlar hisoblanadi:

1. **Ishonchli vaqt samaradorligi:** Iteratsion sikllar ko'p marta takrorlanishi mumkin bo'lgani uchun, ularning vaqt samaradorligi yuqori bo'lishi kerak. Agar sikl odatiy vazifalarni bajarishda ishona bo'lsa, bu samaradorlikni oshiradi.
2. **Hajm samaradorligi:** Sikl tashqi vaqt va xotira resurslarini iste'mol qiladi. Agar siklning har qanday o'tkazish operatsiyasi uchun xotira zaxiralanishi kerak bo'lsa, bu hajm samaradorligini pastaytiradi.
3. **Kod qiyinchiliklari:** Kodni tushunish, o'qish va yozish qulayliklari samaradorlikni oshiradi. Oddiy, tushunishga yoqadi va aniq bo'lishadi, bu esa samaradorligini oshiradi.
4. **Ko'p martalik operatsiyalar:** Siklning o'zida qisqa va oson operatsiyalarni o'z ichiga olgan holda takrorlanish mumkinligi samaradorlikni oshiradi. Agar bir marta ko'p ishlar olib bo'lsa, bu samaradorlikni pastaytiradi.
5. **Siklning tartibi:** Siklning takrorlanish tartibi ham samaradorlikni ta'sir qiladi. Agar ko'p marta qayta ishlash shartlari ajratilgan bo'lsa, bu samaradorlikni pastaytiradi.
6. **Ko'p martalik takrorlanishlar soni:** Agar bir iteratsion sikli odatda ko'p marta takrorlansa, ammo takrorlar soni biror chegaraga yetib kelgan bo'lsa, bu samaradorlikni ta'sir qiladi.

Iteratsion sikllar samaradorligini baholash uchun yuqoridagi qatorlarni hisobga olsak, quyidagi yechimlar paydo bo'ladi:

- **Oddiylik va osonlik:** Iteratsion siklning kodini o'qish va tushunish oson bo'lishi.
- **Vaqt samaradorligi:** Siklning har takrorlanish jarayoni uchun sarflangan vaqt minimal bo'lishi.
- **Xotira sarflanishi:** Siklning har bir takrorlanishi uchun sarflangan xotira miqdori minimal bo'lishi.
- **Ishonchli vaqt samaradorligi:** Siklning qandaydir o'zgaruvchilarga bog'liq qilib takrorlanishni to'xtatishga imkon berish.
- **Har qanday qiyinchiliklarning kamligi:** Siklda kod qiyinchiliklari yoki uzoq operatsiyalar past bo'lishi kerak.

- **Ko'p martalik takrorlanishlar soni:** Agar bir sikl ko'p marta takrorlansa, ammo shu vaqt esa bir chegaraga yetib kelgan bo'lsa, bu samaradorlikni oshiradi.

Bunday yechimlar yordamida iteratsion sikllar samaradorligini baholash oson bo'ladi va uning qanday darajada optimallashtirilishi kerakligini aniqlashga yordam beradi.

### 1. Algoritmlar samaradorligini baholash:

- Algoritmlar samaradorligini baholash uchun asosiy ikkita xarakteristika qo'llaniladi:
  - **Vaqt murakkabligi** - algoritm ishlatadigan vaqtni baholaydi.
  - **Xotira murakkabligi** - algoritm uchun zarur bo'lgan xotira hajmini baholaydi.
- Vaqt murakkabligini baholash uchun "Big O" notatsiyasi ( $O(f(n))$ ) qo'llaniladi. Bu algoritm ishlash vaqtining ma'lum bir o'sish tezlanishini ifodalaydi.
- Xotira murakkabligini baholash uchun ham "Big O" notatsiyasi qo'llaniladi, lekin bu algoritm uchun kerak bo'lgan xotira hajmi o'sishini ifodalaydi.
- Algoritmlarning samaradorligi qancha past "Big O" qiymatiga ega bo'lsa, ular shuncha samarali hisoblanadi.

### 2. Iteratsion sikllar:

- Iteratsion sikllar - algoritmda ma'lum bir qadamni qayta-qayta bajarishni ta'minlovchi konstruksiyalar.
- Asosiy turlari:
  - **For tsikli:**
    - Tsikl boshlang'ich qiymatdan nihoyatgi qiymatgacha ma'lum qadamda takrorlanadi.
    - Umumiy ko'rinishi: for i = start to end step step
  - **While tsikli:**
    - Tsikl ma'lum shart bajarilguncha davom etadi.
    - Umumiy ko'rinishi: while condition do
  - **Repeat-Until tsikli:**
    - Tsikl kamida bir marta bajariladi, so'ngra shart tekshiriladi.
    - Umumiy ko'rinishi: repeat until condition
- Tsikllar algoritmlar tarkibida asosiy element bo'lib, ularning noto'g'ri yoki samarali ishlatilishi algoritm samaradorligini sezilarli darajada o'zgartirishi mumkin.

Xulosa qilib aytganda, algoritmlar samaradorligini baholash uchun vaqt va xotira murakkabligi ko'rsatkichlari muhim bo'lib, iteratsion sikllar algoritmlar tuzilishida asosiy element hisoblanadi.

## S209 Algebraik va transcendent tenglamalarni Vatarlar, Nyuton usullari.

Algebraik va transcendent tenglamalarni Vatarlar (Bisection) va Nyuton (Newton) usullari orqali taqribiy yechish usullari juda samarali va kuchli usullardir. Ularning qo'llanish prinsiplari quyidagicha:

### 1. Vatarlar (Bisection) Usuli:

Vatarlar usuli, tenglamalarni yechish uchun juda oson va samarali usul hisoblanadi. U quyidagi tartibda ishlaydi:

- Boshlang'ich intervallarni tanlash: Berilgan intervall  $[a, b]$  ni tanlang.
- Interval o'rtasini hisoblash:  $c = \frac{a+b}{2}$   $c = \frac{a+b}{2}$  formulasi yordamida intervalning o'rtasini toping.
- Tenglama qiymatini hisoblash: Tenglama qiymatini  $f(c)$  hisoblash.
- Yangi intervalni tanlash: Agar  $f(a)f(c) < 0$  va  $f(c)f(b) > 0$  teng bo'lsa, yangi interval  $[a, c]$  ni tanlash; aks holda, yangi interval  $[c, b]$  ni tanlash.
- Qadamni takrorlash: Barcha qadamni takrorlash va tenglama qiymatini to'g'ri qiymatga yaqinlashish.

Bu algoritm usulida tenglama yechimi tez va ishonchli ko'rinishda topiladi. Shuningdek, bu usul o'nlikligi qadrlash va intervalning kengligi o'rtasida o'nlikligi oshirish yordamida to'g'ri yechimi aniqlash uchun mos ravishda ishlaydi.

### 2. Nyuton (Newton) Usuli:

Nyuton usuli, tangent chiziqli orqali tenglamalar yechimini topish uchun ishlatiladi. U quyidagi tartibda ishlaydi:

- Boshlang'ich taxminni tanlash: Berilgan boshlang'ich taxmini  $x_0$  tanlash.
- Yechimning taxminiy qiymatini topish:  $x_0$  da funksiyaning qiymatini  $f(x_0)$  toping.
- Tangent chiziqli orqali yangi taxminni topish:  $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$   $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$  formulasi yordamida yangi taxminni toping.
- Taxminni yakuniy taxmin bilan solishtirish: Taxminni o'zgaruvchilarning mosligini solishtirish. Agar taxminning aniqligi belgilangan chegaradan kam bo'lsa, yechim topildi deb hisoblanadi. Aks holda, yangi taxminni topish uchun qadamni takrorlash.

Nyuton usuli, boshlang'ich taxminni yaxshi tanlash va har bir taxminni yangilab borish orqali tez yechimni topishda juda samarali bo'ladi. Agar funksiya bilan uning

birinchi va ikkinchi turli turlari mosligi yaxshi bo'lsa, Nyuton usuli shighar bilan ishlaydi. Ammo, uning eng yaxshi ishchiligi taxminning yaqinligi shartiga bog'liqdir.

Bular ikkala usulga misol keltirish mumkin:

### Misol:

Berilgan funksiya  $f(x)=x^3-2x-5$  uchun Vatarlar va Nyuton usullarini ishlatib, tenglama yechimini toping.

### Vatarlar Usuli:

Boshlang'ich interval:  $[2, 3]$

Birinchi o'rtasining qiymati:  $c=\frac{2+3}{2}=2.5$

Birinchi o'rtasining qiymati:  $f(c)=(2.5)^3-2\cdot 2.5-5=-0.875$

Boshlang'ich interval  $[2, 2.5]$  deb o'zgartiramiz, chunki  $f(2)\cdot f(2.5)<0$

Keyingi o'rtasining qiymati:  $c=\frac{2+2.5}{2}=2.25$

Keyingi o'rtasining qiymati:  $f(c)=(2.25)^3-2\cdot 2.25-5=0.234375$

Keyingi o'rtasining qiymati:  $c=\frac{2+2.25}{2}=2.125$

Keyingi o'rtasining qiymati:  $f(c)=(2.125)^3-2\cdot 2.125-5=-0.328125$

Boshlang'ich interval  $[2.125, 2.25]$  deb o'zgartiramiz, chunki  $f(2.125)\cdot f(2.25)<0$

Yana bir necha qadamlarni takrorlaymiz va  $x\approx 2.094$  va  $f(x)\approx 0$  qo'llanishini aniqlab olamiz.

### Nyuton Usuli:

Boshlang'ich taxmin:  $x_0=2$

Boshlang'ich qiymat:  $f(x_0) = (2)^3 - 2 \cdot 2 - 5 = 1$   
 $f(x_0) = (2)^3 - 2 \cdot 2 - 5 = 1$

Boshlang'ich chiziqining taxminiy qiymati:  $x_1 = 2 - \frac{1}{3} \cdot (2)^2 - 2 = 53 \approx 1.667$   
 $x_1 = 2 - \frac{1}{3} \cdot (2)^2 - 2 = 53 \approx 1.667$

Yangi qiymat:  $f(x_1) = (53)^3 - 2 \cdot 53 - 5 \approx -0.704$   
 $f(x_1) = (53)^3 - 2 \cdot 53 - 5 \approx -0.704$

Yangi chiziqining taxminiy qiymati:  $x_2 = 53 - 0.704 \cdot \frac{1}{3} \cdot (53)^2 - 2 \approx 2.079$   
 $x_2 = 53 - 0.704 \cdot \frac{1}{3} \cdot (53)^2 - 2 \approx 2.079$

Yangi qiymat:  $f(x_2) \approx 0.014$   
 $f(x_2) \approx 0.014$

Yana bir necha qadamlarni takrorlaymiz va  $x \approx 2.094$   
 $x \approx 2.094$  va  $f(x) \approx 0$  qo'llanishini aniqlab olamiz.

Shunday qilib, Vatarlar va Nyuton usullari yordamida tenglamalar yechimi taqribiy qiymatlarga tez aniqlanadi.

### 1. Algebraik tenglamalarni yechish usullari:

#### a) Biseksiya (Vatarlar) usuli:

- Algebraik tenglamani ikkita qiymat orasidagi kesmani davriy ravishda kichraytirib borish yordamida yechish usuli.
- Algoritm quyidagi asosiy bosqichlardan iborat:
  - Tenglamaning ildizlari joylashgan oraliqni aniqlash.
  - Oraliq uchun boshlang'ich va oxirgi qiymatlarni tanlash.
  - Oraliqning o'rtasidagi nuqtani hisoblash.
  - Yangi qiymat oraliqdagi ildizni o'z ichiga olishini tekshirish.
  - Agar yangi qiymat ildizni o'z ichiga olsa, u holda yangi boshlang'ich yoki oxirgi qiymat sifatida qabul qilinadi.
  - Agar kerakli aniqlik yoki shart bajarilmasa, 3-5 qadam takrorlanadi.
- Usul sekin, lekin yaxshi ishlatiladi, agar ildizlar yaxshi aniqlanib olsa.

#### b) Newton-Raphson usuli:

- Algebraik tenglamani yechish uchun iteratsion usul.
- Algoritm quyidagi asosiy bosqichlardan iborat:
  - Boshlang'ich taxminiy yechim  $x_0$  ni tanlab olish.
  - Quyidagi iteratsion formuladan foydalanib, yangi yechim  $x_1$  ni hisoblash:  
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

3. Agar  $|x_1 - x_0| < \epsilon$  (aniqlik shartini) bajarilmasa, 2-qadam takrorlanadi.

- Usul tezkor, lekin boshlang'ich taxmin aniq bo'lishini talab qiladi.

2. **Transendent tenglamalarni yechish usullari:**

a) **Biseksiya (Vatarlar) usuli:**

- Transendent tenglamani ikkita qiymat orasidagi kesmani davriy ravishda kichraytirib borish yordamida yechish usuli.
- Algoritmi algebraik tenglamalar uchungisi bilan o'xshash.

b) **Newton-Raphson usuli:**

- Transendent tenglamani yechish uchun iteratsion usul.
- Algoritmi algebraik tenglamalar uchungisi bilan o'xshash, faqat hisoblash jarayonida transendent funksiya va uning hosilasi ishlatiladi.

Umumiy xulosaga kelganda, Vatarlar usuli barcha tenglamalar uchun ishlaydi, lekin sekin, Newton-Raphson usuli esa tezkor, ammo boshlang'ich taxmin aniq bo'lishini talab qiladi. Tenglamaning xususiyatlariga qarab, ushbu usullardan biri yoki boshqa usullar qo'llanilishi mumkin.

## S210.Oddiy iteratsiya algoritmlari

Oddiy iteratsiya algoritmlari - bu qator takrorlanuvchi (iteratsiya) operatsiyalar orqali muammoni yechish uchun ishlatiladi. Bu algoritmlar boshdan oxirigacha ketma-ket bajariladi, har bir qadam oldingi qadam natijasiga asoslanadi. Ular quyidagi xususiyatlarga ega:

1. **Ketma-ketlik:** Algoritmlar bosqichma-bosqich va navbat bilan bajariladi. Har bir qadam oldingi qadam natijalari asosida amalga oshiriladi.
2. **Takrorlanish:** Muammoni yechish uchun bir necha marta takrorlanuvchi qadamlar bajariladi. Qadamlar soni muammoning xususiyatlariga bog'liq bo'ladi.
3. **Boshlang'ich qiymatlar:** Algoritm boshlanganda muammoning ma'lum boshlang'ich qiymatlari beriladi. Keyingi qadam natijasi ushbu qiymatlar asosida hisoblanadi.
4. **Yaqinlashuv:** Algoritm boshlangich qiymatlardan boshlab yechimga yaqinlashib boradi. Yetarli qadar takrorlanganidan so'ng, yechim hisoblangan natijaga yaqinlashadi.

Oddiy iteratsiya algoritmlariga quyidagilar misol bo'ladi:

1. **Ikkitalik qidiruv (Binary Search):**
  - Tartiblanmagan massivda ma'lum element mavjudligini tekshirish.
  - Har bir qadam massivning yarmini tekshirib, qidiruvni qisqartirib borish.
2. **Newton-Raphson usuliga ko'ra ildizlarni topish:**
  - Funksiya ildizlarini topish uchun iterativ qiymatlarni hisoblab borish.
  - Har bir qadam funksiya qiymatini va uning birinchi hosilasini hisoblab, yangi taxmin qiymatni topadi.
3. **Sanoq usulidan foydalangan holda ildizlar topish:**
  - Funksiya ildizlarini taxminiy topish uchun intervalni kesib borish.
  - Har bir qadam interval uzunligini kamaytirib, ildizni aniqroq topadi.
4. **Iterativ hall qilish:**
  - Chiziqli va chiziqli bo'lmagan tenglamalar tizimini yechish.
  - Har bir qadam oldingi yechimlarni qayta ishlab, yangi yechimni hisoblaydi.

Oddiy iteratsiya algoritmlari sodda bo'lsa-da, murakkab muammolarni yechishda muhim rol o'ynaydi. Ular yechimga yaqinlashuv jarayonini ta'minlaydi va boshqarilishi oson. Ammo murakkab muammolar uchun yetarli emasligi mumkin, bu holda boshqa, samaraliroq algoritmlardan foydalanish kerak bo'ladi.



## **S211 Chiziqli dasturlash masalalarining matematik modellari, iqtisodiy tahlili.**

Chiziqli dasturlash masalalarining matematik modellari iqtisodiy tahlil uchun keng qo'llaniladi. Bu modellarga ko'plab turli iqtisodiy tahlillar (masalan, istiqbol va iste'mol, sarmoya qo'llanilishi, narxlarning tahlili va boshqalar) uchun foydalaniladi. Quyidagi matematik modellarga misollar keltirib, ularga iqtisodiy tahlil qilishni ko'ramiz:

### **1. Istiqlol va iste'mol (Supply and Demand):**

Bu modell, mahsulotlar istiqloli (qanchalik mahsulot iste'foda etiladi) va iste'mol (qanchalik mahsulot ishlatiladi) o'rtasidagi munosabatlarni modellelashda foydalaniladi. Istiqlol va iste'mol funksiyalari odatda narxga bog'liq hisoblanadi.

#### **Matematik modellari:**

- Istiqlol funksiyasi:  $Q_s = f(P)$
- Iste'mol funksiyasi:  $Q_d = g(P)$

Bu yerda  $Q_s$  mahsulot istiqloli,  $Q_d$  mahsulot iste'moli,  $P$  esa mahsulot narxi. Iqtisodiy tahlil uchun, istiqloli va iste'mol funksiyalari odatda kvadratik, lineyka yoki logarifmik formulalar yordamida ifodalangan.

### **2. Narxlarning tahlili (Price Analysis):**

Bu modell, mahsulotlar va xizmatlar uchun belgilangan narxlarni o'rganishda qo'llaniladi. Bu modell istiqloli va iste'molning o'rtasidagi munosabatlarni o'rganishda asosiy qo'llaniladi.

#### **Matematik modellari:**

- Narxning o'zgarishi:  $P = f(Q)$

Bu yerda  $P$  narx,  $Q$  esa istiqloli yoki iste'mol miqdori.

### **3. Sarmoya qo'llanilishi (Investment Analysis):**

Bu modell, sarmoya qo'llanilishining moliyaviy samaradorligi va xavfsizlikni baholashda qo'llaniladi. Bu modellga chiziqli dasturlash yordamida sarmoya qo'llanishning modellarini tuzish mumkin.

#### **Matematik modellari:**

- Moliyaviy samaradorlik (ROI):  $ROI = \frac{P_f - P_i}{P_i} \times 100$

Bu yerda  $P_f$  sarmoya tug'ilgan darajadagi qiymati,  $P_i$  esa sarmoya boshlang'ich qiymati. ROI (sarmoya darajasi) ko'p vaqt vaqtiiy, dastlabki sarmoya (initial investment) miqdorini hisoblashda ishlatiladi.

#### 4. O'ziga xoslik (Uniqueness):

Bu modell, mahsulotlar yoki xizmatlar uchun xaridorni qanday sifatida xoslashga qaratilgan uchun xaridorni qanday sifatida qanday qo'llanishga ta'sir ko'rsatishda qo'llaniladi.

#### Matematik modellari:

- Xaridor xosligi indeksi:  $UI = \frac{\text{Foydalanilgan qo'llanish}}{\text{Barcha qo'llanishlar}}$

Bu yerda "UI" xaridor xosligi indeksi, "Foydalanilgan qo'llanish" esa xaridorni o'zga xos qiluvchi xususiyatni ifodalovchi qiymat, "Barcha qo'llanishlar" esa umumiy xaridorni keng xil qo'llanishlar indeksi.

#### 5. Ma'lumotlar tizimi va texnologik innovatsiyalar (Information Systems and Technological Innovations):

Bu modell, ma'lumotlar tizimi va texnologik innovatsiyalar o'rtasidagi munosabatlarni ifodalaydi. Misol uchun, yangi texnologik innovatsiyalar kiritilganda, istiqolol va iste'mol darajasi o'zgarishi modeli.

#### Matematik modellari:

- Yangi innovatsiyalar kiritilganda istiqolol va iste'mol funksiyasi:  
 $Q_{s/new} = f(P)$

Bu yerda  $Q_{s/new}$  yangi innovatsiyalar kiritilganda mahsulot istiqololi.

Bu matematik modellarning o'rniga, Python dasturlash tili yordamida taqribiy iqtisodiy tahlilni amalga oshirish mumkin

### 3.5

Chiziqli dasturlash masalalarining matematik modellari va iqtisodiy tahlili:

### 1. Matematik model:

Chiziqli dasturlash masalasi quyidagi umumiy ko'rinishda ifodalanadi:

Maqsad funksiyasi:  $\max$  (yoki  $\min$ )  $z = c^T * x$

Cheklovlar:

$$A * x \leq b$$

$$x \geq 0$$

Bu yerda:

- $x$  - yechim vektori ( $n$  o'lchamli)
- $c$  - maqsad funksiyasi koeffitsientlari vektori ( $n$  o'lchamli)
- $A$  - cheklov tenglamalar matrisi ( $m \times n$  o'lchamli)
- $b$  - cheklov tenglamalar oq'izmalar vektori ( $m$  o'lchamli)

### 2. Iqtisodiy tahlil:

Chiziqli dasturlash masalasining iqtisodiy ma'nosi quyidagicha:

- Maqsad funksiyasi - foyda, daromad, ishlab chiqarish hajmi kabi miqdorni maksimallashtirish yoki xarajatlarni, ittifoqni, ishsizlikni minimallashtirish.
- Cheklovlar - resurslarning cheklanganligi (xomashyo, ish kuchi, moddiy-texnik vositalar va h.k.), texnologik shartlar, marketing talablari va boshqa cheklovlar.
- Yechim vektori - ishlab chiqarish hajmlari, sotish miqdorlari, resurslardan foydalanish darajalari va h.k.

### 3. Iqtisodiy ma'no va amaliy qo'llanilishi:

- Resurslardan samarali foydalanish
- Mahsulotlar yoki xizmatlarning optimal nisbatlarini aniqlash
- Ishlab chiqarish rejalarini optimallashtirish
- Logistika va taqsimot masalalarini yechish
- Portfel menejmenti va investitsiya qarorlarini qabul qilish
- Moliyaviy rejalashtirish va byudjetlash
- Marketing va sotuvni muvofiqlashtirish

Chiziqli dasturlash modellarining iqtisodiy mohiyati shundaki, ular iqtisodiy sub'ektlar (kompaniyalar, davlatlar, xo'jaliklar va h.k.) oldida turgan maqsad va cheklovlarni matematik tarzda ifoda etish va yechish imkonini beradi. Bu esa samarali qarorlar qabul qilish va resurslardan oqilona foydalanishga imkon yaratadi.

## S212 Chiziqli dasturlash masalalarining maqsad funksiyasi.Egizak masala.

Chiziqli dasturlash masalalarida maqsad funksiyasi, maksimallashtirilayotgan yoki minimallashtirilayotgan miqdorni ifodalaydi. Egizak masalasida misol ko'ramiz: Misol: Bir kompaniya egizaklarni ishlab chiqarish uchun xomashyo va ishchi resurslardan foydalanishi kerak. Kompaniya 100 soatlik xomashyo va 80 soatlik ishchi vaqti bo'lsa, egizak tayyorlash va ishlab chiqarish uchun har bir egizak uchun 2 soat xomashyo va 4 soat ishchi vaqti kerak. Egizakni sotish narxi esa 4000 so'm bo'lsa, kompaniya qancha egizak ishlab chiqarish orqali maksimal daromad olishishi mumkin?

Maqsad funksiyasi: Daromadni maksimallashtirish

Yechim o'zgaruvchilari:

x: Ishlab chiqariladigan egizaklar soni

Cheklovlar:

- Xomashyo cheklovi:  $2x \leq 100$
- Ishchi vaqti cheklovi:  $4x \leq 80$

Maqsad funksiyasini tushunish uchun daromadning ifodalanishini ko'ramiz. Har bir egizakning sotish narxi 4000 so'm bo'lsa, egizaklar soni bilan ko'paytiriladi va daromad sifatida ifodalashadi.

Shuning uchun maqsad funksiyasi:

$$z = 4000x$$

Chiziqli dasturlash masalasini yechish uchun cheklovlar bilan birga yuqoridagi maqsad funksiyasini maksimallashtirish kerak. Bu masalani shunday yechish mumkin:

Chiziqli dasturlash masalasini yechish:

Maksimallashtirilayotgan maqsad funksiyasi:  $z = 4000x$

Cheklovlar:

- $2x \leq 100$
- $4x \leq 80$
- $x \geq 0$

Bu masalani yechish uchun chiziqli dasturlash algoritmlaridan, masalan, Simpleks usulidan foydalanish mumkin. Lekin masalaning oddiyligi sababli, uni osonlik bilan ham yechish mumkin. Cheklovlar orqali egizaklar sonini chegaralashimiz va shu chegaralanayotgan qiymatlarni maqsad funksiyasiga kiritib, maksimal daromadni topamiz.

Masalning yechimi:

1. Xomashyo cheklovini hisoblash:  
 $2x \leq 100$   
 $x \leq 50$  (x ni cheklovga ko'paytirib, tenglik belgisini o'ngga o'tkazamiz)
2. Ishchi vaqti cheklovini hisoblash:  
 $4x \leq 80$   
 $x \leq 20$

3. Egizaklar sonini chegaralash:

$x \leq 20$  ( $x$  ni cheklovga ko'paytirib, tenglik belgisini o'ngga o'tkazamiz)

4. Maqsad funksiyasini hisoblash:

$$z = 4000x$$

$$z(\max) = 4000 * 20 = 80,000$$

Shu jihatdan, kompaniya 20 ta egizak ishlab chiqarish orqali maksimal 80,000 so'm daromad olishi mumkin.

Chiziqli dasturlash masalalarida maqsad funksiyasi, masalani yechishni maqsad qilgan funksiya, ya'ni, muayyan shartlarga muvofiq eng yaxshi (ya'ni, maksimum yoki minimum) natijani topish uchun ishlatiladi.

Egizak masala (unimodal problem), masalani yechish uchun yagona maksimum yoki minimum nuqtani qidirmoqda. Bundan tashqari, egizak masalalarda, funksiyaning yalpi kengligi ichida birinchi turli (maksimum yoki minimum) nuqta mavjud bo'ladi.

Masalan, biz bir funksiyaning maksimum nuqtasini topishni maqsad qilamiz. Ushbu nuqta quyidagi formuladan foydalanib topiladi:

$$x^* = \arg \max_{x \in [a, b]} f(x) \quad x^* = \arg \max_{x \in [a, b]} f(x)$$

Bu yerda  $f(x)$  funksiya,  $x^*$  maksimum nuqta, va  $[a, b]$  masala chegaralari.

Quyidagi misolga e'tibor bering:

### Misol:

Berilgan funksiya  $f(x) = x^2 - 4x + 4$  uchun maksimum nuqtani toping.

### Yechim:

1. Maqsad funksiyasi:

$$f(x) = x^2 - 4x + 4$$

2. Masala chegaralari:  $x \in [0, 4]$

Shunday qilib, funksiyaning o'zgaruvchilarning kengligi orqali mos holda taqribiy hisoblash uchun chiziqli dasturlash (masalan, Nyuton-Rafson usuli) qo'llaniladi.

Chiziqli dasturlash masalalaridagi maqsad funksiyasi, masalani yechish uchun maqsadlangan funksiya, ya'ni, muayyan shartlarga muvofiq eng yaxshi (ya'ni, maksimum yoki minimum) natijani topishda yordam beradi.

Egizak masalalar (unimodal problems) odatda bir maksimum yoki minimum nuqta mavjud bo'lgan masalalardir. Ular masala chiziqli funksiyalari orqali ko'rsatiladi. Maqsad funksiyasi, masalani yechishdagi maqsadni aniqlaydi.

Masalan, biz bir funksiyaning maksimum nuqtasini topishni maqsad qilamiz. Ushbu nuqta quyidagi formuladan foydalanib topiladi:

$$x^* = \arg \max_{x \in [a, b]} f(x) \quad x^* = \arg \max_{x \in [a, b]} f(x)$$

Bu formulada  $f(x)$  funksiya,  $x^*$  maksimum nuqta, va  $[a, b]$  masala chegaralari.

Quyidagi misolga e'tibor bering:

### **Misol:**

Berilgan funksiya  $f(x) = x^2 - 4x + 4$  uchun maksimum nuqtani toping.

### **Yechim:**

1. Maqsad funksiyasi:

$$f(x) = x^2 - 4x + 4$$

2. Masala chegaralari:  $x \in [0, 4]$

Shunday qilib, funksiyaning o'zgaruvchilarning kengligi orqali mos holda taqribiy hisoblash uchun chiziqli dasturlash (masalan, Nyuton-Rafson usuli) qo'llaniladi.

## S213 Chiziqli dasturlash masalalarini yechishning simpleks usuli.

Chiziqli dasturlash masalalarini yechishning Simpleks usuli quyidagi bosqichlardan iborat:

**1. Masalani standart ko'rinishga keltirish:**

- Maqsad funksiyasini maksimallashtirish ko'rinishiga keltirish
- Barcha cheklovlarni " $\leq$ " ko'rinishiga keltirish
- Xususiy o'zgaruvchilarni kiritish, agar cheklovlar " $\geq$ " yoki " $=$ " ko'rinishida bo'lsa

**2. Boshlang'ich asosiy echimni topish:**

- Cheklovlarni qoniqtiruvchi boshlang'ich asosiy echimni topish (masalan, grafik yoki sun'iy o'zgaruvchilardan foydalanish orqali)

**3. Simpleks jadvalini tuzish:**

- Maqsad funksiyasi va cheklovlarni simpleks jadvalida ifodalash
- Boshlang'ich asosiy echimni jadvalga yozish

**4. Simpleks algoritmi:**

- Tekshirish: Maqsad funksiyasi koeffitsientlari orasida musbatlari mavjudmi?
- Agar mavjud bo'lsa, eng katta musbat koeffitsientga mos o'zgaruvchini asosiy o'zgaruvchi sifatida tanlash
- Cheklovlar orasidan eng kichik nisbiy qiymat beradigan (eng chegaralovchi) cheklovni tanlash
- Yangi asosiy o'zgaruvchini va yangi asosiy echimni hisoblash
- Yangi simpleks jadvalini tuzish

**5. Takrorlash:**

- 4-bosqichni maqsad funksiyasi koeffitsientlari orasida musbatlari qolmaguncha davom ettirish

**6. Yechimni talqin qilish:**

- Topilgan yechim (asosiy echim) orqali maqsad funksiyasi qiymati va u ga mos o'zgaruvchilarning qiymatlarini aniqlash

Simpleks usulida har bir qadam yangi yaxshiroq echimni topishga qaratilgan bo'ladi. Bunda maqsad funksiyasini maksimallashtirish uchun, cheklovlarni qondiruvchi, lekin maqsad funksiyasini maksimal qiymatga yetkazuvchi echim topiladi. Usulning afzalliklari orasida quyidagilarni aytish mumkin:

- Chiziqli dasturlash masalalarini yechishda samarali
- Masalalar o'lcham va murakkabligidan qat'iy nazar, ishlash mumkin
- Cheklovlar soni va tuzilishi o'zgarsa ham, usulni qo'llay olish

Shuning uchun Simpleks usuli chiziqli dasturlash masalalarini hal qilishda eng keng tarqalgan va samarali usullardan biri hisoblanadi.

Chiziqli dasturlash masalalarini yechishning simpleks usuli (Simplex method) o'rtacha o'zgaruvchilar soni va qiymatlari ko'paytirib, optimal yechimni topishda ishlatiladi. Ushbu usul yengil, amalga oshirish bo'yicha moslashtirilgan va chiziqli

dasturlash masalalarining yechimlarini topish uchun o'rtacha, ortiqcha va buzuvchi o'zgaruvchilarni qo'llab-quvvatlaydi.

Simpleks usulni tushunish uchun quyidagi formulani ko'rib chiqamiz:

### **Masala shakli:**

Chiziqli dasturlash masalasi o'zgaruvchilarning lineyka funksiyasi (lineyka o'zgaruvchilar) shaklida ifodalangan bo'lsa, masala quyidagi ko'rinishda bo'ladi:

$$\max_{f_0} f(x) = c^T x \quad \text{shartlar: } \begin{cases} Ax \leq b \\ x \geq 0 \end{cases}$$

Bu yerda:

- $x$  -  $n$  ta o'zgaruvchi (variabllar) vektori.
- $c$  -  $n$  ta o'zgaruvchi uchun funksiyaning koeffitsientlari.
- $A$  -  $m \times n$  o'lchamdagi matrix, shartlar ning koeffitsientlari.
- $b$  -  $m$  ta o'zgaruvchi uchun shartlar.

Simpleks usuli quyidagi bosqichlardan iborat:

1. **Boshlang'ich yutuqlar** - Boshlang'ich yutuqlar (vertices) ko'rsatkichlar to'plami bilan aniqlovchi orta nuqta yoki boshlang'ich burchaklar to'plamini topish.
2. **Yuqori yutuqlarni topish** - Funksiya qiymatlarini baholash va yangi yutuqlarni aniqlash.
3. **Yangi yutuqlar** - Keyingi yutuqni topish uchun o'zgaruvchilarning qiymatlarini yangilash.
4. **Optimal yutuq** - Harakat qadamini takrorlash va optimal yutuqni aniqlash.



## S214 Chiziqli dasturlash masalalarini yechishning simpleks algoritmi va uni tahlil qilish

Chiziqli dasturlash masalalarini yechishning Simpleks algoritmi quyidagi bosqichlardan iborat:

1. **Masalani standart ko'rinishga keltirish:**

- Maqsad funksiyasini maksimallashtirish ko'rinishiga keltirish
- Barcha cheklovlarni " $\leq$ " ko'rinishiga keltirish
- Xususiy o'zgaruvchilarni kiritish, agar cheklovlar " $\geq$ " yoki " $=$ " ko'rinishida bo'lsa

2. **Boshlang'ich asosiy echimni topish:**

- Cheklovlarni qoniqtiruvchi boshlang'ich asosiy echimni topish (masalan, grafik yoki sun'iy o'zgaruvchilardan foydalanish orqali)

3. **Simpleks jadvalini tuzish:**

- Maqsad funksiyasi va cheklovlarni simpleks jadvalida ifodalash
- Boshlang'ich asosiy echimni jadvalga yozish

4. **Simpleks algoritmi:**

- Tekshirish: Maqsad funksiyasi koeffitsientlari orasida musbatlari mavjudmi?
- Agar mavjud bo'lsa, eng katta musbat koeffitsientga mos o'zgaruvchini asosiy o'zgaruvchi sifatida tanlash
- Cheklovlar orasidan eng kichik nisbiy qiymat beradigan (eng chegaralovchi) cheklovni tanlash
- Yangi asosiy o'zgaruvchini va yangi asosiy echimni hisoblash
- Yangi simpleks jadvalini tuzish

5. **Takrorlash:**

- 4-bosqichni maqsad funksiyasi koeffitsientlari orasida musbatlari qolmaguncha davom ettirish

6. **Yechimni talqin qilish:**

- Topilgan yechim (asosiy echim) orqali maqsad funksiyasi qiymati va u ga mos o'zgaruvchilarning qiymatlarini aniqlash

Simpleks algoritmi yordamida chiziqli dasturlash masalalarini yechishning tahlili:

1. **Asosiy yechimga etish:** Simpleks algoritmi boshlang'ich asosiy echimdan boshlanadi va har bir qadam yangi, yaxshiroq echim topishga qaratilgan. Bu bosqichda algoritm maqsad funksiyasini maksimallashtirish va cheklovlarni qondirish uchun eng yaxshi yechimni topishga xizmat qiladi.
2. **Algoritm samaradorligi:** Simpleks algoritmi o'zining matematik asoslanganlik va qo'llashning osonligi tufayli juda samarali hisoblanadi. Hatto murakkab masalalarni ham tezkor va aniq yechish imkonini beradi.
3. **Algoritmning cheklovlari:** Simpleks algoritmi faqat chiziqli dasturlash masalalarini yecha oladi. Agar masala nochiziqli bo'lsa, boshqa usullar qo'llanilishi kerak. Shuningdek, algoritm yordamida erishiladigan yechimlar faqat tashqi nuqtalarga mos keladi, ichki nuqtalar hisobga olinmaydi.

4. **Ko'p o'lchamli masalalarni yechish:** Simpleks algoritmi ko'p o'lchamli masalalarni ham yecha oladi. Masalaning o'lchami kattaygan sari algoritm biraz murakkablashadi, lekin yechish imkoni saqlanadi.

Umuman olganda, Simpleks algoritmi chiziqli dasturlash masalalarini yechishda eng samarali va keng tarqalgan usul hisoblanadi. U o'z matematikaviy asoslanganligini, universal qo'llanilishini va tezkor hamda aniq yechimlar berishini namoyon etadi.

Simpleks algoritmi, chiziqli dasturlash masalalarini yechish uchun ishlatiladi. Ushbu algoritm maqsad funksiyasi (objective function) bilan chegaralarga tegishli chiziqli funktsiyani bajarib chiqish uchun yengil yechimlarni topishda ishlatiladi. Keyingi qadamda, topilgan yengil yechimlarning optimal yechimni topish uchun tekshiriladi.

Simpleks algoritmi quyidagi bosqichlardan iborat:

1. **Boshlang'ich yutuqlarni aniqlash:** Boshlang'ich yutuqlar (vertices) ko'rsatkichlar to'plami bilan aniqlovchi orta nuqta yoki boshlang'ich burchaklar to'plamini topish.
2. **Yuqori yutuqlarni aniqlash:** Funksiya qiymatlarini baholash va yangi yutuqlarni aniqlash.
3. **Yangi yutuqlarni hisoblash:** Keyingi yutuqni topish uchun o'zgaruvchilarning qiymatlarini yangilash.
4. **Optimal yutuqni aniqlash:** Harakat qadamini takrorlash va optimal yutuqni aniqlash.

## **S215 Simpleks usulida topilgan yechimning iqtisodiy tahlili. Xulosa va takliflar**

Simpleks usulida topilgan yechimning iqtisodiy tahlili quyidagilarni o'z ichiga oladi:

### **1. Maqsad funksiyasi qiymati tahlili:**

- Topilgan yechim orqali maqsad funksiyasi qiymatini aniqlash
- Bu qiymat iqtisodiy jihatdan qanday ma'no kasb etishi (foyda, daromad, sarflar va h.k.)
- Agar maqsad funksiyasi minimallashtirilsa, unda topilgan qiymat xarajatlar/sarflarni aks ettiradi

### **2. Asosiy o'zgaruvchilar tahlili:**

- Topilgan yechimda asosiy o'zgaruvchilarning qiymatlari
- Bu qiymatlarning iqtisodiy ma'nosi (ishlab chiqarish hajmlari, resurslardan foydalanish darajalari va h.k.)
- Agar ba'zi o'zgaruvchilarning qiymatlari nolga teng bo'lsa, bu qanday ma'noga ega

### **3. Chegaralovchi o'zgaruvchilar tahlili:**

- Simpleks jadvalida qaysi o'zgaruvchilar chegaralovchi bo'lganini aniqlash
- Bu o'zgaruvchilarning iqtisodiy ma'nosi (cheklovlar, resurs cheklovlari va h.k.)
- Chegaralovchi o'zgaruvchilarning qiymatlari iqtisodiy jarayonlar uchun muhim

### **4. Ikkilamchi o'zgaruvchilar tahlili:**

- Simpleks jadvalida ikkilamchi o'zgaruvchilarning qiymatlari
- Bu qiymatlarning iqtisodiy ma'nosi (resurslar uchun qo'shimcha talab, dastlabki cheklovlarni qattiqlashtirish imkoniyatlari va h.k.)

Xulosa va takliflar:

- Simpleks usulida topilgan yechim iqtisodiy jarayon va parametrlarni chuqur tushunish imkonini beradi
- Maqsad funksiyasi qiymati, asosiy va chegaralovchi o'zgaruvchilar qiymatlarini tahlil qilish orqali optimal iqtisodiy qarorlar qabul qilish mumkin
- Ikkilamchi o'zgaruvchilar tahlili qo'shimcha imkoniyatlar va resurs talablarini aniqlash uchun foydali
- Tahlil natijalarini inobatga olib, jarayonlarni optimallashtirish, resurslardan samarali foydalanish, xarajatlarni kamaytirish va foydani oshirish bo'yicha takliflar ishlab chiqish mumkin.

Shunday qilib, Simpleks usulida topilgan yechimning iqtisodiy tahlili, kompaniya va korxonalar uchun qimmatli ma'lumotlar olish imkonini beradi.

Simpleks usulini chiziqli dasturlash masalalarini yechishda ishlatiladi. Ushbu usul optimal yechimni topishni ta'minlaydi va undan foydalanish iqtisodiy tahliliga imkoniyat beradi. Quyidagi misolda Simplex usuli orqali topilgan yechimning iqtisodiy tahlili va takliflar berilgan:

## Masala:

Maximize:  $Z=3x_1+2x_2$

Shartlar:

$2x_1 + x_2 \leq 4$   $x_1 + 3x_2 \leq 6$   $3x_1 + x_2 \leq 5$   $x_1, x_2 \geq 0$

Yechim: Chiziqli dasturlash masalalarini simpleks usuli orqali yechishda topilgan yechim:  $(x_1 = 1, x_2 = 1)$ , va  $(Z = 5)$ . Iqtisodiy Tahlil: 1. Maximize maqsadining aniq tushunchasi: Bizning maqsadimiz  $(Z)$  ni maksimum qilishdir. Masalada berilgan funksiya  $(Z = 3x_1 + 2x_2)$  ni maksimum qilish uchun biz  $(x_1)$  va  $(x_2)$  o'zgaruvchilarini aniqlashimiz kerak. 2. Masalani hal qilish va chegaralarni tushunish: Masala, berilgan shartlarga muvofiq,  $(x_1)$  va  $(x_2)$  ni burchak chizmasi orqali ta'riflab beradi. Biz  $(x_1)$  va  $(x_2)$  ni burchak chizmasida,  $(x_1)$  o'qidan  $(x_2)$  o'qiga, shuningdek,  $(x_1)$  o'qisida yuqori chizmaga chizimlab chizamiz. 3. Yechimning natijalari: Simplex usuli orqali topilgan yechim:  $(x_1 = 1, x_2 = 1)$ , va  $(Z = 5)$ . 4. Yechimning tushunchalari: Yechimning  $(x_1 = 1, x_2 = 1)$  va  $(Z = 5)$  ni topish demak,  $(Z)$  funksiyasi uchun maksimum yechimni ta'minlaydi. Bu yerda  $(x_1)$  va  $(x_2)$  o'zgaruvchilari optimal qarorlarni bildiradi. Xulosa va Takliflar: - Masalani bajarish jarayonida, berilgan maqsadning aniqlik bilan tushuntirilganligi va uning muhimligi oldindan bilib olingan. Bu masala yechishda, o'zgaruvchilar va maqsad funksiyasi aniq belgilangan va masala chegaralarini sodda yechish usulini qo'llab-quvvatlaydi. - Simplex usulini ishlatish natijalarini tahlil qilishda muhim bo'lgan nuqtalardan biri yechimning tushunchalari. Ushbu masalada topilgan yechim  $(x_1 = 1, x_2 = 1)$  va  $(Z = 5)$  ni ta'minlaydi. Bu, masalani bajarish jarayonida optimal yechimni topishni aniqlaydi. - Masala yechishda Simplex usulining xavfsiz va ishonchli yechimlarni ta'minlashda qanday ishlatilganligini tushunish ham muhimdir. - Qanday qilib masala yechishda Simplex usulidan foydalanishni ko'proq amaliy misollar orqali o'rganish tavsiya etiladi. Bu masalani yechishda, masalaning ko'priki yechimlarini topish uchun Simplex usulini ishlatish muhim bo'lishi mumkin. Masalani bajarishda qo'llaniladigan usul va algoritmlarni o'rganish va tushuntirish ishlab chiqarilgan yechimlar tahlilidan foydalanishga yordam beradi.

## Xulosa va Takliflar:

- **Xulosa:** Masalani bajarish jarayonida, berilgan maqsadning aniqlik bilan tushuntirilganligi va uning muhimligi oldindan bilib olingan. Bu masala yechishda, o'zgaruvchilar va maqsad funksiyasi aniq belgilangan va masala chegaralarini sodda yechish usulini qo'llab-quvvatlaydi.
- **Simplex usulini ishlatish natijalarini tahlil qilishda muhim bo'lgan nuqtalardan biri yechimning tushunchalari.** Ushbu masalada topilgan

yechim  $x_1=1, x_2=1$   $x_1 = 1, x_2 = 1$   $x_1=1, x_2=1$  va  $Z=5$  ni ta'minlaydi. Bu, masalani bajarish jarayonida optimal yechimni topishni aniqlaydi.

- **Masala yechishda Simplex usulining xavfsiz va ishonchli yechimlarni ta'minlashda qanday ishlatilganligini tushunish ham muhimdir.**
- **Qanday qilib masala yechishda Simplex usulidan foydalanishni ko'proq amaliy misollar orqali o'rganish tavsiya etiladi.** Bu masalani yechishda, masalaning ko'priki yechimlarini topish uchun Simplex usulini ishlatish muhim bo'lishi mumkin. Masalani bajarishda qo'llaniladigan usul va algoritmlarni o'rganish va tushuntirish ishlab chiqarilgan yechimlar tahlilidan foydalanishga yordam beradi.
- **Yechimning ma'nosi va aniqligi:** Topilgan yechim  $x_1=1, x_2=1$   $x_1 = 1, x_2 = 1$   $x_1=1, x_2=1$  va  $Z=5$ , deb ma'nosini anglatadi. Bu masaladagi  $ZZZ$  funksiyasi uchun maksimum yechimni ta'minlaydi. Bu, masala yechishning optimal natijasini ko'rsatadi.
- **Masala yechishda foydalanilgan Simplex usuli:** Simplex usuli masala yechish uchun samarali algoritim bo'lib, ushbu masalani yechishda tez va aniq yechimlar beradi. Ushbu usul yechimning aniqligi va to'liqligi haqida ishonch yaratadi.
- **Qo'llanilgan yechimlar tahlili:** Masala yechish jarayonida topilgan yechimning tushunchalari va ma'nosi qanday e'tiborli bo'lishi kerak. Bu tahlil natijasida, masala yechish algoritmini qanday darajada ishlatishimiz va uning ta'siri haqida to'liq fahmlashimiz mumkin.
- **Sug'atlanayotgan ko'rinishlar:** Masala yechish natijalarini tahlil qilishda sug'atlanayotgan ko'rinishlar, masalani eng samarali va to'liq hal qilishga qaratilgan yo'nalishlarni aniqlashga yordam beradi. Ushbu tushunchalar ishlab chiqarilgan yechimlar tahlilidan qanday foydalanishimizni ko'rsatadi.

## S216 Chiziqli dasturlash masalalarining tayanch yechimlari soni va tuzish usuli.

Chiziqli dasturlash masalalarining tayanch yechimlari soni va tuzish usuli quyidagicha bo'ladi:

### 1. Tayanch yechimlari soni:

- Chiziqli dasturlash masalasining tayanch yechimlari soni, masalaning tarkibiy elementi va qo'yilgan cheklovlar soni bilan aniqlanadi.
- Agar  $n$  ta o'zgaruvchi va  $m$  ta cheklov ( $m \leq n$ ) bo'lsa, unda masalaning tayanch yechimlari soni  $m!$  (factorial) ga teng bo'ladi.
- Agar masalada qo'shimcha cheklovlar mavjud bo'lsa, unda tayanch yechimlari soni kamroq bo'lishi mumkin.

### 2. Tayanch yechimlari tuzish usuli:

- Tayanch yechimlari tuzish uchun quyidagi yondashuvlardan foydalanish mumkin:
  - Tayyor jadvallardan foydalanish (masalan, simpleks metodi jadvali).
  - Algebraik usuldan foydalanib,  $m$  ta cheklovdan  $m$  ta asosiy o'zgaruvchini aniqlab, ularni o'zaro almashtirib, tayanch yechimlari tuzish.
  - Grafik usuldan foydalanish, ya'ni masalani grafik yechish va tayanch yechimlari sifatida grafik yechimlarni olish.

Masalan, 3 ta o'zgaruvchi va 2 ta cheklovdan iborat bo'lgan chiziqli dasturlash masalasida:

- Tayanch yechimlari soni:  $2! = 2$  ta
- Tayanch yechimlari tuzish:
  1. Birinchi cheklovdan birinchi va ikkinchi o'zgaruvchini, ikkinchi cheklovdan uchinchi va birinchi o'zgaruvchini topib, ularni o'zaro almashtirib, birinchi tayanch yechimni tuzish.
  2. Birinchi cheklovdan birinchi va uchinchi o'zgaruvchini, ikkinchi cheklovdan ikkinchi va birinchi o'zgaruvchini topib, ularni o'zaro almashtirib, ikkinchi tayanch yechimni tuzish.

Demak, chiziqli dasturlash masalasining tayanch yechimlari soni va tuzish usuli, masalaning tarkibiy elementlari va qo'yilgan cheklovlarga bog'liq bo'ladi.

Chiziqli dasturlash masalalarining tayanch yechimlarini topish uchun bir nechta tuzilish usullari mavjud, ammo ularning barchasi kritik sonli hisoblashni ta'minlashni maqsad qiladi. Ushbu usullar quyidagilar bo'lishi mumkin:

1. **Simplex Usuli:** Bu usul, kuchli va keng tarqalgan tuzilish usuli hisoblanadi. Ushbu usulda, masala linear-programming (LP) shaklida formulalangan va optimal yechimni topish uchun matritsalar yoki cizgilar bilan ifodalangan funksiya orqali yechishni o'rganadi.

2. **Gradientni Tushuntirish Usuli:** Ushbu usul, funksiyaning tushuntirilgan gradienti orqali yechishni amalga oshiradi. Masala gradientni tushuntirish formulalar orqali ifodalangan bo'lsa, bu usul o'zgacha yechishni amalga oshiradi.
3. **Boshlang'ich Qo'llanilgan Nuqtalarni Izlash:** Masala boshlang'ich qiymatlari asosida funksiyaning kengi oraliqda yechimni topishga qaratiladi. Usul har qanday funksiya uchun ishlaydi, ammo uni optimallashtirish va to'liq yechimni topish shart emas.
4. **N-yechimlarni qabul qilish (Nelder-Mead):** Bu usulda, masalani hal qilish uchun NNN ta boshlang'ich nuqta tanlanadi. Har bir bosqichda, yuqori qiymatga ega bo'lgan nuqta tanlanadi va yangi yechimlar topiladi.
5. **Qadamlashtirilgan (Iterative) Usullar:** Bu turdagi usullar masala hal qilish jarayonida funksiya qiymatlarini qadamlashtirib yechimni aniqlashda ishlatiladi. Misol uchun, ko'rsatkichlar qatlamalaridan o'tib, optimal yechimni topish uchun iteratsiyalarni ishga tushirish mumkin.

Bu usullar orasida simplex usuli keng tarqalgan dasturlash masalalarini yechishda ko'p qo'llaniladi. Ularni yuzaga chiqarish, optimal yechimni topishga qaratilgan amalga oshirish qulayligi va samaradorlik sababidan kelib chiqqan. Boshqa masalalarda esa boshqa usullar ishlatilishi lozim bo'ladi. Bu masalalarda masalaning xususiyatlariga qarab optimal yechimni topish uchun mos keluvchi usulni tanlash kerak.

Chiziqli dasturlash masalalarining tayanch yechimlarini topish uchun bir nechta tuzilish usullari mavjud. Ularning asosiylari quyidagilardir:

1. **Simplex Usuli:** Bu usul kuchli va ommalashgan usul bo'lib, keng tarqalgan chiziqli dasturlash masalalarini yechishda ishlatiladi. Simplex usuli masala linear-programming (LP) shaklida formulalangan va optimal yechimni topish uchun matritsalar yoki cizgilar bilan ifodalangan funksiya orqali yechishni o'rganadi.
2. **Gradientni Tushuntirish Usuli:** Bu usul funksiyaning tushuntirilgan gradienti orqali yechishni amalga oshiradi. Agar masala gradientni tushuntirish formulalar orqali ifodalangan bo'lsa, bu usul o'zgacha yechishni amalga oshiradi.
3. **Boshlang'ich Qo'llanilgan Nuqtalarni Izlash:** Masala boshlang'ich qiymatlari asosida funksiyaning kengi oraliqda yechimni topishga qaratiladi. Usul har qanday funksiya uchun ishlaydi, ammo uni optimallashtirish va to'liq yechimni topish shart emas.
4. **N-yechimlarni qabul qilish (Nelder-Mead):** Bu usulda, masalani hal qilish uchun NNN ta boshlang'ich nuqta tanlanadi. Har bir bosqichda, yuqori qiymatga ega bo'lgan nuqta tanlanadi va yangi yechimlar topiladi.

5. **Iteratsion Usullar:** Bu turdagi usullar masala hal qilish jarayonida funksiya qiymatlarini qadamlashtirib yechimni aniqlashda ishlatiladi. Misol uchun, ko'rsatkichlar qatlamalaridan o'tib, optimal yechimni topish uchun iteratsiyalarni ishga tushirish mumkin.

Bu usullar orasida, simplex usuli keng tarqalgan dasturlash masalalarini yechishda ko'p qo'llaniladi. Ularni yuzaga chiqarish, optimal yechimni topishga qaratilgan amalga oshirish qulayligi va samaradorlik sababidan kelib chiqqan. Boshqa masalalarda esa boshqa usullar ishlatilishi lozim bo'ladi. Masalan, masalaning xususiyatlariga qarab optimal yechimni topish uchun mos keluvchi usulni tanlash kerak.

Chiziqli dasturlash masalalarining tayanch yechimlarini topishda quyidagi tuzilish usullari mavjud:

1. **Simpleks Usuli:** Bu usul kuchli va ko'p qo'llaniladigan tuzilish usulidir. Simpleks usuli masala lineyar-programming (LP) shaklida formulalangan va optimal yechimni topish uchun matritsalar yoki chizg'ilar bilan ifodalangan funksiya orqali yechishni o'rganadi.
2. **Gradient Tushuntirish Usuli:** Bu usul funksiyaning gradienti yordamida yechishni amalga oshiradi. Agar masala gradientni tushuntirish formulalar orqali formulalanib bo'lsa, bu usul o'zgacha yechishni amalga oshiradi.
3. **Boshlang'ich Qo'llanilgan Nuqtalar:** Masala boshlang'ich qiymatlari asosida funksiyaning kengi oraliqda yechimni topishga qaratiladi. Usul har qanday funksiya uchun ishlaydi, ammo uni optimallashtirish va to'liq yechimni topish shart emas.
4. **N-yechimlarni Qabul Qilish (Nelder-Mead):** Bu usulda, masala hal qilish uchun NNN ta boshlang'ich nuqta tanlanadi. Har bir bosqichda, yuqori qiymatga ega bo'lgan nuqta tanlanadi va yangi yechimlar topiladi.
5. **Iteratsion Usullar:** Bu turdagi usullar masala hal qilish jarayonida funksiya qiymatlarini qadamlashtirib yechimni aniqlashda ishlatiladi. Misol uchun, ko'rsatkichlar qatlamalaridan o'tib, optimal yechimni topish uchun iteratsiyalarni ishga tushirish mumkin.

Bu usullar orasida, simplex usuli keng tarqalgan dasturlash masalalarini yechishda ko'p qo'llaniladi. Ularni yuzaga chiqarish, optimal yechimni topishga qaratilgan amalga oshirish qulayligi va samaradorlik sababidan kelib chiqqan. Boshqa masalalarda esa boshqa usullar ishlatilishi lozim bo'ladi. Masalan, masalaning xususiyatlariga qarab optimal yechimni topish uchun mos keluvchi usulni tanlash kerak.





## **S217 Chiziqli dasturlash masalalarining Transport masalasi.**

Chiziqli dasturlash masalalarining Transport masalasi quyidagilarni o'z ichiga oladi:

### **1. Transport masalasi nima?**

- Transport masalasi - mahsulotlarni bir qancha manbalardan boshqa qancha nuqtalarga tashish bo'yicha optimal rejani tuzishga qaratilgan chiziqli dasturlash masalasidir.
- Masalada asosiy maqsad - mahsulotlarni yetkazish xarajatlarini eng kam qilish.

### **2. Transport masalasining asosiy elementlari:**

- Manbalar (ishlab chiqaruvchilar, omborlar va h.k.)
- Iste'molchilar (bozorlar, do'konlar va h.k.)
- Har bir manbadan har bir iste'molchiga mahsulot yetkazish xarajatlari

### **3. Transport masalasining matematik modeli:**

- Manbalar soni -  $m$ , iste'molchilar soni -  $n$
- $a[i]$  -  $i$ -chi manbadagi mahsulot miqdori
- $b[j]$  -  $j$ -chi iste'molchining talabi
- $c[i,j]$  -  $i$ -chi manbadan  $j$ -chi iste'molchiga yetkazish xarajatlari
- $x[i,j]$  -  $i$ -chi manbadan  $j$ -chi iste'molchiga yetkazilayotgan mahsulot miqdori

Maqsad funksiyasi:  $\sum_{i=1 \text{ dan } m \text{ gacha}} \sum_{j=1 \text{ dan } n \text{ gacha}} c[i,j] * x[i,j] \rightarrow \min$   
Cheklovlar:

- $\sum_{j=1 \text{ dan } n \text{ gacha}} x[i,j] \leq a[i] \quad (i=1,2,\dots,m)$
- $\sum_{i=1 \text{ dan } m \text{ gacha}} x[i,j] = b[j] \quad (j=1,2,\dots,n)$
- $x[i,j] \geq 0$

### **4. Transport masalasini yechish usullari:**

- Shimol-Garbiy burchak usuli
- Eng kichik xarajat usuli
- Potensiallar usuli

Xulosa qilib aytganda, transport masalasi chiziqli dasturlash masalalarining muhim turlaridan biri bo'lib, mahsulotlarni manbalardan iste'molchilarga eng kam xarajat bilan yetkazish rejalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba jjj uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_{ij}$ ,  $j=1,2,\dots,n$
2. Har bir tovar iii uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_{ij}$ ,  $i=1,2,\dots,m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:**  $m$  ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:**  $n$  ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_j$  -  $j$ -tomonidagi tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  -  $i$ -tomonidagi tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  -  $i$ -tomonidagi tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n, j=1, 2, \dots, n$   
 $x_{ij} \leq a_j, j=1, 2, \dots, n, j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m, i=1, 2, \dots, m$   
 $x_{ij} \leq b_i, i=1, 2, \dots, m, i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0, x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:**  $m$  ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:**  $n$  ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_j$  -  $j$ -tomonidagi  $j$ -tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  -  $i$ -tomonidagi  $j$ -tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  -  $i$ -tomonidagi  $j$ -tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n, j=1, 2, \dots, n$   
 $x_{ij} \leq a_j, j=1, 2, \dots, n, j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m, i=1, 2, \dots, m$   
 $x_{ij} \leq b_i, i=1, 2, \dots, m, i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0, x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba jjj uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_{ij}$ ,  $j=1,2,\dots,n$
2. Har bir tovar iii uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_{ij}$ ,  $i=1,2,\dots,m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba jjj uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_{ij}$ ,  $j=1,2,\dots,n$
2. Har bir tovar iii uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_{ij}$ ,  $i=1,2,\dots,m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:**  $m$  ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:**  $n$  ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_j$  -  $j$ -tomonidagi tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  -  $i$ -tomonidagi tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  -  $i$ -tomonidagi tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:



1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n, j=1, 2, \dots, n$   
 $x_{ij} \leq a_{ij}, \sum_{i=1}^m x_{ij} \leq a_{ij}, j=1, 2, \dots, n, j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m, i=1, 2, \dots, m$   
 $x_{ij} \leq b_{ij}, \sum_{j=1}^n x_{ij} \leq b_{ij}, i=1, 2, \dots, m, i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0, x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:**  $m$  ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:**  $n$  ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  -  $i$ -tomonidagi  $j$ -tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  -  $i$ -tomonidagi  $j$ -tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  -  $i$ -tomonidagi  $j$ -tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

Minimize  $Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$  \text{Minimize }  $Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n, j=1, 2, \dots, n$   
 $x_{ij} \leq a_{ij}, \sum_{i=1}^m x_{ij} \leq a_{ij}, j=1, 2, \dots, n, j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m, i=1, 2, \dots, m$   
 $x_{ij} \leq b_{ij}, \sum_{j=1}^n x_{ij} \leq b_{ij}, i=1, 2, \dots, m, i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0, x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba jjj uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_{ij}$ ,  $j=1,2,\dots,n$
2. Har bir tovar iii uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_{ij}$ ,  $i=1,2,\dots,m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba jjj uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_{ij}$ ,  $j=1,2,\dots,n$
2. Har bir tovar iii uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_{ij}$ ,  $i=1,2,\dots,m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:**  $m$  ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:**  $n$  ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_j$  -  $j$ -tomonidagi tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  -  $i$ -tomonidagi tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  -  $i$ -tomonidagi tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n, n_j=1, 2, \dots, n$   
 $x_{ij} \leq a_{ij}, \sum_{i=1}^m x_{ij} \leq a_{ij}, j=1, 2, \dots, n, n_j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m, m_i=1, 2, \dots, m$   
 $x_{ij} \leq b_{ij}, \sum_{j=1}^n x_{ij} \leq b_{ij}, i=1, 2, \dots, m, m_i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0, x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:**  $m$  ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:**  $n$  ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  -  $i$ -tomonidagi  $j$ -tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  -  $i$ -tomonidagi  $j$ -tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  -  $i$ -tomonidagi  $j$ -tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n, n_j=1, 2, \dots, n$   
 $x_{ij} \leq a_{ij}, \sum_{i=1}^m x_{ij} \leq a_{ij}, j=1, 2, \dots, n, n_j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m, m_i=1, 2, \dots, m$   
 $x_{ij} \leq b_{ij}, \sum_{j=1}^n x_{ij} \leq b_{ij}, i=1, 2, \dots, m, m_i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0, x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba jjj uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_{ij}$ ,  $j=1,2,\dots,n$
2. Har bir tovar iii uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_{ij}$ ,  $i=1,2,\dots,m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba jjj uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_{ij}$ ,  $j=1,2,\dots,n$
2. Har bir tovar iii uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_{ij}$ ,  $i=1,2,\dots,m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:**  $m$  ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:**  $n$  ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_j$  -  $j$ -tomonidagi tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  -  $i$ -tomonidagi tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  -  $i$ -tomonidagi tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:



1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n, n_j = 1, 2, \dots, n$   
 $x_{ij} \leq a_{ij}, \sum_{i=1}^m x_{ij} \leq a_{ij}, j=1, 2, \dots, n, n_j = 1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m, m_i = 1, 2, \dots, m$   
 $x_{ij} \leq b_{ij}, \sum_{j=1}^n x_{ij} \leq b_{ij}, i=1, 2, \dots, m, m_i = 1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0, x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:**  $m$  ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:**  $n$  ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  -  $i$ -tomonidagi  $j$ -tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  -  $i$ -tomonidagi  $j$ -tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  -  $i$ -tomonidagi  $j$ -tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba  $j$  uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_j, j=1, 2, \dots, n, n_j = 1, 2, \dots, n$   
 $x_{ij} \leq a_{ij}, \sum_{i=1}^m x_{ij} \leq a_{ij}, j=1, 2, \dots, n, n_j = 1, 2, \dots, n$
2. Har bir tovar  $i$  uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_i, i=1, 2, \dots, m, m_i = 1, 2, \dots, m$   
 $x_{ij} \leq b_{ij}, \sum_{j=1}^n x_{ij} \leq b_{ij}, i=1, 2, \dots, m, m_i = 1, 2, \dots, m$
3. Xohlagancha  $x_{ij} \geq 0, x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

Transport masalasi, muntazam joylashgan manbalar va ehtiyotlar o'rtasida tovarlarni o'chirish jarayonini optimallashtirish uchun amaliy masalalardan biridir. Bu masalada, muntazam joylashgan manbalar va ehtiyotlar o'rtasida transportning narxi va chegara ko'rsatkichlar bilan tovarlar o'chiriladi. Ushbu masala tovarlar, manbalar va ehtiyotlar o'rtasidagi aloqani yetkazish jarayonini eng samarali vaqt va mablag' bilan optimallashtirishni maqsad qiladi.

Transport masalasi formulatsiyasi quyidagi ko'rinishda ifodalangan bo'ladi:

1. **Tovarlar:** mmm ta tovarlar (mashina, konteynerlar, to'rtliklar kabi).
2. **Manbalar:** nnn ta manbalar (qurilish materiallari, mahsulotlar, xom-ashyo, kabi).
3. **Ehtiyotlar:**  $a_{ij}$  - iii-tomonidagi jjj-tovar biriktirish joyining miqdori.
4. **Tovarlar yetkazish narxi:**  $c_{ij}$  - iii-tomonidagi jjj-tovarlar biriktirish narxi.
5. **Tovarlar yechimlari:**  $x_{ij}$  - iii-tomonidagi jjj-tovarlar miqdori.

Masala, quyidagi formulatsiyada ifodalangan bo'lib, tovarlar yetkazishining minimal narxini topish uchun yechimlar miqdorini topishni maqsad qiladi:

$$\text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad \text{Minimize } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

Shartlar:

1. Har bir manba jjj uchun taqrimiy chegaralar:  $\sum_{i=1}^m x_{ij} \leq a_{ij}$ ,  $j=1,2,\dots,n$
2. Har bir tovar iii uchun taqrimiy chegaralar:  $\sum_{j=1}^n x_{ij} \leq b_{ij}$ ,  $i=1,2,\dots,m$
3. Xohlagancha  $x_{ij} \geq 0$  va butun sonlar bo'lishi kerak.

Transport masalasi, ko'p sohada, masalan, transport va logistika sohalarida, masofaviy joylashtirish va mablag' hisoblashda, amaliyotlarda va dasturlashda ommalashtiriladi.

## S218 Chiziqli dasturlash masalalarining O'yinlar nazariyasi

Chiziqli dasturlash masalalarining O'yinlar nazariyasi quyidagicha:

### 1. O'yinlar nazariyasi nima?

- O'yinlar nazariyasi - raqobatli vaziyatlarda qarorlar qabul qilishni o'rganadigan matematika sohasidir.
- O'yinlar nazariyasida ishtirokchilar (o'yinchilar) o'z manfaatlarini qondirish uchun raqobatlashadilar.

### 2. Chiziqli dasturlash va o'yinlar nazariyasi aloqasi:

- Chiziqli dasturlash masalalarida ham o'yinlar nazariyasi prinsiplari qo'llaniladi.
- Chiziqli dasturlash masalasining maqsad funktsiyasi va cheklavlari o'yinchilarning manfaatlarini ifodalaydi.
- Masalani yechish orqali o'yinchilarning eng yaxshi strategiyalari topiladi.

### 3. O'yin shakllari:

- Ikki o'yinchi o'rtasidagi o'yin (ikki o'yinchili o'yin)
- Ikki guruh o'rtasidagi o'yin (ko'p o'yinchili o'yin)

### 4. O'yin turlari:

- Nol yig'indili o'yin (yutgan biri, yutqazgan ikkinchisi)
- Nol bo'lmagan yig'indili o'yin (barcha o'yinchilar bir-birlarini to'liq qondira olmasliklari)

### 5. O'yin yechimi usullari:

- Grafik usul
- Simpleks usul
- Dual simpleks usul

Xulosa qilib aytganda, chiziqli dasturlash masalalarida o'yinlar nazariyasi prinsiplari qo'llaniladi. O'yin shakli va turi hamda yechish usullari masalaning xususiyatlariga bog'liq bo'ladi.

O'yinlar nazariyasi, chiziqli dasturlashning bir sohasidir, u o'yinlar va chiziqli modellar yordamida muammolarni yechishni qo'lga tutadi. Bu qisqa muddatli vaziyatlarda ishlab chiqarish jarayonlarini optimallashtirishga qaratilgan mexanizmlardan foydalanadi. O'yinlar nazariyasi, masofaviy mablag'ni aniqlash, tizimni optimallashtirish va sifatni oshirish, yoki maxsus maqsadlar uchun eng yaxshi natijalarni olishda muvaffaqiyatli bo'lishi mumkin bo'lgan xilma-xil chiziqli dasturlash masalalarini ko'rsatadi.

Quyidagi formulatsiya chiziqli dasturlashda o'yinlar nazariyasini ifodalaydi:

1. **O'yinlar:**  $NNN$  ta o'yin mavjud.
2. **O'yinlar boshlanish holati:**  $sis\_isi, i=1,2,...,N; i=1, 2, ..., N; i=1,2,...,N$ .
3. **Har bir o'yin uchun so'nggi o'rin:**  $tit\_iti, i=1,2,...,N; i=1, 2, ..., N; i=1,2,...,N$ .
4. **Barcha o'yinlar uchun sarflangan vaqtni hisoblash:**

$$T = \sum_{i=1}^N t_i - \sum_{i=1}^N s_i$$

Bu formulatsiya o'yinlar bo'yicha sarflangan vaqtni hisoblashda yordam beradi. O'yinlar boshlanish vaqti  $s_i$  va har bir o'yin davomida o'zgaruvchan vaqt  $t_i$  kiritilgan. Shuningdek, o'yinlar boshlanish holati o'zgartiriladi va so'nggi o'yinlar  $t_i$  hisoblanadi. Uchta matnlar NNN o'yinlar uchun aniqlanadi.

O'yinlar nazariyasi, bir qancha sohalarida muvaffaqiyatli bo'lishi mumkin, masalan, transport va logistika sohalarida, masofaviy mablag' hisoblashda, va mijozlarga qo'llab-quvvatlashda amaliyotlarda va dasturlashda ommalashtiriladi.

## S219 Raqamli axborotlarni Fure qatoriga yoyish algoritmi. Ishonchligini baholash.

Raqamli axborotlarni Fure qatoriga yoyish algoritmi quyidagicha:

### 1. Fure qatoriga yoyish nima?

- Fure qatoriga yoyish - vaqt sohasidagi ixtiyoriy davriy signallarni, cheksiz yig'indi ko'rinishidagi trigonometrik qatorga yozish usuli.
- Bunda signal ixtiyoriy davriy funksiya sifatida tavsiflanadi va uning spektri aniqlanadi.

### 2. Fure qatoriga yoyish algoritmi:

- Signalning davrini  $T$  ga teng deb olamiz.
- Signalning Fure koeffitsientlarini quyidagi formulalar yordamida hisoblash:

$$a_0 = (1/T) \int_0^T f(t) dt$$

$$a_n = (2/T) \int_0^T f(t) \cos(2\pi n t/T) dt$$

$$b_n = (2/T) \int_0^T f(t) \sin(2\pi n t/T) dt$$

- Bu koeffitsientlar asosida Fure qatorini yozish:

$$f(t) = a_0/2 + \sum_{n=1}^{\infty} (a_n \cos(2\pi n t/T) + b_n \sin(2\pi n t/T))$$

### 3. Raqamli axborotlar uchun algoritm:

- Signalning diskret namunalarini olamiz:  $f[k] = f(k \cdot T/N)$ ,  $k=0,1,\dots,N-1$
- Fure koeffitsientlarini diskret Fure almashtirish (DFT) formulasi yordamida hisoblash:

$$a_0 = (1/N) \sum_{k=0}^{N-1} f[k]$$

$$a_k = (2/N) \sum_{k=0}^{N-1} f[k] \cos(2\pi k n/N)$$

$$b_k = (2/N) \sum_{k=0}^{N-1} f[k] \sin(2\pi k n/N)$$

- Fure qatorini shu koeffitsientlar asosida yozish.

### 4. Ishonchlilik baholash:

- Fure qatoriga yoyish jarayonida xatoliklar sodir bo'lishi mumkin:
- Signal diskretlashtirilganda
- Koeffitsientlar hisoblanganda
- Qator yig'indisi chekli qilib olinganda
- Bunday xatoliklarni kamaytirish uchun:
- Namuna olinayotgan signalning davri kam bo'lishi kerak
- Koeffitsientlar soni yetarlicha katta bo'lishi kerak
- Yig'indida chekli sonli hadlardan foydalanish

Xulosa qilib aytganda, Fure qatoriga yoyish algoritmi raqamli signallarning spektral tahlilini amalga oshirish uchun keng qo'llaniladi. Uning ishonchliligini oshirish uchun diskretlash va qator yig'indisi parametrlarini to'g'ri tanlash muhim.

Fure qatoriga yoyish algoritmi, bir nechta raqamli ma'lumotlarni qabul qilib, ularni o'zida saqlab, ulardan kelgan bo'shliklarni to'ldirib ketishni amalga oshiradi. Bu algoritmda ma'lumotlar Fure qatorida saqlanadi. Fure qatori - bu qatordan iborat

o'lchamli maydon. Agar bu qator bo'sh bo'lsa, ma'lumotlar yig'ilib qolgan tomondan joylashadi. Bu algoritim o'nlik sonlar ro'yxatini o'z ichiga oladi va har bir raqamni kerakli joyga joylashtiradi, shunday qilib, chiziqli vaqtda sodda to'ldirish algoritmi hisoblanadi. Bu algoritmda har bir yangi raqamni joylashish uchun qator bo'ylab o'zgaruvchan takrorlash kerak bo'ladi. Demak, u  $n$  raqamli ma'lumotlar uchun  $O(n^2)$  vaqt muddatini talab qiladi.

Bu algoritmda amalga oshiriladigan takrorlar soni raqamli ma'lumotlar soniga bog'liq bo'ladi. Raqamlar soni ko'payganda yoki o'zgarishga bog'liq bo'lib, uning samaradorligi o'zgaradi. Shuning uchun, katta miqdordagi raqamlar uchun bu algoritim murakkab hisoblanadi. Keyinchalik, qo'shimcha ma'lumotlar bilan ishlashda yoki kichik miqdordagi raqamlar uchun foydalanish mumkin.

## S220 Signal yetakchi garmonikalarini ajratish algoritmi. Spektral tahlil.

Signal yetakchi garmonikalarini ajratish algoritmi va spektral tahlil quyidagicha:

### 1. Yetakchi garmonikalarni ajratish nima uchun kerak?

- Ko'plab real signallar murakkab tarkibga ega bo'lib, bir nechta garmonikalardan iborat bo'ladi.
- Signalning yetakchi (dominant) garmonikalarini ajratish maqsadga muvofiq bo'ladi, chunki:
  - Yetakchi garmonikalar signalning asosiy xususiyatlarini ifodalaydi.
  - Yetakchi garmonikalarni ajratish orqali signalni tahlil qilish va qayta ishlash osonlashadi.

### 2. Yetakchi garmonikalarni ajratish algoritmi:

- Signalni Fure qatoriga yoyish.
- Fure koeffitsientlari ro'yxatini kamayish tartibida saralash.
- Eng katta koeffitsientlarga mos keladigan garmonikalarni ajratib olish.
- Ajratilgan garmonikalar asosida yangi signal tuzish:  
$$f(t) = \sum_{k=1}^n (A_k \cos(2\pi k f t) + B_k \sin(2\pi k f t))$$
- Bunga  $n$  ta yetakchi garmonika kiritiladi,  $f$  - yetakchi garmonikalar chastotasi.

### 3. Spektral tahlil:

- Signalning spektri - uning Fure koeffitsientlari.
- Signalning spektrini tahlil qilish orqali quyidagilarni aniqlash mumkin:
  - Signaldagi yetakchi chastotalar
  - Garmonikalarning nisbiy kuchliligi
  - Signalning davriy xarakteri
  - Signalning asosiy xususiyatlari

### 4. Spektral tahlil usullari:

- Grafik usul (signalning Fure koeffitsientlari grafigi)
- Raqamli usul (diskret Fure almashtirish, DFT)
- Matematik usul (Fure qatori formulalari)

Xulosa qilib aytganda, yetakchi garmonikalarni ajratish algoritmi signallarni tahlil qilish va qayta ishlashda muhim ahamiyatga ega. Spektral tahlil esa signallarning chastotaviy tarkibini o'rganish imkonini beradi.

Signal yetakchi garmonikalarini ajratish algoritmi va spektral tahlilning asosiy qadami Furie o'zgaruvchiligi (Fourier transform) hisoblanadi. Ushbu algoritm, bir signalni o'zining to'plam amplitudalariga bo'lgan garmonikalar (yoki sin va kosinus funktsiyalarining birligi) ajratadi.

`np.fft.fft` funksiyasi signalning Furie o'zgaruvchiligini hisoblaydi va `np.fft.fftfreq` funksiyasi spektrni olish uchun kerakli spektral qismatlar (frequency bins) ro'yxatini olish uchun ishlatiladi.

Bu algoritm spektral tahlilning bir qismini tashkil etadi va bir signalning yetakchi garmonikalarini aniqlash uchun juda samarali bo'lib, bir nechta sohada, masalan, audiolarda, elektronika, telekommunikatsiya va boshqalarida amalga oshiriladi.

3.5



## S221 Statistik modellashtirishda eng kichik kvadratlar usuli.

Statistik modellashtirishda eng kichik kvadratlar usuli quyidagicha amalga oshiriladi:

### 1. Eng kichik kvadratlar usulining maqsadi:

- Tajriba yoki kuzatuvlar natijasida olingan ma'lumotlar asosida matematik model tuzish.
- Modelning parametrlarini belgilash va model yordamida kuzatuvlar natijalarini yaqinlashtirish.

### 2. Usulning asosiy tamoyili:

- Haqiqiy ma'lumotlar va model natijalarining kvadratik farqining yig'indisi eng kichik bo'lishi kerak.
- Yani, model va haqiqiy ma'lumotlar orasidagi umumiy xatolik eng kichik bo'lishi lozim.

### 3. Eng kichik kvadratlar usulining algoritmi:

- Modelning umumiy ko'rinishi:  $y = f(x, a_1, a_2, \dots, a_n)$
- Haqiqiy kuzatuvlar ma'lumotlari:  $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$
- Quyidagi yig'indini eng kichik qilish kerak:  
$$Q = \sum_{i=1}^m (y_i - f(x_i, a_1, a_2, \dots, a_n))^2$$
- Bu yig'indini eng kichik qilishi uchun,  $Q$  ning  $a_1, a_2, \dots, a_n$  ga nisbatan xocil-larini 0 ga teng qo'yish va tenglamalar sistemasini yechish.

### 4. Usulning afzalliklari:

- Sodda va aniq algoritim
- Modelning parametrlarini hisoblash uchun qo'l keladi
- Matematik statistika nazariyasi asosida asoslangan

### 5. Usulning cheklavlari:

- Xato tarqalishi normal taqsimotga muvofiq bo'lishi kerak
- Kuzatuvlar soni yetarli bo'lishi kerak
- Modelning to'g'ri tanlanishi muhim

Xulosa qilib aytganda, eng kichik kvadratlar usuli statistik modellashtirishda keng qo'llaniladi. U model parametrlarini aniqlab, model va haqiqiy ma'lumotlar o'rtasidagi farqni minimal qilishga imkon beradi.

Statistik modellashtirishda eng kichik kvadratlar (Least Squares) usuli, ma'lumotlar jadvallaridan olingan ma'lumotlar to'plamini ta'minlovchi funktsiyani ifodalashda foydalaniladi. Ushbu usulda, ma'lumotlar jadvallaridan olingan ko'p nusxalarni ( $x$  qiymatlari) ta'minlovchi funktsiyalarga mos keluvchi yomonliklar kvadratlar yig'indisi (summasi) eng kam bo'lgan mos funktsiya aniqlanadi. Ushbu usul statistik modellashtirishda o'rganilgan ma'lumotlarni o'rganilgan modellarga mos ravishda masofaviy ma'lumotlarni qo'llab-quvvatlashda va modellarning ma'lumotlar jadvallariga mosligini aniqlashda juda samarali bo'lgan eng keng tarqalgan usullardan biridir.

Statistik modellashtrishda eng kichik kvadratlar (Least Squares) usuli, ma'lumotlar jadvalaridan olingan ma'lumotlar to'plamini ta'minlovchi funktsiyani ifodalashda foydalaniladi. Ushbu usulda, ma'lumotlar jadvalaridan olingan ko'p nusxalarni ( $x$  qiymatlari) ta'minlovchi funktsiyalarga mos keluvchi yomonliklar kvadratlar yig'indisi (summasi) eng kam bo'lgan mos funktsiya aniqlanadi.

Ushbu usul statistik modellashtrishda o'rganilgan ma'lumotlarni o'rganilgan modellarga mos ravishda masofaviy ma'lumotlarni qo'llab-quvvatlashda va modellarning ma'lumotlar jadvalariga mosligini aniqlashda juda samarali bo'lgan eng keng tarqalgan usullardan biridir.

Statistik modellashtrishda eng kichik kvadratlar (Least Squares) usuli, ma'lumotlar jadvalaridan olingan ma'lumotlar to'plamini ta'minlovchi funktsiyani aniqlashda foydalaniladi. Ushbu usulda, ma'lumotlar jadvalaridan olingan ko'p nusxalarni ( $x$  qiymatlari) ta'minlovchi funktsiyalarga mos keluvchi yomonliklar kvadratlar yig'indisi (summasi) eng kam bo'lgan mos funktsiya aniqlanadi.

Ushbu usul statistik modellashtrishda o'rganilgan ma'lumotlarni o'rganilgan modellarga mos ravishda masofaviy ma'lumotlarni qo'llab-quvvatlashda va modellarning ma'lumotlar jadvalariga mosligini aniqlashda juda samarali bo'lgan eng keng tarqalgan usullardan biridir.

## **S222 Dinamik dasturlash.**

Dinamik dasturlash quyidagi asosiy prinsiplarga asoslangan:

### **1. Umumiy yondashuv**

- Murakkab masalani kichik, oson yechiluvchi sub-masalalarga bo'lish.
- Sub-masalalarning optimal yechimlaridan foydalanib, asosiy masalaning eng yaxshi yechimini topish.

### **2. Optimallik prinsiplari**

- Har bir sub-masalaning optimal yechimi ushbu sub-masala uchun eng yaxshi yechimdir.
- Optimal yechim oldingi qadamlardagi eng yaxshi yechimlardan tuziladi.

### **3. Rekursiv yondashuv**

- Murakkab masalani kichik sub-masalalarga bo'lish.
- Sub-masalalarning yechimlaridan foydalanib, asosiy masalaning yechimini rekursiv ravishda hisoblash.

### **4. Dinamik dasturlash algoritmi**

- Bosqichma-bosqich yechimlarni hisoblash:
  1. Boshlang'ich holat va cheklovlarni aniqlash
  2. Rekursiv tenglamalarni tuzish
  3. Bosqichlar bo'yicha yechimlarni hisoblash
  4. Oxirgi bosqich yechimidan boshlab, rekursiv ravishda yechimni topish

### **5. Afzalliklari**

- Murakkab masalalarni osonroq sub-masalalardan tuzish
- Optimallik prinsipini qo'llab, samarali yechimlarni topish
- Takrorlanuvchi sub-masalalarni aniqlash va ularga yanadan foydalanish

### **6. Qo'llanish sohalari**

- Optimal boshqaruv masalalari
- Tayinlash va marshrutlash masalalari
- Dinamik loyihalash va tarmoqli rejalash
- Maxsus dasturlash masalalari

Xulosa qilib aytganda, dinamik dasturlash murakkab masalalarni samarali yechish uchun keng qo'llaniladi. U optimallik prinsiplaridan foydalanib, rekursiv yondashuvni amalga oshiradi.

Dinamik dasturlash, dasturlashda ma'lumotlar strukturasi va aloqada dinamik o'zgarishlarni yaratish va boshqarishga oid umumiy konseptdir. Bu struktura yordamida, dastur dastlabki ishlovini bajarishda kerak bo'lgan ma'lumotlar va shartlar bilan birlashtiriladi, keyin dastur ishlayotgan paytda dinamik ravishda o'zgarishi mumkin bo'lgan ma'lumotlar bilan doimiy ravishda yangilanadi.

Dinamik dasturlashning bir necha muhim yonlari bor, masalan:

1. **Dinamik ma'lumotlar strukturasi:** Ma'lumotlar o'zgarishiga moslashtirilgan struktura, qo'shimcha elementlarni qo'shish, o'chirish va o'zgartirish imkoniyatini beradi. Bu struktura yordamida, ma'lumotlar boshqarilishi dastlabki kodni o'zgartirishsiz yoki boshqa qismlarni doimiy tuzatishsiz amalga oshiriladi.
2. **Dinamik dasturlash algoritmlari:** Dinamik dasturlash algoritmlari, dasturlash paytida ma'lumotlar o'zgarishlarini aniqlash uchun ishlatiladi. Bu algoritmlar qo'shimcha ma'lumotlarni qo'shish, o'chirish va o'zgartirishning turli ko'rinishlarini ta'minlaydi.
3. **Dinamik tuzatuv:** Dinamik dasturlashda, dastur ishga tushirilgandan so'ng ma'lumotlar strukturasi o'zgartirish uchun tuzatilgan jarayonlarga murojaat qilish mumkin bo'ladi. Bu ma'lumotlar qo'shimcha ma'lumotlarni saqlash, ma'lumotlarni o'chirish, o'zgarishlar qilish, ma'lumotlar bilan operatsiyalarni amalga oshirish, va hokazo.

Dinamik dasturlashning bir qancha turlari mavjud, masalan, dinamik ma'lumotlar strukturasi (LinkedList, Stack, Queue kabi), dinamik ma'lumotlarni o'zgartirish algoritmlari (Array resize, Insertion/Deletion, Sorting algoritmlari kabi), dinamik tuzatuvlar (Memory management, Garbage Collection kabi).

Dinamik dasturlash, dasturlashda ma'lumotlar o'zgarishlari va strukturani boshqarishga oid keng qamrovli va samarali usul. Uning yordamida, dasturlashda kerak bo'lgan muammolarni yechish, dastlabki kodni o'zgartirmaslik va dastur davomida yangilanadigan ma'lumotlarni qabul qilish va qo'shilish, o'chirish va o'zgartirish imkoniyatlarini olish mumkin bo'ladi.

## S223 Chiziqli, kvadratik modellar tuzish algoritmi va dasturi.

Chiziqli va kvadratik modellarni tuzish algoritmi quyidagicha:

### Chiziqli model tuzish algoritmi:

1. Modelning umumiy ko'rinishini belgilash:
  - Chiziqli model:  $y = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n$
2. Tajriba yoki kuzatuvlar natijalaridan olingan ma'lumotlarni yig'ish:
  - $(x_{11}, x_{12}, \dots, x_{1n}, y_1), (x_{21}, x_{22}, \dots, x_{2n}, y_2), \dots, (x_{m1}, x_{m2}, \dots, x_{mn}, y_m)$
3. Eng kichik kvadratlar usulidan foydalanib, model parametrlarini ( $a_0, a_1, a_2, \dots, a_n$ ) hisoblash:
  - Matrits tenglamalarini tuzish va yechish
4. Model yordamida yangi ma'lumotlarni bashorat qilish:
  - $y = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n$

### Kvadratik model tuzish algoritmi:

1. Modelning umumiy ko'rinishini belgilash:
  - Kvadratik model:  $y = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n + b_{11}x_1^2 + b_{22}x_2^2 + \dots + b_{nn}x_n^2$
2. Tajriba yoki kuzatuvlar natijalaridan olingan ma'lumotlarni yig'ish:
  - $(x_{11}, x_{12}, \dots, x_{1n}, y_1), (x_{21}, x_{22}, \dots, x_{2n}, y_2), \dots, (x_{m1}, x_{m2}, \dots, x_{mn}, y_m)$
3. Eng kichik kvadratlar usulidan foydalanib, model parametrlarini ( $a_0, a_1, a_2, \dots, a_n, b_{11}, b_{22}, \dots, b_{nn}$ ) hisoblash:
  - Matrits tenglamalarini tuzish va yechish
4. Model yordamida yangi ma'lumotlarni bashorat qilish:
  - $y = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n + b_{11}x_1^2 + b_{22}x_2^2 + \dots + b_{nn}x_n^2$

Chiziqli va kvadratik modellar tuzish uchun eng kichik kvadratlar (Least Squares) usulidan foydalanish mumkin. Bu usulda, ma'lumotlar jadvallaridan olingan ma'lumotlar to'plamini ta'minlovchi funktsiyani ifodalashda foydalaniladi.  $x$  va  $y$  ma'lumotlarini qabul qiladi va ko'rsatkichlar  $a, b$  (linearni),  $a, b, c$  (kvadratikni) ni qaytaradi, qo'llanish ko'rsatilgan.

Ushbu usul statistik modellashtirishda o'rganilgan ma'lumotlarni o'rganilgan modellarga mos ravishda masofaviy ma'lumotlarni qo'llab-quvvatlashda va modellarning ma'lumotlar jadvallariga mosligini aniqlashda juda samarali bo'lgan eng keng tarqalgan usullardan biridir.

Chiziqli va kvadratik modellar tuzish uchun eng kichik kvadratlar (Least Squares) usulidan foydalanish mumkin. Bu usulda, ma'lumotlar jadvallaridan olingan ma'lumotlar to'plamini ta'minlovchi funktsiyani ifodalashda foydalaniladi.

x va y ma'lumotlarini qabul qiladi va ko'rsatkichlar  $a, b$  (linearni),  $a, b, c$  (kvadratikni) ni qaytaradi, qo'llanish ko'rsatilgan.

Ushbu usul statistik modellashtirishda o'rganilgan ma'lumotlarni o'rganilgan modellarga mos ravishda masofaviy ma'lumotlarni qo'llab-quvvatlashda va modellarning ma'lumotlar jadvalariga mosligini aniqlashda juda samarali bo'lgan eng keng tarqalgan usullardan biridir.

Chiziqli va kvadratik modellar tuzish uchun eng kichik kvadratlar (Least Squares) usulidan foydalanish mumkin. Bu usulda, ma'lumotlar jadvalaridan olingan ma'lumotlar to'plamini ta'minlovchi funktsiyani ifodalashda foydalaniladi.

x va y ma'lumotlarini qabul qiladi va ko'rsatkichlar  $a, b$  (linearni),  $a, b, c$  (kvadratikni) ni qaytaradi, qo'llanish ko'rsatilgan.

Ushbu usul statistik modellashtirishda o'rganilgan ma'lumotlarni o'rganilgan modellarga mos ravishda masofaviy ma'lumotlarni qo'llab-quvvatlashda va modellarning ma'lumotlar jadvalariga mosligini aniqlashda juda samarali bo'lgan eng keng tarqalgan usullardan biridir.

Chiziqli va kvadratik modellar tuzish uchun eng kichik kvadratlar (Least Squares) usulidan foydalanish mumkin. Bu usulda, ma'lumotlar jadvalaridan olingan ma'lumotlar to'plamini ta'minlovchi funktsiyani ifodalashda foydalaniladi.

x va y ma'lumotlarini qabul qiladi va ko'rsatkichlar  $a, b$  (linearni),  $a, b, c$  (kvadratikni) ni qaytaradi, qo'llanish ko'rsatilgan.

Ushbu usul statistik modellashtirishda o'rganilgan ma'lumotlarni o'rganilgan modellarga mos ravishda masofaviy ma'lumotlarni qo'llab-quvvatlashda va modellarning ma'lumotlar jadvalariga mosligini aniqlashda juda samarali bo'lgan eng keng tarqalgan usullardan biridir.

## S224 “Dag'al kuch” usuli bilan tartiblashtirish

"Dag'al kuch" (Merge Sort) usuli, O'rtacha sodda (Divide and Conquer) algoritmlarning biri hisoblanadi. Bu algoritm qat'iy miqdorda ma'lumotlarni bo'lib segmentlarga ajratadi, har bir segmentni saralash uchun yana alohida segmentlarga ajratadi va keyinchalik birlashtiradi. Bu usul yaxshi tartiblash va o'rtacha so'ralganliq uchun juda samarali bo'lgan bir algoritm hisoblanadi. "Dag'al kuch" usuli (Brute Force) bilan tartiblashtirish quyidagi algoritm asosida amalga oshiriladi:

### **Algoritm:**

1. Boshlangich holatni tanlang (masalan, to'g'ri tartiblanmagan massiv).
2. Barcha mumkin bo'lgan tartiblarni tekshiring.
3. Eng yaxshi (eng optimal) tartibni toping.

### **Bosqichlar:**

1. **Boshlangich holatni tanlash:**
  - Masalan, tartiblanmagan massivni boshlangicha holatga oling.
2. **Barcha mumkin bo'lgan tartiblarni tekshirish:**
  - Massivdagi har bir element bilan barcha boshqa elementlarni almashtiring.
  - Har bir almashtirish uchun massivning yangi tartibini hisoblab chiqing.
  - Barcha mumkin bo'lgan tartiblarni tekshirib chiqing.
3. **Eng yaxshi (eng optimal) tartibni tanlash:**
  - Tekshirilgan tartiblar orasidan eng yaxshisini tanlang.
  - Eng yaxshi tartib - massivning eng kichik (yoki eng katta) elementlar ketma-ketligi bo'lishi mumkin.

## S225 Kommivoyajer haqidagi masala.

Kommivoyajer masalasi (Traveling Salesman Problem, TSP) - grafda barcha tugunlarni eng qisqa yo'l bilan aylanib o'tish masalasidir.

### Masalaning qo'yilishi:

Berilgan:  $N$  ta shahar va ular o'rtasidagi masofalar matritsasi.

Topish kerak: eng qisqa umumiy yo'l, ya'ni barcha shaharlarga bir marta tashrif buyurib, so'ng boshlanish nuqtasiga qaytish.

### Masalani yechish algoritmlari:

#### 1. Dag'al kuch usuli (Brute Force):

- Barcha mumkin bo'lgan yo'llarni tekshirib, eng qisqasini tanlash.
- Murakkablik:  $O(n!)$

#### 2. Yaqinlash usuli (Approximation):

- Yaqinlash algoritmlaridan foydalanish (masalan, Nearest Neighbor, Christofides algoritmi).
- Murakkablik:  $O(n^2)$
- Qat'iy optimal yechimni kafolatlaydi, ammo yaxshi yechimni taqdim etadi.

#### 3. Dinamik dasturlash usuli:

- Butun masalani kichik qismlar (subshehamalar) ga bo'lish.
- Har bir qism uchun eng yaxshi yechimni topish va ularni birlashtirish.
- Murakkablik:  $O(n^2 * 2^n)$

#### 4. Vektor optimizatsiyasi usuli:

- Genetik algoritmlar, tabu qidiruvi, simulated annealing kabi evristik usullardan foydalanish.
- Murakkablik:  $O(n^2 * 2^n)$
- Optimal yechimni kafolatlamamasada, yaxshi natijalar beradi.

Kommivoyajer masalasi, bir nechta shaharlar orasidagi eng qisqa yo'lda barcha shaharlarni bir marta borish va qaytib keldikdan so'ng boshlang'ich shaharga qaytishni takomillashtirishni talab qiladi. Bu masala kombinatorlik optimizatsiya masalalari orasida eng mashhur bo'lib, quyidagi shaklda bayon etiladi:

1. **Masala formulatsiyasi:** Berilgan shaharlar ro'yxati va ular orasidagi masofalar. Kommivoyajer barcha shaharlarni bir marta borish va qaytib keldikdan so'ng boshlang'ich shaharga qaytishni takomillashtirishni talab qiladi. Undan tashqari, har bir shahar bir marta va faqat bir marta faol ko'rib chiqilishi kerak.
2. **Algoritm yechimi:** Kommivoyajer masalasini yechish uchun eng optimal yo'lni topish uchun ko'p o'rganilgan algoritm "Branch and Bound" usulidan foydalaniladi. Bu usul qator joylashgan yechimlar o'rnini topishda yordam beradi.



Kommivoyajer masalasi, bir nechta shaharlar orasidagi eng qisqa yo'lda barcha shaharlarni bir marta borish va qaytib keldikdan so'ng boshlang'ich shaharga qaytishni takomillashtirishni talab qiladi. Bu masala kombinatorlik optimizatsiya masalalari orasida eng mashhur bo'lib, quyidagi shaklda bayon etiladi:

1. **Masala formulatsiyasi:** Berilgan shaharlar ro'yxati va ular orasidagi masofalar. Kommivoyajer barcha shaharlarni bir marta borish va qaytib keldikdan so'ng boshlang'ich shaharga qaytishni takomillashtirishni talab qiladi. Undan tashqari, har bir shahar bir marta va faqat bir marta faol ko'rib chiqilishi kerak.
2. **Algoritm yechimi:** Kommivoyajer masalasini yechish uchun eng optimal yo'lni topish uchun ko'p o'rganilgan algoritm "Branch and Bound" usulidan foydalaniladi. Bu usul qator joylashgan yechimlar o'rnini topishda yordam beradi.

## S226 “Xassis” algoritmlar.

"Xassis" (Greedy) algoritmlar - bu yechimni qabul qilish jarayonida har bir qadamda eng yaxshi ko'rinayotgan tanlovni qilish orqali muammoni yechish usulini bildiradi.

"Xassis" algoritmi quyidagi xususiyatlarga ega:

1. **Lokal optimizatsiya:** Har bir qadamda eng yaxshi ko'rinayotgan tanlovni qabul qiladi, ammo global optimizatsiya kafolatlangani yo'q.
2. **Tezkor:** Yechimni tezda topishga intiladi, shuning uchun ko'pincha optimal yechimni taqdim etmaydi.
3. **Soddalik:** "Xassis" algoritmi asosida yotuvchi mantiq oddiyligi bilan ajralib turadi.

"Xassis" algoritmi eng yaxshi yechimni kafolatlash imkoniyatiga ega emas, lekin ko'p hollarda oson reallashtiriladigan va tezkor yechimlarni taqdim etadi.

**"Xassis" algoritmlar qo'llaniladigan ba'zi masalalar:**

1. **Eng qisqa yo'l topish** (Dijkstra algoritmi)
2. **Minimal qamrovli daraxt toppish** (Kruskal algoritmi)
3. **Maksimal oqim topish** (Ford-Fulkerson algoritmi)
4. **O'zgaruvchi kattaliklar asosida tartiblash** (Quicksort algoritmi)

**Misol: Optimal ob'ektlarni yig'ish (Knapsack masalasi)**

Berilgan:

- Har bir ob'ekt uchun qiymati va og'irligi.
- Maksimal olib ketish mumkin bo'lgan og'irlik.

Maqsad: Maksimal qiymatga ega bo'lgan ob'ektlarni tanlash.

"Xassis" algoritmi:

1. Har bir ob'ektni qiymat/og'irlik nisbatiga ko'ra tartiblash.
2. Tartiblanmagan ro'yxatdan eng katta qiymat/og'irlik nisbatiga ega bo'lgan ob'ektni olib, u uchun joyni band qilish.
3. Qolgan joyni mumkin bo'lgan eng yaxshi ob'ekt bilan to'ldirish.
4. To'ldirish jarayoni tugaguncha takrorlash.

Bu algoritmi global optimal yechimni kafolatlamas, ammo tezkor va oson amalga oshiriladi.

"Xassis" algoritmlar, ma'lum bir masalani yoki muammoyi yechish uchun mahsulotlashtirilgan algoritmlardir. Ular masalaning yomon yechimlarini qidirishda, yo'qotishlarni minimallashtirishda yoki bir turdagi optimallashtirilgan yechimlarni topishda yordam beradi. Bu tur algoritmlar boshqa barcha tegishli muammolarni yechish uchun tasarlanmagan, balki belgilangan maqsadni amalga oshirishga xoslashtirilgan bo'lib, ular o'zlariga xos xususiyatlarga ega.

Ko'p xassis algoritmlar mavjud bo'lib, ularning bir qanchasi quyidagilardir:

1. **Branch and Bound:** Eng yaxshi yechimni topish uchun kombinatsiyalarni hisoblash va ko'rsatilgan chegaralar orqali optimal yechimlarni izlash.
2. **Genetik algoritmlar:** Biologik genetika asosida yaratilgan optimizatsiya algoritmi, qisman harakat qiluvchi va xususiy kombinatsiyalar orqali eng yaxshi yechimlarni izlash.
3. **Simulatlangan annealing:** Metall isitish jarayoniga bog'liq boshqarish usuli, optimallashtirilgan yechimlarni izlashda ishlatiladi.
4. **Qaror qabul qilish shajarasi (Decision Tree):** Ma'lumotlar qabul qilinayotgan jarayonlarda eng yaxshi yechimni topishda ishlatiladi.
5. **Optimallashtirilgan qidiruv (Optimized Search):** Qidiruv algoritmlari yomon yechimlarni hisoblashda ishlatiladi va uning natijalarini optimallashtiradi.

Bu algoritmlar ma'lum bir maqsadni yechish uchun xususiy xususiyatlar bilan mahsulotlashtirilgan. Masalan, "Branch and Bound" algoritmi kombinatorlik optimizatsiya masalalarini yechishda foydalaniladi, genetik algoritmlar esa asosiy fikrlar evolyutsiyasini asoslaydi. "Xassis" algoritmlar asosan optimizatsiya masalalarini yechish uchun ishlatiladi.

"Xassis" algoritmlar, mahsulotlashtirilgan algoritmlar bo'lib, ma'lum bir masalani yechish uchun maxsus xususiyatlarga ega. Bu algoritmlar masalani yechish uchun mahsulotlashtirilgan va optimallashtirilgan xususiyatlarga ega bo'lgan, masalan, yomon yechimlarni qidirishda yordam beradigan algoritmlardir.

Quyidagi "xassis" algoritmlar misollarini ko'rsataman:

1. **Branch and Bound:** Eng yaxshi yechimni topish uchun kombinatsiyalarni hisoblash va ko'rsatilgan chegaralar orqali optimal yechimlarni izlash.
2. **Genetik algoritmlar:** Biologik genetika asosida yaratilgan optimizatsiya algoritmi, qisman harakat qiluvchi va xususiy kombinatsiyalar orqali eng yaxshi yechimlarni izlash.
3. **Simulatlangan annealing:** Metall isitish jarayoniga bog'liq boshqarish usuli, optimallashtirilgan yechimlarni izlashda ishlatiladi.
4. **Qaror qabul qilish shajarasi (Decision Tree):** Ma'lumotlar qabul qilinayotgan jarayonlarda eng yaxshi yechimni topishda ishlatiladi.
5. **Optimallashtirilgan qidiruv (Optimized Search):** Qidiruv algoritmlari yomon yechimlarni hisoblashda ishlatiladi va uning natijalarini optimallashtiradi.

"Xassis" algoritmlar ma'lum bir maqsadni yechish uchun maxsus xususiyatlarga ega va kerak bo'lgan muammolarni yechish uchun yaratilgan. Masalan, "Branch and

Bound" algoritmi kombinatorlik optimizatsiya masalalarini yechishda foydalaniladi, genetik algoritmlar esa asosiy fikrlar evolyutsiyasini asoslaydi.

"Xassis" algoritmlar, ma'lum bir masalani yechish uchun maxsus xususiyatlarga ega bo'lgan algoritmlardir. Bu algoritmlar o'zlariga xos maqsadlarni yechishda yordam beradilar.

Quyidagi "xassis" algoritmlar misollarini ko'rsataman:

1. **Branch and Bound:** Eng yaxshi yechimni topish uchun kombinatsiyalarni hisoblash va ko'rsatilgan chegaralar orqali optimal yechimlarni izlash.
2. **Genetik algoritmlar:** Biologik genetika asosida yaratilgan optimizatsiya algoritmi, qisman harakat qiluvchi va xususiy kombinatsiyalar orqali eng yaxshi yechimlarni izlash.
3. **Simulatlangan annealing:** Metall isitish jarayoniga bog'liq boshqarish usuli, optimallashtirilgan yechimlarni izlashda ishlatiladi.
4. **Qaror qabul qilish shajarasi (Decision Tree):** Ma'lumotlar qabul qilinayotgan jarayonlarda eng yaxshi yechimni topishda ishlatiladi.
5. **Optimallashtirilgan qidiruv (Optimized Search):** Qidiruv algoritmlari yomon yechimlarni hisoblashda ishlatiladi va uning natijalarini optimallashtiradi.

"Xassis" algoritmlar, kerakli maqsadni yechish uchun maxsus xususiyatlar va chegaralar bilan mahsulotlashtirilgan bo'lib, o'zlariga xos yechimlarni topishda yordam beradi. Masalan, "Branch and Bound" algoritmi kombinatorlik optimizatsiya masalalarini yechishda, genetik algoritmlar esa asosiy fikrlar evolyutsiyasini asoslaydi.

"Xassis" algoritmlar, ma'lum bir masalani yechish uchun maxsus xususiyatlarga ega bo'lgan algoritmlardir. Ular maqsadga muvofiqlashgan holda mahsulotlashtirilgan va optimallashtirilgan xususiyatlar bilan yaratiladi.

Quyidagi misollarni ko'rsataman:

1. **Branch and Bound:** Eng yaxshi yechimni topish uchun kombinatsiyalarni hisoblash va ko'rsatilgan chegaralar orqali optimal yechimlarni izlash.
2. **Genetik algoritmlar:** Biologik genetika asosida yaratilgan optimizatsiya algoritmi, qisman harakat qiluvchi va xususiy kombinatsiyalar orqali eng yaxshi yechimlarni izlash.
3. **Simulatlangan annealing:** Metall isitish jarayoniga bog'liq boshqarish usuli, optimallashtirilgan yechimlarni izlashda ishlatiladi.
4. **Qaror qabul qilish shajarasi (Decision Tree):** Ma'lumotlar qabul qilinayotgan jarayonlarda eng yaxshi yechimni topishda ishlatiladi.

5. **Optimallashtirilgan qidiruv (Optimized Search):** Qidiruv algoritmlari yomon yechimlarni hisoblashda ishlatiladi va uning natijalarini optimallashtiradi.

"Xassis" algoritmlar, kerakli maqsadni yechish uchun maxsus xususiyatlar va chegaralar bilan mahsulotlashtirilgan bo'lib, o'zlariga xos yechimlarni topishda yordam beradi. Masalan, "Branch and Bound" algoritmi kombinatorlik optimizatsiya masalalarini yechishda, genetik algoritmlar esa asosiy fikrlar evolyutsiyasini asoslaydi.

"Xassis" algoritmlar, ma'lum bir masalani yechish uchun mahsulotlashtirilgan algoritmlardir. Ular masalaning yomon yechimlarini qidirishda, yo'qotishlarni minimallashtirishda yoki bir turdagi optimallashtirilgan yechimlarni topishda yordam beradi. Bu tur algoritmlar boshqa barcha tegishli muammolarni yechish uchun tasarlانmagan, balki belgilangan maqsadni amalga oshirishga xoslashtirilgan bo'lib, ular o'zlariga xos xususiyatlarga ega.

Quyidagi "xassis" algoritmlar misollarini ko'rsataman:

1. **Branch and Bound:** Eng yaxshi yechimni topish uchun kombinatsiyalarni hisoblash va ko'rsatilgan chegaralar orqali optimal yechimlarni izlash.
2. **Genetik algoritmlar:** Biologik genetika asosida yaratilgan optimizatsiya algoritmi, qisman harakat qiluvchi va xususiy kombinatsiyalar orqali eng yaxshi yechimlarni izlash.
3. **Simulatsion annealing:** Metall isitish jarayoniga bog'liq boshqarish usuli, optimallashtirilgan yechimlarni izlashda ishlatiladi.
4. **Qaror qabul qilish shajarasi (Decision Tree):** Ma'lumotlar qabul qilinayotgan jarayonlarda eng yaxshi yechimni topishda ishlatiladi.
5. **Optimallashtirilgan qidiruv (Optimized Search):** Qidiruv algoritmlari yomon yechimlarni hisoblashda ishlatiladi va uning natijalarini optimallashtiradi.

"Xassis" algoritmlar, kerakli maqsadni yechish uchun maxsus xususiyatlar va chegaralar bilan mahsulotlashtirilgan bo'lib, o'zlariga xos yechimlarni topishda yordam beradi. Masalan, "Branch and Bound" algoritmi kombinatorlik optimizatsiya masalalarini yechishda, genetik algoritmlar esa asosiy fikrlar evolyutsiyasini asoslaydi.

## S227 Kruskal algoritmi.

Kruskal algoritmi - grafda minimal qamrovli daraxt (Minimum Spanning Tree, MST) ni topish uchun ishlatiladigan "xassis" algoritm.

### **Kruskal algoritmi qadamlari:**

1. Graftagi barcha qirralarni qiymatiga ko'ra o'sish tartibida saralash.
2. Bir-biriga tutashgan ikkita tugunni birlashtiradigan qirrani MST ga qo'shish.
  - Agar ushbu qirra qo'shilsa, MST da yangi halqa hosil bo'lsa, u rad etiladi.
3. Barcha tugunlar birlashtirilguncha 2-qadamni takrorlash.

### **Kruskal algoritmining ishlash prinsipi:**

- Grafning barcha qirralarini o'sish tartibida saralash.
- So'ng, har bir qirradan boshlab, agar u ikkita alohida domenni birlashtirsagina, uni MST ga qo'shish.
- Bu jarayon, barcha tugunlar bitta domenga birlashgunga qadar davom etadi.

### **Algoritmning muhim xususiyatlari:**

1. **Soddalik:** Kruskal algoritmi tushunish va amalga oshirish juda oson.
2. **Effektivlik:** Algoritm  $O(E \log E)$  murakkablikka ega, bu juda samarali.
3. **Optimal yechim:** Kruskal algoritmi har doim minimal qamrovli daraxtni topadi.

Kruskal algoritmi, minimal panjara bo'lgan va panjaradagi barcha tugunlarning yig'indisi minimal bo'lgan grafni yaratish uchun ishlatiladigan algoritm hisoblanadi. Bu algoritm katta grafni qisqa yo'lda panjara bo'lgan grafga aylantiradi.

Kruskal algoritmi quyidagi bosqichlardan iborat:

1. Barcha tugunlarni (kenarlar) masofaviy tartibda saralang.
2. Bo'sh panjarani yaratib olamiz.
3. Tugunlarni eng kam bo'lgan danagi bilan boshlanib, ularni panjaraga qo'shamiz.
4. Agar tugun panjarada yagona sikllangan bo'lsa, uni olib tashlaymiz.
5. Qolgan tugunlarni tayyorroqdan yuborib, birinchi bosqichga qaytib, 3-4 bosqichlarni takrorlaymiz.

Kruskal algoritmi minimal panjara bo'lgan va panjaradagi barcha tugunlarning yig'indisi minimal bo'lgan grafni yaratish uchun ishlatiladigan algoritm hisoblanadi. Bu algoritm katta grafni qisqa yo'lda panjara bo'lgan grafga aylantiradi.

Kruskal algoritmi quyidagi bosqichlardan iborat:

1. Barcha tugunlarni (kenarlar) masofaviy tartibda saralang.
2. Bo'sh panjarani yaratib olamiz.

3. Tugunlarni eng kam bo'lgan danagi bilan boshlanib, ularni panjaraga qo'shamiz.
4. Agar tugun panjarada yagona sikllangan bo'lsa, uni olib tashlaymiz.
5. Qolgan tugunlarni tayyorroqdan yuborib, birinchi bosqichga qaytib, 3-4 bosqichlarni takrorlaymiz.

Kruskal algoritmi minimal panjara bo'lgan va panjaradagi barcha tugunlarning yig'indisi minimal bo'lgan grafni yaratish uchun ishlatiladigan algoritm hisoblanadi. Bu algoritm katta grafni qisqa yo'lda panjara bo'lgan grafga aylantiradi.

Kruskal algoritmi quyidagi bosqichlardan iborat:

1. Barcha tugunlarni (kenarlar) masofaviy tartibda saralang.
2. Bo'sh panjarani yaratib olamiz.
3. Tugunlarni eng kam bo'lgan danagi bilan boshlanib, ularni panjaraga qo'shamiz.
4. Agar tugun panjarada yagona sikllangan bo'lsa, uni olib tashlaymiz.
5. Qolgan tugunlarni tayyorroqdan yuborib, birinchi bosqichga qaytib, 3-4 bosqichlarni takrorlaymiz.

Kruskal algoritmi minimal panjara bo'lgan va panjaradagi barcha tugunlarning yig'indisi minimal bo'lgan grafni yaratish uchun ishlatiladigan algoritm hisoblanadi. Bu algoritm katta grafni qisqa yo'lda panjara bo'lgan grafga aylantiradi.

Kruskal algoritmi quyidagi bosqichlardan iborat:

1. Barcha tugunlarni (kenarlar) masofaviy tartibda saralang.
2. Bo'sh panjarani yaratib olamiz.
3. Tugunlarni eng kam bo'lgan danagi bilan boshlanib, ularni panjaraga qo'shamiz.
4. Agar tugun panjarada yagona sikllangan bo'lsa, uni olib tashlaymiz.
5. Qolgan tugunlarni tayyorroqdan yuborib, birinchi bosqichga qaytib, 3-4 bosqichlarni takrorlaymiz.

## S228 Prima algoritmi va dasturlari.

Prima algoritmi - grafda minimal qamrovli daraxt (Minimum Spanning Tree, MST) ni topish uchun ishlatiladi. U "xassis" algoritmlar guruhiga kiradi.

### Prima algoritmining asosiy qadamlari:

1. Ixtiyoriy boshlang'ich tugunni tanlash.
2. Tanlangan tugunning barcha qo'shni tugunlari orasidan eng kichik og'irlikka ega bo'lgan qirradi bilan MST ga qo'shish.
3. MST ga yangi qo'shilgan tugun bilan uning qo'shni tugunlari orasidan eng kichik og'irlikka ega bo'lgan qirradi MST ga qo'shiladi.
4. 3-qadam, MST ga barcha tugunlar qo'shilguncha davom etadi.

### Prima algoritmining xususiyatlari:

1. **Lokal optimizatsiya:** Har bir qadamda eng yaxshi ko'rinayotgan tanlovni amalga oshiradi, lekin global optimal yechimni kafolatlaydi.
2. **Minimal qamrovli daraxt:** Prima algoritmi har doim minimal qamrovli daraxtni topadi.
3. **Soddalik:** Prima algoritmi tushunish va amalga oshirish juda oson.
4. **Effektivlik:** Prima algoritmi  $O(V^2)$  murakkablikka ega, bu nisbatan samarali.
5. "Prima algoritmi" ya'ni Prima algoritmi (ya'ni Prim's algorithm) graf teorisi va diskret matematikada ishlatiladigan bir algoritm. Bu algoritm orqali berilgan aloqador grafda minimal to'plamlarga ega bo'lgan yoki minimal bir tarqalganlar qo'shimchasiz orqali bir qismi (minimal enkürsiya) topish mumkin.
6. Algoritmning eng oddiy tushunchasi shundaki, boshlang'ich tugunchalar (boshlang'ich bo'lgan nuqta yoki to'plam) bilan boshlanib, undan boshlab yangi tugunchalar qo'shiladi va bu tugunchalarni bir-biriga bog'laydigan eng qisqa aloqalarni topish orqali minimal enkürsiyaning qisqasini topish.
7. Prima algoritmi tarmoq yoki aloqador grafning minimal ostiqlarini topishda juda qulay ishlaydi. U eng yaxshi tanilgan algoritmlardan biri hisoblanadi. Quyidagi dasturlar bilan bu algoritmi amalga oshirishingiz mumkin:



## S229 Xoffman daraxtlari, algoritm va dasturlari

Huffman daraxtlari - ma'lumotlarni siqish uchun ishlatiladigan statistik asosli kod. Ushbu daraxtlar ma'lumotlarni eng samarali siqish uchun qo'llaniladi.

### **Huffman algoritmi qadamlari:**

1. Matnda uchragan har bir belgining chastotasini hisoblash.
2. Chastotasi eng kam ikki belgini birlashtirib, ularning chastotasi yig'indisiga teng bo'lgan yangi belgi hosil qilish.
3. 2-qadamni, faqat bitta belgi qolib ketgunga qadar takrorlash.
4. Olingan daraxt Huffman daraxtini hosil qiladi. Har bir bargda bitta belgi, uning oldidagi raqamlar esa shu belgining kodini ifodalaydi.

### **Huffman algoritmining ishlash prinsipi:**

- Matnda uchraydigan har bir belgining chastotasini hisoblash.
- Eng kam chastotali ikki belgini birlashtirib, yangi birlashgan belgi yaratish.
- Ushbu jarayon faqat bitta belgi qolguncha davom etadi.
- Olingan daraxt Huffman daraxtini hosil qiladi.

### **Huffman daraxtining xususiyatlari:**

1. **Samaradorlik:** Huffman daraxtlari ma'lumotlarni siqishda juda samarali.
2. **Optimal yechim:** Huffman algoritmi har doim optimal kod uzunligini topadi.
3. **Soddalik:** Huffman algoritmi tushunish va amalga oshirish juda oson.
4. **Adaptivlik:** Huffman algoritmi matn tarkibiga moslashadi, ya'ni matnning tarkibiga qarab, har xil kod uzunliklariga ega bo'ladi.

## **S230 Kesishmaydigan to'plam ostilari va birlashmalarini qidirish algoritmi.**

Kesishmaydigan to'plam ostilari va birlashmalarini qidirish algoritmi, ikki yoki undan ko'p to'plamlarning kesishmasini topish va birlashmasini hisoblash uchun ishlatiladi. Bu algoritm, to'plamlar va birlashmalarni ma'lumotlar strukturasi sifatida olib, ularga ma'lumotlarni qo'shish, o'chirish, biriktirish va hisoblash imkoniyatini beradi.

**Kesishmaydigan to'plam ostilari va birlashmalarini qidirish algoritmining asosiy qadamlari:**

1. Berilgan to'plamlarni, ostilarni va birlashmalarni ma'lumotlar strukturasi sifatida ifodalash.
2. To'plamlarni, ostilarni va birlashmalarni kerakli operatsiyalarni amalga oshirish orqali boshqarish.
3. To'plamlarni, ostilarni va birlashmalarni biriktirish, hisoblash va qidirish uchun mos algoritmani yozish.
4. Siz "kesishmaydigan to'plam ostilari va birlashmalarini qidirish algoritmi" haqida so'z keltirgansiz. Bu amaliy masalani ko'rsatadigan algoritmdan boshqa nomlar ostidagi usul haqida gaplashsak, yordam bera olaman. Bu tushuncha "kesishmaydigan to'plam ostilari va birlashmalarini qidirish algoritmi" asosan "Disjoint Set Union" (DSU) algoritmi deb ataladi.
5. DSU algoritmi, to'plamlar va ularning birlashmalari ustida amal qilish uchun ishlatiladi. Bu algoritmnini qo'llab-quvvatlash maqsadida qo'shimcha yordamchi ma'lumotlar strukturasi, masalan, "Union-Find" strukturasi ham ishlatish mumkin. Bu struktura to'plamlarni yig'ish (union) va ularni tekshirish (find) funksiyalarini o'z ichiga oladi.

## S231 NP-to'liq masalalar. Hisoblashda yechilmaslik hollari.

NP-to'liq masalalar, nondeterministik polinomial samarali masalalar, hisoblashda yechilmaslik holatiga ega bo'lgan masalalardir. Bu masalalar qandaydir o'zgaruvchilarni, masalan, grafning uchuvchi to'plamlarini (subset) topish, bo'luvchi bo'lmagan bo'luvchini topish kabi masalalarga mas'ul.

Bu turi masalalar uchun oddiy yechimlar yo'q. Ammo, agar bir to'plamni olish va tekshirishning vaqt iste'moli qisqa bo'lsa, masala NP-to'liq deyiladi. NP-to'liq masalalarga yechim topish uchun avtomatizatsiya qilib borish mumkin, lekin bir bor tekshirish uchun juda katta vaqt talab qiladi. Agar yechim topish uchun vaqt kuchli algoritmlar bo'lsa, masala NP-to'liq deyiladi.

Quyidagi misollar NP-to'liq masalalarga misollar hisoblanadi:

1. **Bir to'plamning uchuvchi to'plamini topish (Subset Sum problem):** Berilgan sonlar to'plamidan berilgan mablag'ni tuzish mumkinligini aniqlash. Masalan, berilgan sonlar to'plamidan berilgan mablag'ni tuzish mumkinmi?
2. **Grafning maksimal yopiq bo'luvlari (Maximum Independent Set problem):** Berilgan grafda yopiq bo'luvlar ko'paytirish kerak. Boshqa so'zlar bilan, berilgan grafda barcha to'plamlar har bir uchuvchining uchuvchi to'plamini o'z ichiga olmasligi lozim.
3. **Mamlakatlararo pochtamatlar masalasi (Travelling Salesman Problem):** Berilgan mamlakatlararo shaharlar orasida eng qisqa yo'l qidirish. Bu, har bir shaharni faqat bir marta tashqi bo'lib borish, keyinroq qaytish shartlariga ega masala.

Bu kabi masalalar uchun hisoblashda yechilmaslik hollari ma'lum bo'lib, ularning yechimlarini topish uchun eng yaxshi algoritmlar avtomatizatsiya qilib borishga yordam beradi. Biroq, bu masalalar eng yaxshi yechim topish uchun kuchli algoritmlar talab qiladi, shuning uchun ularning yechimlari topsa ham, uni tekshirishning vaqt oson emas.

### 3.5

NP-to'liq masalalar - bu hisoblash nazariyasida maxsus o'ringa ega bo'lgan masalalar guruhi. Ularni yechish uchun ma'lum algoritmlarni yaratish juda qiyin yoki mumkin emas.

**NP-to'liq masalalarning asosiy xususiyatlari:**

1. **Yechimni tekshirish osonligi:** NP-to'liq masalalarning yechimlarini tekshirish nisbatan oson, lekin yechimni topish juda qiyin.

2. **Optimal yechimni topish qiyinligi:** Bu masalalarning optimal yechimlarini topish juda qiyin, hatto agar ma'lum vaqtda yechim topilgan bo'lsa.
3. **Exponensial murakkablik:** NP-to'liq masalalarning eng yaxshi algoritmlari exponensial vaqt murakkabligiga ega.

**NP-to'liq masalalarning misollari:**

1. **SAT (Satisfiability):** Mantiqiy ifodalarni qondiradigan qiymatlarni topish.
2. **Hamiltonch siklni topish:** Grafda har bir nuqtani faqat bir marta o'tib, yakuniy nuqtaga qaytadigan eng qisqa yo'lni topish.
3. **Umumlashtirilgan Sudoku:** Sudoku to'ldirishning umumlashtirilgan versiyasi.
4. **Bag'lash masalasi:** Berilgan ogirliklar va qiymatlar bilan predmetlar to'plamidan eng qimmatli to'plamni tanlash.

**Hisoblashda yechilmaslik hollariga misol:**

1. **Halting problem:** Berilgan dastur to'xtatilishini aniqlash muammosi.
2. **Aksiomlarga asoslangan matematika:** Matematik aksiomalardan foydalanib, to'g'ri yoki notog'ri mavjudligini aniqlash muammosi.
3. **Diofant tenglamalar:** Butun sonli yechimlarni topish muammosi.

Ushbu masalalar algoritmik yechimga ega emas, chunki ularni yechish uchun foydalaniladigan hisoblashlarning turli xil versiyalari mavjud bo'lib, ularning yechimi yoki yechilmasligini aniqlash juda qiyindir.

Bunday masalalarni yechish uchun odatda taxminiy yoki evristik algoritmlardan foydalaniladi. Biroq, bu algoritmlar optimal yechimni kafolatlashmaydi va ko'pincha sub-optimal yechimlarni taqdim etadi.

## S232 N-to'liq masalalarni yechish algoritmlarni qiyinligini baholash.

NP-to'liq masalalarni yechish algoritmlarining qiyinligini baholash uchun quyidagi faktorlarni ko'rib chiqamiz:

1. **Yechimning topilishi uchun kerak bo'lgan vaqt:** NP-to'liq masalalarni yechish algoritmlari uchun kerak bo'lgan vaqt murakkabligi kengayishi bilan algoritmlar qiyinlashadi. Agar masala kichik miqdordagi to'plamlar uchun ham yechim topishni talab etayotgan vaqt o'zgarishi kuchaygan bo'lsa, bu algoritm qiyin deb hisoblanadi.
2. **Masala miqdori:** Masala miqdori (size) o'zgaruvchisi ko'proq bo'lgan masalalar, ya'ni katta miqdorda ma'lumotlar uchun yechim topish algoritmlari qiyinlashadi. Masalan, grafning o'lchami, berilgan sonlar to'plami miqdori keng ko'paytirilgan bo'lsa, algoritmdagi ishlovlar soni murakkabligi kuchayadi.
3. **Algoritmdagi qiyinlik:** Algoritmdagi qiyinlik, masala o'ziga xos xususiyatlari bo'yicha algoritmlarning qiyinligini belgilaydi. Agar algoritm murakkabroq yoki muddati uzun bo'lsa, u qiyin deb baholash mumkin.
4. **Algoritmdagi yadrovi optimallashtirishlar:** Algoritmdagi yadrovi optimallashtirishlar, yani algoritmdagi optimallashtirishlar va to'g'ri ishlovlar soni, algoritmdagi qiyinlikka ta'sir qiladi. Bazi algoritm turlari, masalani yechishda optimallashtirilgan yordamni ishlatish orqali qisqa vaqt talab etishadi.
5. **Algoritmdagi yadrovi murakkablik:** Algoritmdagi murakkablik, yani xotira va protsessor kuchidagi miqdor, algoritmdagi qiyinlikka ta'sir qiladi. Keng miqdorda ma'lumotlar bilan ishlash va murakkablik oshirilgan algoritmlar murakkab deb hisoblanadi.

Masala yechish algoritmlarini baholashda ushbu faktorlarni ko'rib chiqish juda muhimdir. Agar algoritm kerak bo'lgan vaqtdan oshib ketayotgan, murakkab bo'lsa, yoki miqdori katta bo'lsa, uni qiyin algoritm deb baholash mumkin. Boshqa so'zlar bilan, eng yaxshi algoritm kerak bo'lgan vaqt, masala miqdori, algoritmdagi murakkablik va optimallashtirishlar bo'yicha qiyinligini baholashga asoslangan.

N-to'liq masalalarni yechish algoritmlarining qiyinligi baholashning asosiy usullari quyidagilar:

1. **Vaqt murakkabligi (Time Complexity):**
  - N-to'liq masalalar uchun eng yaxshi algoritmlar eksponensial vaqt murakkabligiga ega, ya'ni vaqt murakkabligi  $O(2^n)$  yoki  $O(n!)$  kabi.
  - Bu shuni anglatadiki, masalaning o'lchamlari ( $n$ ) oshishi bilan algoritm ishlash vaqti keskin oshib ketadi.

- Masalaning o'lchamlarini cheklash orqali algoritmni samarali qilish mumkin.

## 2. **Fazoviy murakkablik (Space Complexity):**

- N-to'liq masalalarni yechish uchun algoritmlar ko'pincha eksponensial fazoviy murakkablikka ega bo'ladi.
- Algoritmning foydalanayotgan xotira hajmi masalaning o'lchamlariga nisbatan eksponensial oshadi.
- Fazoviy murakkablikni kamaytirish uchun algoritmni modifikatsiya qilish yoki dinamik dasturlash kabi usullardan foydalanish mumkin.

## 3. **Parallelizatsiya qobiliyati:**

- N-to'liq masalalarni yechish uchun algoritmlar ko'pincha parallel hisoblashlarni qo'llab-quvvatlash qiyin bo'ladi.
- Masalaning tarkibiy qismlari o'zaro bog'liq bo'lganligi sababli, parallel hisoblarni amalga oshirish murakkab bo'ladi.
- Ba'zan parallel hisoblarni qo'llash orqali vaqt murakkabligini kamaytirish mumkin, lekin bu yondashuvda ma'lum cheklovlar mavjud.

## 4. **Taxminiy algoritmlar:**

- N-to'liq masalalarni yechish uchun taxminiy algoritmlardan foydalanish mumkin.
- Taxminiy algoritmlar optimal yechimni kafolatlashmaydi, lekin ma'lum vaqt va fazoviy cheklovlarda ishlashi mumkin.
- Taxminiy algoritmlar asosan evristik yondashuvlarga asoslangan bo'lib, yechimning sifatini kafolatlashmaydi.

Umumiy qilib aytganda, N-to'liq masalalarni yechish juda qiyin bo'lgan muammolar hisoblanadi. Ularni yechish uchun qo'llaniladigan algoritmlar asosan eksponensial vaqt va fazoviy murakkablikka ega bo'ladi. Shu sababli, masalaning o'lchamlarini cheklash, parallel hisoblashlarni qo'llash yoki taxminiy algoritmlardan foydalanish kabi usullar qo'llaniladi.

## S233 Graflarda erkin uchlarni tanlash, bo'yash. To'plamlarning to'plam ostilarini aniqlash, birlashtirish.

Graflar nazariyasida uchlarni tanlash, bo'yash va to'plamlarni aniqlash kabi masalalar muhim o'rin tutadi. Ushbu masalalar quyidagi yo'nalishlarga bo'linadi:

### 1. Erkin uchlarni tanlash (Independent Set):

- Erkin uchlar - bu bir-biri bilan ulanmagan uchlar to'plami.
- Eng katta erkin uchlar to'plami (Maximum Independent Set) - eng ko'p uchdan iborat bo'lgan erkin uchlar to'plami.
- Bu masala NP-to'liq masala hisoblanadi, chunki eng katta erkin uchlarni topish juda murakkab.
- Taxminiy algoritmlar orqali eng katta erkin uchlarni taxminiy topish mumkin.

### 2. Uchlarni bo'yash (Graph Coloring):

- Uchlarni bo'yash - graflarning uchlarni turli ranglar bilan bo'yash.
- Bundan maqsad: grafning har bir uchiga unga tutash uchlar bilan farq qiladigan rang berish.
- Eng kam rang soni (Chromatic Number) - grafni bo'yash uchun kerak bo'lgan eng kam rang soni.
- Bu masala ham NP-to'liq masala hisoblanadi.
- Taxminiy algoritmlar va evristik yondashuvlar orqali graflarni bo'yash mumkin.

### 3. To'plamlarning to'plam ostilari (Set Cover):

- To'plam ostisi - to'plamlarning kichik to'plami bo'lib, asl to'plam elementlarini o'z ichiga oladi.
- Eng kichik to'plam ostisi (Minimum Set Cover) - eng kam elementdan iborat to'plam ostisi.
- Bu masala ham NP-to'liq masala hisoblanadi.
- To'plam ostisini topish uchun heuristic va approksimatsion algoritmlardan foydalanish mumkin.

### 4. To'plamlarni birlashtirish (Set Packing):

- To'plamlarni birlashtirish - o'zaro kesishmagan to'plamlarni topish.
- Eng katta to'plam paketi (Maximum Set Packing) - eng ko'p elementdan iborat o'zaro kesishmagan to'plamlar to'plami.
- Bu masala ham NP-to'liq masala hisoblanadi.
- To'plam paketini topish uchun heuristic va approksimatsion algoritmlardan foydalanish mumkin.

Ushbu masalalar NP-to'liq muammolar bo'lib, ularni yechish uchun eksponensial vaqt va fazoviy murakkablikka ega algoritmlardan foydalanish kerak. Shu sababli, graflarni so'ndirib, taxminiy yechimlar topish yoki parallel hisoblash usullaridan foydalanish mumkin.

Graflarda erkin uchlarni tanlash (Vertex Cover problem) va to'plamlarning to'plam ostilarini aniqlash va birlashtirish (Union of Sets problem) masalalari ham ko'p yechim topishga oid NP-to'liq masalalardan ba'zilaridir. Bu masalalar uchun yechim topish uchun murakkab algoritmlar mavjud.

1. **Graflarda erkin uchlarni tanlash (Vertex Cover problem):** Berilgan grafda har bir yelka (edge) uchun eng kam uchlu yelkalarni tanlash. Bu masala NP-to'liq masalalarga misol bo'lib, eng kam uchlu yelkalarning to'plami (erkin uchlarni to'plami) topilishi juda murakkabdir. Eng yaxshi algoritm, ishlab chiqilgan "Greedy" algoritm bo'lib, bu algoritm grafdagi erkin uchlarni har bir yelkani qoplab borish orqali tanlaydi.
2. **To'plamlarning to'plam ostilarini aniqlash va birlashtirish (Union of Sets problem):** Berilgan to'plamlarning to'plam ostilarini topish va ularni birlashtirish ham NP-to'liq masalalardan biridir. Bu masalada, har bir to'plamning osti topilishi va uni boshqa to'plamlar bilan birlashtirish zarur. Agar to'plam osti topilgan bo'lsa, uni birlashtirishning minimal usuli topilishi kerak. Bu masala uchun ham "Greedy" algoritm, qo'llaniladi, lekin ba'zi holatlarda optimal yechim topish murakkab bo'lishi mumkin.

"Greedy" algoritmlar "Vertex Cover" va "Union of Sets" masalalarining qiyinligini baholash uchun yordam beradi. Bundan tashqari, har bir algoritmning natijalarini sinovga solish va ularni murakkablik, kerak bo'lgan vaqt, va natijalar kengligi bo'yicha baholash kerak. Quydagi misollar "Greedy" algoritmlarini ishlatish orqali "Vertex Cover" va "Union of Sets" masalalariga yechim topishning oddiy qadamlarini ko'rsatadi: Bu dasturlar "Greedy" algoritmlarini ishlatish orqali "Vertex Cover" va "Union of Sets" masalalariga yechim topishning oddiy yechimini ko'rsatadi. Buni sinovga solish va yechimlarni baholash muhimdir. Agar sizning boshqa savollaringiz yoki qo'ng'iroqlaringiz bo'lsa, menga yozing!

"Greedy" algoritmlar "Vertex Cover" va "Union of Sets" masalalariga yechim topishda keng qo'llaniladi. Ammo ularning barcha holatlarda optimal yechimni topishga erishishini ta'minlash kutilmas, shuning uchun ularni qanday darajada yaxshi yoki darajada yomon yechishganligini tushuntirish muhimdir.

Birinchi narsa, "Greedy" algoritmlar oddiyroq va tez ishlaydigan algoritmlardir. Bu algoritmlar kerak bo'lgan vaqt miqdorini kammaytiradigan ko'paytirilgan yechimlarga olib borishadi. Buning natijasida, ular ko'p masalalarda ishlaydi, ammo ularning yechimlari optimal emas bo'lishi mumkin.



## S234 Taqribiy integrallash usullari. Zaruriy aniqlikni ta'minlovchi qadamni tanlash.

Taqribiy integrallash usullari orasida eng mashhurlaridan biri to'g'ri to'rtburchak usuli (Rectangle Rule) hisoblanadi. Bu usulda integral quyidagi formula bilan hisoblangan:

$$\int_a^b f(x) dx \approx \sum_{i=0}^{n-1} f(x_i) \Delta x$$

Bu yerda  $\Delta x = \frac{b-a}{n}$  bo'lib,  $n$  ta qadamdan iborat.

Ushbu usul bir qator taqribiy usullar orasida eng sodda usul hisoblanadi. Biroq, aniqlik darajasi past. Aniqlikni oshirish uchun ushbu usulning modifikatsiyalari mavjud:

### 1. Trapetsiya usuli (Trapezoidal Rule):

$$\int_a^b f(x) dx \approx \Delta x \left[ \frac{f(x_0) + f(x_n)}{2} + \sum_{i=1}^{n-1} f(x_i) \right]$$

Bu usulda yoysimon qismlar hisobga olinadi va aniqlik to'g'ri to'rtburchak usulidan yaxshiroq bo'ladi.

### 2. Simpson usuli (Simpson's Rule):

$$\int_a^b f(x) dx \approx \Delta x \left[ \frac{f(x_0) + f(x_n)}{3} + \frac{4}{3} \sum_{i=1,3,5,\dots,n-1} f(x_i) + \frac{2}{3} \sum_{i=2,4,6,\dots,n-2} f(x_i) \right]$$

Simpson usuli parabola kesmalaridan foydalanadi va to'g'ri to'rtburchak va trapetsiya usullariga nisbatan aniqroq natija beradi.

Zaruriy aniqlikni ta'minlash uchun qadamni ( $n$ ) maqbul tanlash muhim ahamiyatga ega. Qadamni tanlashda quyidagilarni hisobga olish lozim:

1. **Xatolik baholash:** Tanlangan qadamga muvofiq yig'indi bilan haqiqiy integral o'rtasidagi farq (xatolik) ma'lum chegaradan oshmasligi kerak.
2. **Integralni ifodalovchi funksiya:** Funksiyaning o'zgarishi, o'zgarish tezligi va hokazolar aniqlik darajasiga ta'sir qiladi.
3. **Masshtab:** Integral oralig'i kengayishi bilan qadamlarning soni oshishi kerak.

Aniqlik darajasi va hisoblash samaradorligi o'rtasidagi muvozanatni topish maqsadida, algoritmik yondashuvlar yoki empirik tajribalar yordamida qadamni maqbul tanlash mumkin. Bu masalada optimal yechimni topish nisbatan murakkab hisoblanadi.

Taqribiy integral hisoblash usullari, integralni taqribiy hisoblash uchun ishlatiladigan har xil usullardir. Bu usullar matematik hisoblashni oddiy va tezlashtirish uchun foydalaniladi. Quyidagi taqribiy integral hisoblash usullaridan ba'zilarini ko'rib chiqamiz:

1. **Taqribiy to'g'ri birlashtirish (Rectangular Approximation):** Bu usulda, integrallangan maydon to'g'ri birlashtirilgan kiritish orqali tezlashtiriladi. Maydonni kiritish esa kvadrat, to'g'ri to'rtburchak yoki boshqa shaklda bo'lishi mumkin. To'g'ri birlashtirilgan usulga kiritilgan ta'limotlar soni o'sish bilan integralning aniqligi oshadi.

2. **Taqribiy chapboshlash (Trapezoidal Approximation):** Bu usulda, integralni yalpi birlashtirilgan maydon yordamida yuzasini topish uchun to'g'ri chiziqlar bilan to'g'ri birlashtirilgan maydonning yuzasini hisoblash. Bu usul, to'g'ri birlashtirilgan usulga nisbatan aniqlikni oshiradi.
3. **Simpson formulasi:** Simpson formulasi, integralni aniqlik bilan hisoblash uchun chiziqli va katranli funksiyalarni ishlatadi. Bu usulga ko'rsatilgan funksiya qo'shimcha ma'lumotlar yordamida ikki katran, to'rtburchak yoki boshqa shaklda hisoblanadi.
4. **Midpoint Approximation:** Bu usulda, integralni taqribiy hisoblash uchun o'lchov markazidan yarim uyalar bilan ta'limot olinadi. Taqribiy hisoblashning eng oson va ishonchli usullaridan biri hisoblanadi.
5. **Monte-Karlo integrallash:** Bu usul statistik nazorat asosida amalga oshiriladi. Ushbu usulda, funksiyaning tesadufiy nuqtalarida qiymatlar olinadi va ularning o'rtacha qiymati integralning qiymati hisoblanadi.

Zaruriy aniqlikni ta'minlovchi qadamni tanlashda, barcha taqribiy usullarning, yechimni chiqarishda iste'mol qilingan nusxalari hisobga olingan o'zgaruvchilar soni, xususan, integralni bir necha to'plamga bo'lish, va uning burchaklarining soni keng ko'payadi. Chunki taqribiy usullar asosida hisoblash bir muddatning o'rtacha qiymatlarini hisoblashga asoslanadi, shuning uchun burchaklar sonining ko'payishi aniqlikni oshiradi.

Biroq, birinchi necha burchaklardan birini tanlash tezligi va hajmi o'zgaradi. Aniqlikni ta'minlovchi eng yaxshi usulni tanlash uchun, barcha faktorlarni, masalan, ushbu burchaklar soni, xususan, yuqori hisoblangan usulni tanlash kerak bo'lgan aniqlik darajasini ta'kiklash uchun hisobga oling. Ma'lumki, katta burchaklar soni oshganidek, hisoblash natijalari aniqlikka ega bo'ladi.

Agar hisoblash uchun tez va aniqlikni ta'minlovchi qadam tanlashni istasangiz, "Simpson formulasi" yoki "Midpoint Approximation" usullarini ko'rishishingiz mumkin, chunki ular hisoblash natijalarini aniqlik va tezlik bilan o'tkazishda yaxshi natija qiladi. Bunday qadamni tanlaganingizdan so'ng, aniqlikni ta'minlovchi eng optimal usulni tanlash uchun bir necha muddatlarda bu usullarni sinab ko'ring.

## S235 “Ajrat va hukmronlik qil” prinsipi bo'yicha ishlaydigan algoritmlarni loyihalash. Elementlar jamlanmasini biror belgi bo'yicha tartiblashtirish algoritmi

"Ajrat va hukmronlik" prinsipi bo'yicha ishlaydigan algoritmlar murakkab masalalarni yechishda keng qo'llaniladi. Ushbu printsipl asosida elementlar jamlanmasini biror belgi bo'yicha tartiblash algoritmi quyidagi tarzda amalga oshiriladi:

### 1. **Ajratish (Divide):**

- Masalani kichik, oson yechiluvchi qismlarga bo'lish.
- Masalan, elementlar jamlanmasini ikkita teng yoki deyarli teng qismga bo'lish.

### 2. **Hukmronlik (Conquer):**

- Har bir kichik qismdagi elementlarni biror belgi bo'yicha tartiblash.
- Masalan, har bir qismni alohida tartiblash.

### 3. **Birlashtirish (Combine):**

- Tartiblangan kichik qismlardagi elementlarni birlashtirish.
- Masalan, ikkita tartiblangan qismni birlashtirish.

Bunday yondashuvda asosiy g'oya masalani kichik, oson yechiluvchi qismlarga bo'lish va so'ng ularni birlashtirish orqali yechishdir. Bu usul, ayniqsa, o'lchamli, murakkab masalalarni yechishda juda samarali.

Elementlar jamlanmasini biror belgi bo'yicha tartiblash algoritmi shu yo'nalishda amalga oshiriladi:

1. **Ajratish:** Jamlanmani ikkita teng yoki deyarli teng qismga bo'lish.
2. **Hukmronlik:** Har bir qismdagi elementlarni tartiblash. Masalan, tezkor tartiblash (Quick Sort) yoki birlashtirish tartiblash (Merge Sort) algoritmlaridan foydalanish mumkin.
3. **Birlashtirish:** Tartiblangan qismlardagi elementlarni birlashtirib, yakuniy tartiblangan jamlanmani hosil qilish.

Bu algoritmnining samaradorligi, asosan, tartiblash algoritmlari tanlanishiga bog'liq. Tezkor tartiblash (Quick Sort) va birlashtirish tartiblash (Merge Sort) algoritmlarining  $O(n \log n)$  murakkablik darajasi bilan samarali hisoblanadi.

"Ajrat va hukmronlik" printsiptan foydalangan holda, murakkab masalalarni samarali hal qilish mumkin. Bunday yondashuvning asosiy afzalligi - murakkab masalani kichik, oson yechiluvchi qismlarga bo'lish va ularni mustaqil yechib, birlashtirish orqali umumiy yechimni topish imkoniyatidir.

"Ajrat va hukmronlik qil" (Divide and Conquer) prinsipi, masalani kichik qismlarga bo'lib ajratib, har bir qismni o'zaro hal qilish va natijada ulardan faqat eng yomon natijani olish prinsipi. Bu prinsip bilan ishlaydigan algoritmlar keng qo'llaniladi va bir nechta muhim algoritmik masalalar uchun asosiy qo'llanma usuli hisoblanadi.

Elementlar jamlanmasini biror belgi bo'yicha tartiblashtirish algoritmi "Ajrat va hukmronlik qil" prinsipi bilan yaxshi bir misol hisoblanadi. Ushbu algoritmi quyidagi ko'rinishda yozilishi mumkin:

1. Elementlar jamlanmasini ikki qisimga ajratamiz.
2. Har bir qismni o'zini bitta elementdan iborat bo'lgan qismlarga bo'lib ajratamiz.
3. Har bir qismni alohida tartiblashtiramiz.
4. So'ng, ikki qismini birlashtiramiz va natijada barcha elementlar tartibga solingan bo'ladi.

## S236 Bog'langan graflarda marshrutlar, ularni narxi(masofasi) bo'yicha baholash. Xasis algoritmlar. Eng qisqa marshrutni aniqlash algoritmi. Uni variantlar soni bo'yicha hajmini baholash.

Bog'langan graflar (Connected Graphs) kontekstida, marshrut narxini (masofasini) baholash va eng qisqa marshrut topish masalasi juda muhim amaliy ahamiyatga ega. Bunday masalalarni yechish uchun "xasis" (greedy) algoritmlar keng qo'llaniladi.

Eng qisqa marshrut aniqlash algoritmlari:

### 1. Dijkstra algoritmi:

- Eng mashhur va keng tarqalgan algoritm.
- Har bir tugundan boshqa barcha tugunlarga eng qisqa masofani topadi.
- Algoritm ishlash prinsipi:
  - Har bir tugunning eng yaqin qo'shni tugunlarigacha bo'lgan masofani hisoblash.
  - Topilgan qisqa masofalarni doimiy ravishda yangilash.
  - Ushbu jarayon barcha tugunlar uchun yakunlanguncha davom etadi.
- Murakkabligi:  $O(n^2)$  yoki  $O(E+n \log n)$  ( $E$  - yoqlar soni,  $n$  - tugunlar soni).

### 2. Bellman-Ford algoritmi:

- Manfiy og'irlikli yoqlarga ega graflar uchun mos keladi.
- Eng qisqa masofani topadi, ammo Dijkstra algoritmiga nisbatan sekinroq.
- Murakkabligi:  $O(n \cdot E)$ .

### 3. A algoritmi:\*

- Hevristik yondashuv asosida ishlaydi.
- Har bir tugundan boshlab, eng qisqa masofani baholaydi.
- Baholash funktsiyasi:
  - $F(n) = G(n) + H(n)$
  - $G(n)$  -  $n$  tugunigacha bo'lgan haqiqiy masofa
  - $H(n)$  -  $n$  tugunidan maqsadli tugunga bo'lgan baholangan masofa (hevristika)
- Murakkabligi:  $O(E + n \log n)$ .

Eng qisqa marshrut aniqlash algoritmlarining variantlar soni bo'yicha hajmini baholash:

- Dijkstra va Bellman-Ford algoritmlari har bir tugun uchun topilgan eng qisqa masofani saqlaydi, shuning uchun variantlar soni  $n$  (tugunlar soni) ga teng.
- A\* algoritmi hevristika funktsiyasiga bog'liq bo'lib, variantlar soni funktsiyaning tanlangan shaklidan kelib chiqadi. Masalan, Evklid masofa hevristigasi ishlatilsa, variantlar soni  $O(b^d)$  bo'ladi,  $b$  - tarmoq omili,  $d$  - maqsadli tugunga bo'lgan masofasi.

Xulosa qilib aytganda, Dijkstra, Bellman-Ford va A\* algoritmlarining har biri o'z xususiyatlariga ega bo'lib, turli sharoitlarda qo'llanilishi mumkin. Variantlar soni bo'yicha hajmni baholash esa algoritmlar tanlashi va ilovaning parametrlarini optimallashtirishda muhim ahamiyatga ega.

Bog'langan graflarda marshrutlar va ularni narxi (masofasi) bo'yicha baholash, xususan, eng qisqa marshrutni aniqlash algoritmi va uni variantlar soni bo'yicha hajmini baholash uchun bir nechta xususiy algoritmlar mavjud. Bu masalalar ham "Ajrat va hukmronlik qil" prinsipi asosida hal qilinadi. Quyidagi Algoritmlar bu masalalarni hal qilishda yordam beradi:

1. **Dijkstra algoritmi:** Dijkstra algoritmi, grafda berilgan boshlang'ich tug'unidan boshlab, barcha boshlang'ich tug'unlar uchun eng qisqa yo'ldan o'tish masofasini topadi. Bu algoritm keng qo'llaniladi va uning ishlash qoidalari oson tushuniladigan bo'lib, bu algoritm yordamida eng qisqa marshrutni aniqlash oson va tezlik bilan amalga oshiriladi.
2. **Floyd-Warshall algoritmi:** Floyd-Warshall algoritmi grafda barcha uchlar orasidagi eng qisqa yo'ldan o'tish masofalarini topadi. Bu algoritm yordamida barcha uchlar orasidagi eng qisqa marshrutlar aniqlanadi.
3. **Bellman-Ford algoritmi:** Bellman-Ford algoritmi ham Dijkstra algoritmi kabi, bir boshlang'ich tug'unidan boshlab, barcha boshlang'ich tug'unlar uchun eng qisqa yo'ldan o'tish masofasini topadi. Bu algoritm negativ ogirliqi (masofasi) bo'lgan yollarni ham qabul qiladi, ammo u eng qisqa marshrutni aniqlashda Dijkstra algoritmi bilan solishtiriladi.
4. **Variantlar soni bo'yicha hajmini baholash:** Marshrutlarni baholashda, boshqa xususiy algoritmlar, masalan, "Brute Force" usul, ishlatilishi mumkin. Ushbu usul grafda barcha yollar orasidagi masofalarni hisoblashni, keyin esa eng qisqa vaqt yoki masofaga ega bo'lgan yollarni aniqlashni amalga oshiradi.

## S237 Algoritmnlarni loyihalash. Algoritm korrekt va samaradorligini baholash. Kvadrat tenglama ildizlarini aniqlash algoritmi. Uchburchak yuzasi uchun Geron formulasi

Bu algoritm kvadrat tenglama ildizlarini aniqlash uchun binar qidiruvdan foydalanadi. Ushbu algoritm  $n$  soni uchun ildizning ya to'g'ri, ya ham past, yoki ya past bo'lishi mumkin. Agar ildiz to'g'ri bo'lsa, algoritm uni qaytaradi. Agar yo'q, ildiz past bo'lsa, past bo'lgan joyidan davom etadi.

Algoritm samaradorligini baholash uchun quyidagi imkoniyatlarni tekshirishimiz mumkin:

1. **To'g'ri ishlash:** Algoritm kiritilgan sonning kvadrat ildizini to'g'ri aniqlaydi.
2. **Samaradorlik:** Kvadrat ildizini aniqlash uchun har bir qadamda sonlar sonini yarimdan pastroq qiladi, shuningdek, oqimga bir necha marta yordam beradi. Ushbu algoritm barcha sonlar uchun o'nlik yorqinlikda ishlaydi, shuning uchun samaradorlikni baholash qiyin bo'ladi.

Algoritm loyihalash, uning korrektiligi va samaradorligini baholash juda muhim masalalardir. Kvadrat tenglamaning ildizlarini aniqlash algoritmi va Uchburchak yuzasi uchun Geron formulasi misolida ushbu jihatlarni ko'rib chiqamiz.

### Kvadrat tenglama ildizlarini aniqlash algoritmi:

1. **Kirish ma'lumotlari:** Kvadrat tenglamaning  $a$ ,  $b$  va  $c$  koeffitsientlari.
2. **Algoritm:**
  - Diskriminantni hisoblash:  $D = b^2 - 4ac$
  - Agar  $D > 0$  bo'lsa, ikkita haqiqiy ildiz bor:
    - $x_1 = (-b + \sqrt{D}) / (2a)$
    - $x_2 = (-b - \sqrt{D}) / (2a)$
  - Agar  $D = 0$  bo'lsa, bitta haqiqiy ildiz bor:
    - $x = -b / (2a)$
  - Agar  $D < 0$  bo'lsa, ikkita kompleks ildiz bor:
    - $x_1 = (-b + i\sqrt{-D}) / (2a)$
    - $x_2 = (-b - i\sqrt{-D}) / (2a)$
3. **Chiqish ma'lumotlari:** Aniqlangan ildizlar.

### Algoritm korrektiligi:

- Algoritm har doim to'g'ri ildizni topadi, chunki u kvadrat tenglamaning yechimlarini to'g'ri hisoblaydi.
- Algoritm har xil holatlarni ( $D > 0$ ,  $D = 0$ ,  $D < 0$ ) to'liq qamrab oladi.

### Algoritm samaradorligi:

- Asosiy hisoblash operatsiyalari: diskriminantni hisoblash, ildizlarni hisoblash.
- Murakkabligi:  $O(1)$ , chunki hisoblashlar doimiy sonli operatsiyalardan iborat.

- Katta o'lchamdagi kvadrat tenglamalarni ham tez va samarali yecha oladi.

### **Uchburchak yuzasi uchun Geron formulasi:**

Uchburchakning a, b va c tomonlari berilgan bo'lsa, uning yuzasini hisoblash uchun Geron formulasi quyidagicha:

1. **Kirish ma'lumotlari:** Uchburchakning a, b va c tomonlari.
2. **Algoritm:**
  - Yarim perimetrni hisoblash:  $s = (a + b + c) / 2$
  - Yuzani hisoblash:  $S = \sqrt{s * (s - a) * (s - b) * (s - c)}$
3. **Chiqish ma'lumotlari:** Uchburchak yuzasi.

### **Algoritm korrektiligi:**

- Geron formulasi uchburchakning yuzasini to'g'ri hisoblaydi va hamma holatlar uchun ishlaydi.
- Formulaning asosligi geometrik va matematikaviy jihatdan isbotlangan.

### **Algoritm samaradorligi:**

- Asosiy hisoblash operatsiyalari: yarim perimetrni hisoblash, ildizni hisoblash.
- Murakkabligi:  $O(1)$ , chunki hisoblashlar doimiy sonli operatsiyalardan iborat.
- Katta o'lchamdagi uchburchaklar uchun ham tez va samarali ishlaydi.

Xulosa qilib aytganda, algoritm loyihalash, uning korrektiligi va samaradorligini baholash muhim masalalar bo'lib, ular algoritm samaradorligini oshirish va uning ishonchliligini ta'minlash uchun juda muhimdir. Kvadrat tenglama ildizlarini aniqlash va Uchburchak yuzasini hisoblash algoritmlari korrekt va samarali bo'lib, turli sharoitlarda qo'llanilishi mumkin.



## **S238 Algoritmning murakkabligini statik va dinamik o'lchovlari. Vaqt va xotira xajmi bo'yicha qiyinchiliklar.**

Algoritmning murakkabligini baholash uchun statik va dinamik o'lchovlar ishlatiladi. Ularning vaqt va xotira hajmi bo'yicha qiyinchiliklari quyidagicha:

### **Statik O'lchovlar:**

- Algoritm murakkabligining statik o'lchovi - algoritmdagi operatsiyalar, sharoitlar, takrorlanishlar, rekursiya va boshqa strukturalarni tahlil qilish asosida aniqlanadi.
- Masalan, ikki sonni yig'ish algoritmining statik murakkabligi  $O(1)$  bo'ladi, chunki u faqat ikkita arifmetik amaldan iborat.
- Statik o'lchovlar algoritm tuzilishiga bog'liq bo'ladi va algoritmnı bajarish davomida o'zgarmaydi.
- Statik o'lchovlar algoritmning eng yomon holatidagi murakkabligini aniqlaydi.

### **Dinamik O'lchovlar:**

- Algoritm murakkabligining dinamik o'lchovi - algoritm bajarilishi jarayonida vaqt va xotira sarfi asosida aniqlanadi.
- Masalan,  $n$  elementli massivni saralash algoritmining dinamik murakkabligi  $O(n^2)$  bo'ladi, chunki u  $n^2$  ta solishtirish va almashtirish amallarini bajaradi.
- Dinamik o'lchovlar algoritmning bajarilish sharoitlariga bog'liq bo'ladi va algoritmnı bajarish jarayonida o'zgarishi mumkin.
- Dinamik o'lchovlar algoritm bajarilishining o'rtacha va eng yomon holatlarini aniqlaydi.

### **Vaqt va xotira hajmi bo'yicha qiyinchiliklar:**

- Vaqt murakkabligi - algoritm bajarilishi uchun kerak bo'lgan vaqt.
- Xotira murakkabligi - algoritm bajarilishi uchun kerak bo'lgan xotira hajmi.
- Statik o'lchovlar vaqt murakkabligining eng yomon holatini aniqlasa, dinamik o'lchovlar vaqt murakkabligining o'rtacha va eng yomon holatini aniqlaydi.
- Statik o'lchovlar xotira murakkabligining eng yomon holatini aniqlasa, dinamik o'lchovlar xotira murakkabligining o'rtacha va eng yomon holatini aniqlaydi.
- Vaqt murakkabligi algoritmning ishlash tezligini, xotira murakkabligi esa uning xotira sarfini ko'rsatadi.

Xulosa qilib aytganda, algoritmning murakkabligini baholash uchun statik va dinamik o'lchovlar qo'llaniladi. Statik o'lchovlar algoritmning tuzilishiga, dinamik o'lchovlar esa uning bajarilish sharoitlariga bog'liq bo'ladi. Vaqt va xotira murakkabliklari esa algoritmning ishlash tezligi va xotira sarfini aks ettiradi.

Algoritmning murakkabligini statik va dinamik o'lchovlaridan foydalanish, ularning vaqt va xotira xajmi bo'yicha qiyinchiliklarini aniqlash uchun juda foydali bo'ladi.

#### **1. Statik o'lchovlar:**

- **Kodning o'lchami:** Statik o'lchovlar, kodning uzunligi yoki sodir bo'lish mumkin bo'lgan xotira hajmini o'lchaydi. Bunda o'lchash uchun kodning qatorlar soni yoki o'lcham tushunchalari ishlatiladi.
- **Mos sharoitlar:** Statik o'lchovlar, algoritmnining qanday qilib ma'lum bir sharoitda ishlashi kerakligini aniqlashda yordam beradi. Misol uchun, bitta kiritilgan son uchun algoritmnining qanday qilib yorqin bo'lishi kerakligini aniqlash.

## 2. Dinamik o'lchovlar:

- **Vaqt o'lchami:** Dinamik o'lchovlar, dastur bajarilish vaqtini o'lchaydi. Bunda algoritmnining bajarilishi uchun sarflangan vaqt miqdori o'lchaydi. Bu algoritmlar vaqt taqsimoti va bajarilishi uchun qancha vaqt sarflanganligini aniqlash uchun juda foydali bo'ladi.
- **Xotira o'lchami:** Dinamik o'lchovlar, dastur tomonidan ishlatilayotgan xotira miqdorini o'lchaydi. Xususan, kiritilgan ma'lumotlar yoki dasturni bajarish uchun qancha xotira xajmi kerakligini aniqlash uchun foydalaniladi.

## Vaqt va xotira xajmi bo'yicha qiyinchiliklar:

### 1. Statik o'lchovlar:

- **Qiyinchiliklar:** Statik o'lchovlar, dasturni bajarish jarayonida aniqlanadigan to'liq vaqt va xotira xajmini belgilashda qiyinchiliklar tug'ishiga olib keladi. Chunki dastur ishga tushgandan keyin aniqlik bilan bilinadigan qanday vaqt va xotira xajmi talab qilinishi mumkin bo'lishi mumkin emas.
- **Xohish etiladigan natija:** Statik o'lchovlar, kodning o'zgaruvchanligini va mustaqil hisoblashni ta'minlaydi. Ammo vaqt va xotira xajmini aniqlashda vaqt xajmi va kiritilgan ma'lumotlar soni o'sishi bilan bog'liq bo'lishi mumkin.

### 2. Dinamik o'lchovlar:

- **Qiyinchiliklar:** Dinamik o'lchovlar, dastur ishga tushirilgandan so'ng aniqlik bilan aniqlik bilan vaqt va xotira xajmini o'lchashga yordam beradi. Ammo, bu o'lchovlar ham aniq vaqt va xotira xajmi hisoblash uchun muddat va xotira xajmi sarflanganligi uchun yorqin resurslarga ega bo'lishi mumkin.
- **Xohish etiladigan natija:** Dinamik o'lchovlar, dastur ishga tushirilganida ma'lumotlar va hajmi o'zgarishi mumkin bo'lgan holatlarda yaxshi natijalarni olishda yordam beradi. Ular dastur aniqlangan sharoitlarda yaxshi darajada ishlashga yordam beradi.

Shuningdek, algoritmlarning vaqt va xotira xajmi bo'yicha qiyinchiliklarini aniqlash uchun profiling vositalari va muhit emulatorlari ham foydalanilishi mumkin. Bu vositalar, dasturlarni to'g'ri ishlashi va o'zaro solishtirilish uchun yordam beradi.

## S239 Algoritmnlarni eng yomon va o'rtacha xolatlarda baholash.

Algoritmnlarni baholashda eng yomon va o'rtacha holatlarni quyidagicha tahlil qilish mumkin:

### Eng yomon holat (Worst-Case):

- Eng yomon holat algoritm bajarilishi uchun kerak bo'lgan eng katta vaqt yoki xotira resurslarini ifodalaydi.
- Eng yomon holatdagi vaqt va xotira murakkabligi algoritmning eng yuqori cheklanishini ko'rsatadi.
- Masalan,  $n$  elementli massivni saralash uchun eng yomon holatdagi vaqt murakkabligi  $O(n^2)$  ga teng, chunki eng yomon holat massivning elementlarini to'liq almashtirishni talab qiladi.
- Eng yomon holatdagi tahlil algoritm ishlashining eng yomon senariysini aniqlaydi va algoritm loyihalashtirilishi jarayoniga ta'sir ko'rsatadi.

### O'rtacha holat (Average-Case):

- O'rtacha holat algoritm bajarilishi uchun kerak bo'lgan o'rtacha vaqt yoki xotira resurslarini ifodalaydi.
- O'rtacha holatdagi vaqt va xotira murakkabligi algoritm bajarilishining ko'plab senariylari uchun o'rtacha qiymatini ko'rsatadi.
- Masalan,  $n$  elementli massivni saralash uchun o'rtacha holatdagi vaqt murakkabligi  $O(n \log n)$  ga teng, chunki ko'pincha massiv elementlari o'rtacha tarzda tarqalgan bo'ladi.
- O'rtacha holatdagi tahlil algoritmning bajarilishi uchun talab qilinadigan o'rtacha resurslarni aniqlaydi va algoritm loyihalashtirilishi jarayoniga ta'sir ko'rsatadi.

Eng yomon holat va o'rtacha holatni tahlil qilish algoritmning bajarilish xususiyatlarini to'liq tushunish uchun muhim.

Eng yomon holat algoritm bajarilishining eng yuqori chegarasini ko'rsatsa, o'rtacha holat esa bajarilishining o'rtacha qiymatini ifodalaydi. Ularning tahlili algoritm loyihasi va sifatini yaxshilash uchun foydalaniladi.

Algoritmnlarni eng yomon va o'rtacha xolatlarda baholash uchun, umuman, ularning murakkabligi, ishga tushirilish vaqti, yordamchi xotira hajmi, va chiqish o'lchami kabi bir nechta ko'rsatkichlarni qarash mumkin. Quyidagi qatorlar ko'rsatilgan algoritmning baholashida yordam berishi mumkin:

1. **Ishga tushirilish vaqti:** Agar algoritmning ishga tushirilish vaqti juda ko'p bo'lsa, bu algoritmni eng yomon xolat deb baholaymiz. Ishga tushirilish vaqti ko'paygan holda, algoritmning amalga oshishi uchun uzun vaqt talab qiladi va bu juda yomon xolat bo'ladi.

2. **Yordamchi xotira hajmi:** Agar algoritmi ko'p xotira sarflaydigan bo'lsa, bu ham uning yomon xolatda ishlayotganligi bilan bog'liq bo'ladi. Chunki katta xotira sarflanishi tizim uchun katta qiyinchilik yaratadi.
3. **Chiqish o'lchami:** Algoritmdan olish kerak bo'lgan natijaning kengligi va o'lchami muhimdir. Agar algoritmi chiqish o'lchami juda katta bo'lsa, bu uning yomon xolatda ishlayotganligini ko'rsatadi.
4. **Murakkabligi:** Agar algoritmi murakkab bo'lsa, bu uning ishga tushirilish vaqti va yordamchi xotira hajmi uchun muammo yaratadi. Qo'shimcha murakkablik tizimga qo'shimcha boshqa kamchiliklar kiritishi mumkin.
5. **Ishga tushirilish muhiti:** Agar algoritmi xususan belgilangan muhitda yaxshi ishlamasa, bu uning yomon xolatda ishlayotganligini anglatadi. Masalan, qattiq qidiruv (brute-force) algoritmi barcha belgilangan ma'lumotlarni tekshirishi kerak bo'lsa, shu muhitda ishlay olmaydi.

## S240 Algoritmnlarni vaqt va hajmiy murakkabligini baholashda tekis va logarifmik solishtirma mezonlar.

Algoritmnlarni vaqt va hajmiy murakkabligini baholashda tekis va logarifmik solishtirma mezonlardan foydalaniladi:

### **Tekis Solishtirma Mezonlar:**

- Tekis solishtirma mezonlarda algoritm murakkabligini baholashda lineap, kvadratik, kublik va boshqa darajali funksiyalardan foydalaniladi.
- Masalan, massivni saralash uchun ishlatilgan algoritm murakkabligi  $O(n^2)$  bo'lsa, bu algoritm kvadratik murakkablikka ega deyiladi.
- Tekis solishtirmalar algoritmning eng yomon holati uchun qo'llaniladi.
- Tekis solishtirmalar oson tushuniladi va algoritmlar taqqoslashda keng qo'llaniladi.
- Lekin tekis solishtirmalar algoritm murakkabligini to'liq aks ettira olmaydi.

### **Logarifmik Solishtirma Mezonlar:**

- Logarifmik solishtirma mezonlarda algoritm murakkabligini baholashda logarifmik funksiyalardan foydalaniladi.
- Masalan, ikili qidiruv algoritmining murakkabligi  $O(\log n)$  bo'lsa, bu algoritm logarifmik murakkablikka ega deyiladi.
- Logarifmik solishtirmalar algoritm bajarilishidagi o'rtacha holatni aks ettiradi.
- Logarifmik solishtirmalar algoritm murakkabligini aniq aks ettiradi, lekin ularni tushunish va taqqoslash murakkabroq.
- Logarifmik solishtirmalar umumiy holda algoritm bajarilishining xarakterlari va tendensiyalarini yaxshiroq tafsirlay oladi.

Algoritmnlarni baholashda quyidagi solishtirma mezonlar qo'llaniladi:

- $O(1)$  - Doimiy murakkablik
- $O(\log n)$  - Logarifmik murakkablik
- $O(n)$  - Lineap murakkablik
- $O(n \log n)$  - Lineap logarifmik murakkablik
- $O(n^2)$  - Kvadratik murakkablik
- $O(n^3)$  - Kublik murakkablik

Tekis va logarifmik solishtirma mezonlari bir-birini to'ldiradi va algoritm murakkabligini baholash uchun kompleks tarzda qo'llaniladi. Ularning afzalliklari va kamchiliklarini hisobga olgan holda algoritmlar samaradorligini tahlil qilish mumkin.

Algoritmnlarni vaqt va hajmiy murakkabligini tekis va logarifmik solishtirma mezonlari bilan baholash, ularning murakkabligi va amalga oshirishning qanchalik resurslar talab qilishi haqida ko'proq ma'lumot beradi. Bu mezonlar algoritmnlarni ta'til o'rganish uchun juda foydali bo'lib, ularning samaradorligini aniqlash va optimallashtirishda yordam beradi.

### **1. Tekis solishtirma mezonlari:**

- **Ishga tushirilish vaqti:** Tekis solishtirma mezonlari, algoritmlar ishga tushirilish vaqti yoki o'zgarishlarning qanday darajada bog'liqligini aniqlaydi. Bu mezonlar algoritmlar orasida to'g'ri taqqoslash uchun juda foydali bo'lib, boshqaruvchi tomonidan algoritmlar ustida boshqarishni osonlashtiradi.
- **Xotira xajmi:** Algoritmlar o'z vaqti davomida qanday qilib katta yoki kichik xotira sarflaydi, shu tarzda xotira xajmini aniqlashga yordam beradi.

## 2. Logorifmik solishtirma mezonlari:

- **Ishga tushirilish vaqti:** Logorifmik solishtirma mezonlari, algoritmlar ishga tushirilish vaqti bilan bog'liq bo'lgan muammo vaqtda qanday o'zgarishlarni ko'rsatish uchun ishlatiladi. Bu mezonlar barcha ma'lumotlar soni o'sishini yoki o'zgarishlar sonini belgilashda yordam beradi.
  - **Xotira xajmi:** Logorifmik solishtirma mezonlari, algoritmlarning katta va kichik ma'lumotlar uchun qanday qilib xotira sarflaydi, shu tarzda xotira xajmini aniqlashda yordam beradi.
3. Bu kod, algoritmlarni tekis va logorifmik solishtirma mezonlaridan foydalanib baholashda yordam beradi. Natijalar olingandan so'ng, algoritmlar uchun vaqt va xotira xajmini chiqarib beradi. Natijalar olingandan so'ng, algoritmlar uchun vaqt va xotira xajmini chiqarib beradi. Natijalar yordamchi xususiyatlari bo'yicha algoritmlar haqida fikr berishda yordam beradi.
  4. Bu code ustida o'zgarishlar qilindi, yaqinroqda algoritmlarni tekis va logorifmik solishtirma mezonlari boyicha baholash yordam berishni o'zgartirishni iloji bor edi. Agar bu code sizga to'g'ri kelmasa, yoki qo'shimcha savollaringiz bo'lsa, menga xabardor qiling!

## S241 To'plamlarda qisqartma akslantirishlar. Ularga va amaliy tatbiqlarga misollar.

To'plamlarda qisqartma akslantirishlariga quyidagi misollarni keltirish mumkin:

### 1. Bitli Bayroqcha (Bit Flag):

- Bu qisqartma aslida to'plamlarni aks ettirish uchun ishlatiladigan ikkilik yoki mantiqiy operatsiyalardan foydalanadi.
- Har bir element bir bitto'plamda aks ettiriladi va ularning holati 0 yoki 1 qiymatida saqlanadi.
- Masalan, 8 ta elementi bo'lgan to'plamni 1 buta songa aylantirish va operatsiyalar bajarish mumkin.
- Bitli bayroqchalardan foydalanish xotirani tejash va to'plamlar ustida tez bajariluvchi operatsiyalarni amalga oshirish uchun qo'l keladi.

### 2. Bitli Vektor (Bit Vector):

- Bu bitli bayroqchalar asosida yaratilganto'plamlar aks ettirilishining bir turi.
- Bitli vektor to'plamni har bir elementi uchun 1 bitdan iborat bo'lgan massiv orqali aks ettiradi.
- Massivdagi indekslar to'plam elementlarini belgilaydi va mos bit qiymati 0 yoki 1 bo'ladi.
- Bitli vektorlardan foydalanish to'plamlar ustidagi amallarni tez bajarish imkonini beradi.

### 3. Siqilgan Bit Vektor (Compressed Bit Vector):

- Bu bitli vektorlarni yanada samarali saqlay oladigan usuldir.
- Siqilgan bit vektor faqat 1 qiymatli bitlarni saqlab, 0 qiymatli bitlarni olib tashlaydi.
- Bu yechim xotira sarfini kamaytirib, to'plamlar ustidagi amallarni tezlashtiradi.
- Siqilgan bit vektorlardan foydalanish katta o'lchamli to'plamlar uchun qulay.

To'plamlarni aks ettirish uchun qisqartma akslantirishlardan foydalanish quyidagi amaliy sohalarda qo'l keladi:

- Korrelyatsiya va regressor tahlili
- Mantiqiy qurilmalar dizayni
- Xabarlar filtrlanishi
- Tasvir ob'yektlarini aniqlash
- Ma'lumotlar siqish
- Kriptografiya

Qisqartma akslantirishlar xotira sarfini kamaytirish va to'plamlar ustidagi amallarni tezlashtirish imkonini beradi. Ular samarali va tejamkor algoritmlar yaratish uchun keng qo'llaniladi.



To'plamlarda qisqartma akslantirishlar (summarization shortcuts) to'plamdagi ma'lumotlarni kichik qismlarga ajratib, ulardan umuman ma'lumot olishda yoki umuman qismlarni ta'riflashda yordam beradigan usullardir. Bu, ma'lumotlar bilan ishlashda va ma'lumotlar tushunchalarini ta'riflashda juda foydali bo'ladi. Quyidagi misollar sizga bu tushunchani kuchaytirishga yordam beradi:

### 1. Misol 1: O'quvchilar soni statistikasi:

Berilgan to'plamdagi o'quvchilar sonini toping:

```
python
Копировать код
students = [85, 90, 92, 88, 75, 80, 95, 78, 82, 87]
total_students = sum(students)
print("O'quvchilar soni:", total_students)
```

Bu kodda students degan ro'yxatni summa funktsiyasi yordamida yig'indi. Natija sifatida o'quvchilar sonini chiqaradi.

### 2. Misol 2: Juft va toq sonlar sonini hisoblash:

Berilgan ro'yxatda nechta juft va nechta toq son borligini hisoblang:

```
python
Копировать код
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_count = sum(1 for num in numbers if num % 2 == 0)
odd_count = sum(1 for num in numbers if num % 2 != 0)
print("Juft sonlar soni:", even_count)
print("Toq sonlar soni:", odd_count)
```

Bu kodda numbers degan ro'yxatda nechta juft va nechta toq sonlar borligini hisoblash uchun qisqartma akslantirish ishlatilgan. Natijalarni ekranga chiqaradi.

### 3. Misol 3: Maksimal va minimal qiymatlarni aniqlash:

Berilgan ro'yxatda maksimal va minimal qiymatlarni aniqlang:

```
python
Копировать код
numbers = [10, 25, 14, 36, 42, 18, 5]
max_value = max(numbers)
```

```
min_value = min(numbers)
print("Eng katta qiymat:", max_value)
print("Eng kichik qiymat:", min_value)
```

Bu kodda numbers ro'yxatidagi eng katta va eng kichik qiymatlarni aniqlash uchun qisqartma akslantirish ishlatilgan. Natijalarni ekranga chiqaradi.

#### 4. **Misol 4: Juft sonlar yig'indisini hisoblash:**

Berilgan ro'yxatdagi juft sonlar yig'indisini toping:

```
python
Копировать код
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum = sum(num for num in numbers if num % 2 == 0)
print("Juft sonlar yig'indisi:", even_sum)
```

Bu kodda numbers ro'yxatidagi juft sonlar yig'indisini topish uchun qisqartma akslantirish ishlatilgan. Natijani ekranga chiqaradi.

#### 5. **Misol 5: Ro'yxatdagi sonlarning darajasini hisoblash:**

Berilgan ro'yxatdagi har bir sonni uning kvadratini hisoblash:

```
python
Копировать код
numbers = [1, 2, 3, 4, 5]
squares = [num ** 2 for num in numbers]
print("Sonlar kvadratlari:", squares)
```

Bu kodda numbers ro'yxatidagi sonlar kvadratini hisoblash uchun qisqartma akslantirish ishlatilgan. Natijani ekranga chiqaradi.

Bu misollar yordamida to'plamlarda qisqartma akslantirishlarni yaxshiroq tushunishingiz mumkin. Ular umuman qanday qilib ma'lumotlarni kichik qismlarga ajratish uchun ko'p marta foydalaniladi.

Bu misollarda, to'plamlarda qisqartma akslantirishlarini, ma'lumotlar to'plamlarini kichik qismlarga ajratish uchun ko'p foydalanilgan usullarni ko'rdim. Agar boshqa misollarda yordam kerak bo'lsa, menga xabardor bo'ling!

## S242 Chiziqli algebraik tenglamalar sistemalarini yechishda oddiy itarasiya va Zeydal usullari, ularning yaqinlashish shartlari

Chiziqli algebraik tenglamalar sistemalarini yechishda qo'llaniladigan oddiy iteratsiya va Zeydal usullari quyidagicha:

### **Oddiy Itarasiya Usuli:**

- Bu usulda tenglamalar sistemasi quyidagi shaklga keltiriladi:  
 $x = f(x, y, z, \dots)$   
 $y = g(x, y, z, \dots)$   
 $z = h(x, y, z, \dots)$
- Dastlabki taxminiy yechim tanlanadi va quyidagi algoritm orqali yangi yechimlar hisoblanadi:  
 $x^{(k+1)} = f(x^{(k)}, y^{(k)}, z^{(k)}, \dots)$   
 $y^{(k+1)} = g(x^{(k+1)}, y^{(k)}, z^{(k)}, \dots)$   
 $z^{(k+1)} = h(x^{(k+1)}, y^{(k+1)}, z^{(k)}, \dots)$
- Bu jarayon ma'lum bir aniqlik yoki maksimal iteratsiya soni erishilguncha davom etadi.
- Oddiy iteratsiya usuli konvergent bo'lishi uchun sistemaning diagonal ustunlarga nisbatan qat'iy dominantligi shart.

### **Zeydal Usuli:**

- Bu usulda tenglamalar sistemasi quyidagi shaklga keltiriladi:  
 $x = f(y, z, \dots)$   
 $y = g(x, z, \dots)$   
 $z = h(x, y, \dots)$
- Dastlabki taxminiy yechim tanlanadi va quyidagi algoritm orqali yangi yechimlar hisoblanadi:  
 $x^{(k+1)} = f(y^{(k)}, z^{(k)}, \dots)$   
 $y^{(k+1)} = g(x^{(k+1)}, z^{(k)}, \dots)$   
 $z^{(k+1)} = h(x^{(k+1)}, y^{(k+1)}, \dots)$
- Bu jarayon ma'lum bir aniqlik yoki maksimal iteratsiya soni erishilguncha davom etadi.
- Zeydal usuli oddiy iteratsiya usulidan tezroq yaqinlashadi va kam xotira sarflaydi.
- Zeydal usuli konvergent bo'lishi uchun sistemaning diagonal ustunlarga nisbatan qat'iy dominantligi shart.

### **Yaqinlashish Shartlari:**

- Yuqoridagi usullarning yaqinlashishi uchun quyidagi shartlar bajarilishi kerak:
  1. Sistem chiziqli algebraik tenglamalar sistemasi bo'lishi lozim.
  2. Sistema diagonal ustunlarga nisbatan qat'iy dominant bo'lishi kerak.
  3. Dastlabki taxminiy yechim etarlicha yaqin bo'lishi shart.

- Agar bu shartlar bajarilsa, algoritm ma'lum bir aniqlik yoki maksimal iteratsiya soni erishilganda to'g'ri yechimga yaqinlashadi.
- Zeydal usuli oddiy iteratsiya usulidan aniqroq va tezroq konvergentdir.

Oddiy iteratsiya va Zeydal usullari chiziqli algebraik tenglamalar sistemalarini yechishda keng qo'llaniladi. Ular samarali va tez hisoblanadigan algoritmlar bo'lib, turli sohalarda qo'l keladi.

Chiziqli algebraik tenglamalar sistemalarini yechish uchun oddiy itarasiya (iterative) va Zeydal usullari (Gauss-Seidel method) keng qo'llaniladigan usullardir. Bu usullar sistemalarni hisoblashda juda samarali bo'lib, ularga qo'shimcha parametr sifatida yaqinlashish shartlari ham kerak bo'ladi.

### **Oddiy Itarasiya (Iterative Method):**

Oddiy itarasiya usulida chiziqli algebraik tenglamalar sistemalarini yechish uchun har bir o'zgaruvchini o'zining oldingi qiymati asosida yangilash (iterate) qilinadi. Bu usul sodda va oson, ammo qo'shimcha parametr - yaqinlashish shartlari -ni qo'llab-quvvatlaydi.

Oddiy itarasiya usulining yaqinlashish shartlari quyidagilardir:

1. Sistemalarning qanoatlari (iteration) ko'p marta bajarilishi kerak. Qanoatlarning soni, belgilangan xatolik miqdoriga va qiymatlarning qanday tez o'zgarishi kerakligiga bog'liq.
2. Yaqinlashish sharti: Itarasiya jarayonida har bir o'zgaruvchining yangi qiymati, oldingi qiymatidan yaqinroq bo'lishi kerak.

### **Zeydal usuli (Gauss-Seidel Method):**

Zeydal usuli ham chiziqli algebraik tenglamalar sistemalarini yechish uchun juda ommabop va samarali usuldir. Bu usulda, har bir o'zgaruvchi uchun yangi qiymat aniqlashda, boshqa o'zgaruvchilarning oldingi qiymatlari hisobga olinadi. Natijada, bu usul tezroq konvergirovchi hisoblash jarayonini ta'minlaydi.

Zeydal usulining yaqinlashish shartlari quyidagilardir:

1. Sistemalarning qanoatlari (iteration) ko'p marta bajarilishi kerak. Bu son, belgilangan xatolik miqdoriga va qiymatlarning qanday tez o'zgarishi kerakligiga bog'liq.
2. Yaqinlashish sharti: Har bir o'zgaruvchining yangi qiymati, oldingi qiymatlari bilan hisoblangan natijaga yaqinroq bo'lishi kerak.

Chiziqli algebraik tenglamalar sistemasini yechishda oddiy itarasiya (iterative) va Zeydal (Gauss-Seidel) usullari juda ommabop va samarali usullardir. Ularga qo'shimcha parametr sifatida yaqinlashish shartlari (convergence conditions) kerak bo'ladi. Quyidagi misollarda oddiy itarasiya va Zeydal usullarining qanday ishlayotganligi keltirilgan:

### 1. Oddiy Itarasiya (Iterative) Usuli:

Oddiy itarasiya usuli matritsa tenglamalar sistemalarini quyidagi tartibda yechish uchun ishlatiladi:

1. Boshlang'ich taxminlar (initial guesses) bilan boshlaymiz.
2. Har bir o'zgaruvchining yangi qiymatini, o'zining oldingi qiymati asosida yangilaymiz.
3. Eski va yangi qiymatlar orasidagi farq (tolerance) xuddi kichik bo'lsa, yoki belgilangan maksimal qadam soniga erishilgan bo'lsa, jarayonni to'xtatamiz.

Oddiy itarasiya usulining yaqinlashish shartlari quyidagilardir:

- **Konvergiya:** Sistemadagi eng katta eigenvaqt chiziqli aljabraviy tenglamalar sistema (diagonal dominanta) bo'lishi kerak.
- **Yaqinlashish sharti:** Har bir o'zgaruvchi uchun yangi qiymat, oldingi qiymati bilan taqriban bir xil bo'lishi kerak.

### Zeydal (Gauss-Seidel) Usuli:

Zeydal usuli oddiy itarasiya usuli bilan o'xshash ravishda ishlaydi, lekin bu usulda o'zgaruvchilarning yangi qiymati aniqlanishida eski qiymatlaridan foydalaniladi. Bu usul o'zgaruvchilarning tez konvergirovchi o'zgaruvchilarga qaraganda tezroq konvergirovchi o'zgaruvchilarni hisoblashda ishlatiladi.

Zeydal usulining yaqinlashish shartlari quyidagilardir:

- **Konvergiya:** Sistemadagi eng katta eigenvaqt chiziqli aljabraviy tenglamalar sistema (diagonal dominanta) bo'lishi kerak.
- **Yaqinlashish sharti:** Har bir o'zgaruvchining yangi qiymati, oldingi qiymatlaridan foydalanilishi kerak.

## S243 Determinantlarni hisoblash uchun bajariladigan amallar sonini baholash.

Determinantlarni hisoblash uchun bajariladigan amallar sonini baholashda quyidagi asosiy holatlarni ko'rib chiqish mumkin:

1.  **$n \times n$  o'lchamdagi determinantlar uchun:**

- Determinantni hisoblash uchun  $n!$  ta qarama-qarshi juftliklar ko'paytmasi hisoblash kerak.
- Har bir juftlik uchun  $n-1$  ta arifmetik amal bajarish kerak.
- Shunga ko'ra,  $n \times n$  o'lchamdagi determinant hisoblash uchun jami  $n! \cdot (n-1)$  ta amal bajariladi.

2.  **$3 \times 3$  o'lchamdagi determinantlar uchun:**

- $3 \times 3$  determinant hisoblash uchun 6 ta qarama-qarshi juftlik ko'paytmasi hisoblanadi.
- Har bir juftlik uchun 2 ta arifmetik amal bajariladi.
- Jami  $6 \cdot 2 = 12$  ta amal bajariladi.

3.  **$2 \times 2$  o'lchamdagi determinantlar uchun:**

- $2 \times 2$  determinant hisoblash uchun 2 ta qarama-qarshi juftlik ko'paytmasi hisoblanadi.
- Har bir juftlik uchun 1 ta arifmetik amal bajariladi.
- Jami  $2 \cdot 1 = 2$  ta amal bajariladi.

Umumiy holda,  $n \times n$  o'lchamdagi determinant hisoblash uchun bajarilishi kerak bo'lgan amallar soni  $O(n!)$  ga teng bo'ladi. Bu juda tez ortib boradigan funksiya bo'lib, katta o'lchamdagi determinantlar hisoblashda juda katta hisoblash qiyinchiliklariga olib keladi.

Shuning uchun, determinantlarni hisoblashda rekursiv yoki boshqa samarali algoritmlardan foydalanish muhim ahamiyatga ega. Bu algoritmlar determinantlar hisoblashini ancha tezlashtiradi va katta o'lchamdagi masalalarni hal qilish imkonini beradi.

Determinantlarni hisoblash uchun bajariladigan amallar sonini baholash, bajariladigan amallarning kiritilgan matritsaga qarab qancha ishchi va xotira talab qilishi lozimligini aniqlashni o'z ichiga oladi. Matritsa katta bo'lishi, katta sonli matritsa yoki murakkablik, xotira talab etadigan bajariladigan amallar sonini oshiradi. Quyidagi qoidalarga e'tibor bering:

1. **Matritsaning kattaligi:** Matritsaning kattaligi o'zgaruvchilarni hisoblash uchun to'g'ri koeffitsiyentlarni aniqlashda talab qiladi. Matritsalar katta bo'lganligi, amallar sonini oshirib boradi.
2. **Matritsaning murakkabligi:** Agar matritsa murakkab bo'lsa, hisoblash uchun kerak bo'lgan o'lchovlarning soni oshadi. Murakkab matritsalar uchun amallar soni ko'payadi.

3. **Kiritilgan amalni qanday bajarilishi:** Matritsada ishlatiladigan bajariladigan amallar sonini qo'llab-quvvatlaydigan muhim ma'lumotlardan biri, uning bajarilishi uchun qancha marta o'zgaruvchilarni hisoblashni talab qilishi. Agar amal murakkab bo'lsa va kattalik bo'lsa, amallar soni qo'shilib boradi.

## S244 Chiziqli algebraik tenglamalar sistemalarini aniq yechish uchun sarflanadigan amallar sonini baholash.

Chiziqli algebraik tenglamalar sistemalarini aniq yechish uchun sarflanadigan amallar sonini baholash uchun quyidagi asosiy yondashuv va natijalar e'tiborga olinadi:

1.  **$n \times n$  o'lchamdagi sistemalar uchun:**

- Aniq yechish uchun eng samarali usul Gauss-Jordan eliminatsiya usuli hisoblanadi.
- Gauss-Jordan eliminatsiya usulida quyidagi amallar bajariladi:
  - $n$  ta tenglamadan boshlab, har biri uchun  $n - i + 1$  ta amal bajariladi.
  - Jami  $n * (n + 1) / 2$  ta amal bajariladi.
- Shunga ko'ra,  $n \times n$  o'lchamdagi sistema uchun jami  $O(n^2)$  ta amal bajariladi.

2.  **$3 \times 3$  o'lchamdagi sistemalar uchun:**

- $3 \times 3$  sistema uchun Gauss-Jordan eliminatsiya usulida  $6 + 5 + 4 = 15$  ta amal bajariladi.

3.  **$2 \times 2$  o'lchamdagi sistemalar uchun:**

- $2 \times 2$  sistema uchun Gauss-Jordan eliminatsiya usulida  $3 + 2 = 5$  ta amal bajariladi.

Umumiy holda,  $n \times n$  o'lchamdagi chiziqli algebraik tenglamalar sistemasini aniq yechish uchun sarflanadigan amallar soni  $O(n^2)$  ga teng bo'ladi. Bu,  $n$  qancha katta bo'lsa, sistemani yechishga sarflanadigan vaqt va xotira ham shuncha ko'p bo'lishini anglatadi.

Shuning uchun, katta o'lchamdagi sistemalarni yechishda qo'shimcha optimallashtirish usullaridan foydalanish zarur bo'ladi. Bunga misol tariqasida, LU yoki Cholesky dekompozitsiya usullari, iteratsion usullar va shu kabilarni ko'rsatish mumkin.

Chiziqli algebraik tenglamalar sistemalarini aniq yechish uchun sarflanadigan amallar sonini baholash, bajariladigan amallar sonini hisoblashda xotira va vaqtga qancha ehtiyoj bo'lishini aniqlashga yordam beradi. Bu, bir algoritmnı bajarish va to'g'ri javobni topish uchun sarflanadigan resurslarning qanday qilib optimallashtirilishi kerakligini tushunishga yordam beradi.

Chiziqli algebraik tenglamalar sistemalarini aniq yechish uchun sarflanadigan amallar sonini baholash uchun quyidagi qoidalarga e'tibor bering:

1. **Matritsaning kattaligi:** Matritsaning o'lchamini oshirish, amallar sonini ko'paytiradi.
2. **Murakkablik darajasi:** Murakkablik darajasi, amallar sonini oshiradi.



3. **Har bir o'zgaruvchi uchun ishlovchi amallar soni:** Har bir o'zgaruvchi uchun bajariladigan amallar soni, sistemalarni yechish jarayonida qancha ishchi amallar qo'llanilishi kerakligini ko'rsatadi.

## S245 Statistik axborotlarni qayta ishlashda va prognoz masalalarida dinamik dasturlash usullari

Statistik axborotlarni qayta ishlashda va prognoz masalalarida dinamik dasturlash usullari keng qo'llaniladi. Quyida ushbu usullarning asosiy xususiyatlari va qo'llanilishi haqida ma'lumot berilgan:

### 1. **Dinamik dasturlash:**

- Murakkab masalalarni kichik, yechimi oson bo'lgan bosqichlarga bo'lish va ularni ketma-ket hal qilish prinsipiga asoslangan.
- Statistik prognoz va qaror qabul qilish masalalarida keng qo'llaniladi, chunki ko'p hollarda optimal yechim uchun rekursiv formulalar olinishi mumkin.
- Masalaning optimal yechimini topish uchun zaxiralangan (memorized) qiymatlardan foydalanish.
- Masalan, Markov zanjirlarini modellashtirish, maqsadli funktsiyani maksimallashtirish, vaqt qatorlarini prognozlash.

### 2. **Xidma (Viterbi) algoritmi:**

- Dinamik dasturlash asosida ishlaydigan optimal yechimni topish algoritmi.
- Maxfiy Markov modellaridan foydalanib nutq tanish, quvonch/kayfiyat aniqlanishi kabi masalalarni hal qilishda qo'llaniladi.
- Qisman kuzatilgan Markov jarayonlarida optimal holat ketma-ketligini tiklash imkonini beradi.

### 3. **Kalman filtrlash:**

- Chiziqli holat modellarida (masalan, vaqt qatorlari) holatlarni optimal baholash algoritmi.
- Prognoz va ma'lumot qayta ishlash masalalarida keng qo'llaniladi, masalan, manzil koordinatalarini aniqlash, shaxsni kuzatish, iqtisodiy prognozlar.
- Holat va kuzatuvlar o'rtasidagi chiziqli munosabatni qo'llaydi va natijada optimal baholash va prognozlashni ta'minlaydi.

### 4. **Dinamik dasturlash va graflar:**

- Graflar va tarmoqlar ustida dinamik dasturlash usullari, masalan, eng qisqa yo'l, eng keng tarqalgan quvvat oqimi, minimal qamrovli yechim topish.
- Bu usullar tarmoq optimizatsiyasi, logistika, kommunikatsiya, ijtimoiy tarmoqlar tahlili kabi masalalarda qo'llaniladi.

Ushbu usullar statistik axborotlarni qayta ishlashda va prognoz masalalarida samarali echimlarni topishga imkon beradi. Ular boshqa qator algoritmlar (masalan, mashinali o'rganish) bilan birgalikda ishlatilib, statistik tahlil va qaror qabul qilish jarayonlarini yaxshilashi mumkin.

Statistik axborotlarni qayta ishlash va prognoz masalalarida dinamik dasturlash usullari juda keng qo'llaniladi. Bu usullar statistik ma'lumotlarni aniqlash, tahlil qilish, o'rganish, va prognozlashda ishlatiladi. Quyidagi usullar statistik axborotlarni qayta ishlash va prognoz masalalarida amalga oshirishda keng qo'llaniladi:

1. **Mashina o'qitish (Machine Learning):** Mashina o'qitish algoritmlari, katta miqdorda axborotlarni o'rganib chiqish, ma'lumotlar tahlili, va prognozlashda juda samarali bo'lib chiqadi. Bu usullar statistik modellarni yaratishda va axborotlarni tushunishda yordam beradi. Ko'p mashina o'qitish usullari mavjud, masalan, k-means, naqshga asoslangan sinov rejimlari (SVM), logistik regressiya, nöron tarmoqlar (neural networks) kabi. Bu usullar bilan, statistik modellar yaratiladi va axborotlarni aniqlash, tahlil qilish, va prognozlashda yordam beriladi.
2. **Vaqtli qo'llash (Time Series Analysis):** Vaqtli qo'llash statistik ma'lumotlarni vaqt o'qimlariga ko'ra tahlil qilish va prognozlashda foydalaniladi. Bu usul yordamida, ma'lumotlar diqqatga olingan tartibda, so'nggi vaqt oralig'idagi o'zgarishlar, sezonal o'zgarishlar, trendlar va boshqa mazmungalarga qarab tahlil qilinadi. Vaqtli ma'lumotlar uchun katta miqdorda statistik modellar va algoritmlar mavjud, masalan, avtomatik regressiya modellari, ARIMA (Autoregressive Integrated Moving Average), prophet, LSTM (Long Short-Term Memory) nöron tarmoqlar, kabi.
3. **Bayes tahlili (Bayesian Analysis):** Bayes tahlili ma'lumotlarni tahlil qilish va prognozlashda foydalaniladi. Ushbu usulda, ma'lumotlar to'g'risida avvalgi aniqlangan ma'lumotlar va yangi ma'lumotlar birgalikda ishlatiladi. Bayes tahlili ma'lumotlarni tahlil qilishda va o'rganishda juda kuchli bo'lib chiqadi, ammo shuningdek, ma'lumotlar o'rtacha o'zgarishlari bilan birgalikda o'qitish imkonini beradi.
4. **Optimizatsiya algoritmlari:** Optimizatsiya algoritmlari ma'lumotlar tahlilini va boshqarishini yaxshilash va maqsadlarni tanlashda ishlatiladi. Bu usulda, maqsad funksiyasi belgilanadi va ma'lumotlar qanday qilib eng yaxshi natijalarni olish uchun optimallashtiriladi. Ko'p optimizatsiya algoritmlari mavjud, masalan, Genetik algoritmlar, Tushuncha optimizatsiyasi, Ko'rsatuvchi optimizatsiya, kabi.
5. **Tizimlar dinamikasi va modelash (System Dynamics and Modeling):** Tizimlar dinamikasi va modelash usullari tizimlarning vaqt o'qimlariga qarab modelashni va aniq vaqt o'qimlariga ko'ra o'zgaruvchilarni tushuntirishni ta'minlaydi. Bu usulda, o'zgaruvchilar, xususiyatlari, va ularga ta'sir etuvchi faktorlar tasavvur qilinadi. So'nggi yillarda, agent bazali model qurish, kompleks tizimlar dinamikasi, va asosiy statistik modellar yaratishda mashhur bo'ldi.

Ushbu usullar ma'lumotlarni qayta ishlash va prognoz masalalarida juda samarali va kuchli bo'lib chiqadi. Har bir usulning xususiyatlari va foydalangan maqsadiga ko'ra ma'lumotlarni optimallashtirish va natijalarni yaxshilashga yordam beradi.

Statistik axborotlarni qayta ishlash va prognoz masalalarida quyidagi dinamik dasturlash usullari keng qo'llaniladi:

1. **Mashina o'qitish (Machine Learning):** Bu usulda, matematik model yaratish uchun mashina o'qitiladi. Ma'lumotlar tahlili, sinov rejimlari, regressiya, tushuncha tahlili, neron tarmoqlar (neural networks), qo'shniy o'qitish (ensemble learning) kabi usullar mashina o'qitishning muhim qismlarini tashkil etadi. Ushbu usul statistik modellarni yaratishda va ma'lumotlarni tushunishda yordam beradi.
2. **Vaqtqi qo'llash (Time Series Analysis):** Vaqtqi qo'llash, ma'lumotlarni vaqt o'qimlariga qarab tahlil qilish va prognozlashda o'zlashtirilgan usullardan biridir. Ushbu usul statistik ma'lumotlarni diqqatga olingan tartibda, so'nggi vaqt oralig'idagi o'zgarishlar, sezonal o'zgarishlar, trendlar va boshqa mazmungalarga qarab tahlil qiladi.
3. **Bayes tahlili (Bayesian Analysis):** Bayes tahlili, ma'lumotlarni tahlil qilish va prognozlashda foydalaniladi. Bu usulda, ma'lumotlar to'g'risida avvalgi aniqlangan ma'lumotlar va yangi ma'lumotlar birgalikda ishlatiladi. Bayes tahlili ma'lumotlarni tahlil qilishda va o'rganishda juda kuchli bo'lib chiqadi.
4. **Optimizatsiya algoritmlari:** Optimizatsiya algoritmlari ma'lumotlar tahlilini va boshqarishini yaxshilash va maqsadlarni tanlashda ishlatiladi. Bu usulda, maqsad funksiyasi belgilanadi va ma'lumotlar qanday qilib eng yaxshi natijalarni olish uchun optimallashtiriladi.
5. **Tizimlar dinamikasi va modellashtirish (System Dynamics and Modeling):** Tizimlar dinamikasi va modellashtirish usullari tizimlarning vaqt o'qimlariga qarab modellashtirishni va aniq vaqt o'qimlariga ko'ra o'zgaruvchilarni tushuntirishni ta'minlaydi. Ushbu usulda, o'zgaruvchilar, xususiyatlari, va ularga ta'sir etuvchi faktorlar tasavvur qilinadi.

Bu usullar statistik ma'lumotlarni tahlil qilish va prognozlashda keng qo'llaniladi. Har bir usulning xususiyatlari va foydalangan maqsadiga ko'ra ma'lumotlarni optimallashtirish va natijalarni yaxshilashga yordam beradi.

## S246 Saralash algoritmlarini sanang va ularni farqlarini tushuntiring

Saralash (sorting) algoritmlari to'plam yoki ketma-ketlikni ma'lum tartibga keltirish uchun qo'llaniladigan asosiy algoritmlardir. Quyida eng mashhur saralash algoritmlarini va ularning farqlarini tushuntiraman:

### 1. Quyidagi saralash algoritmlari:

- Puzircha (Bubble) saralash
- Tanlash (Selection) saralash
- Kiritish (Insertion) saralash
- Tezkor (Quicksort) saralash
- Birlashtirish (Merge) saralash
- Darvoza (Heap) saralash
- Qobiq (Shell) saralash
- Sanash (Counting) saralash
- Radix saralash

### 2. Algoritmlardagi asosiy farqlar:

- **Murakkablik darajasi:** Har bir algoritm o'ziga xos vaqt va xotira murakkabligi bilan farqlanadi. Bu ko'rsatkichlar algoritm samaradorligini belgilaydi.
- **Barqarorlik:** Ba'zi algoritmlarda ketma-ketlikdagi teng elementlar o'z o'rnini saqlab qoladi (barqaror), boshqalarida esa bu holat saqlanmaydi.
- **Xotira sarfi:** Ayrim algoritmlarda faqat kirish ma'lumotlari ishlatilib, boshqalari qo'shimcha xotira oladigan dinamik ma'lumotlar tuzilmalaridan foydalanadi.
- **Parallel ishlash:** Ayrim algoritmlarda parallel ishlatish imkoniyatlari mavjud, boshqalari esa faqat ketma-ket bajariladi.
- **O'ziga xos xususiyatlari:** Masalan, sanash saralash faqat butun sonlar uchun ishlatiladi, radix saralash esa raqamli belgilar bilan ishlaydi.

### 3. Algoritmlarning qo'llanish sohalari:

- Puzircha, kiritish, qobiq - kichik o'lchamdagi ketma-ketliklar uchun qo'llaniladi.
- Tezkor, birlashtirish, darvoza - o'rta va katta o'lchamli ketma-ketliklar uchun moslashgan.
- Sanash, radix - belgilangan oraliqda joylashgan butun sonlar uchun qulay.

Ushbu saralash algoritmlarining o'ziga xos xususiyatlari, murakkabligi va qo'llanish sohalari ularning tanlanishini belgilaydi. Amaliy dasturlashda turli algoritmlarning afzalliklari va kamchiliklari hisobga olinadi.

Saralash algoritmlari, elementlarni tartiblash uchun foydalaniladigan usullardir. Ular elementlarni qat'iy tartibda joylashtirish uchun foydalaniladi. Bu algoritmlar odatda ma'lumotlar to'plamlarini o'sish va qisqartirish, ma'lumotlar qo'shilishi va qidirish,

qavat tartibidagi elementlarni o'zgartirish va boshqalar kabi amallarni bajarishda ishlatiladi. Quyidagi saralash algoritmlari keng qo'llaniladi:

1. **Chiziqsaralash (Bubble Sort):** Chiziqsaralash algoritmi, qiyosiy ikki elementni almashtirib ko'rib, ularni to'g'ri tartibda joylashtirishga qaror qiladi. Agar birinchi element ikkinchisidan katta bo'lsa, ular almashtiriladi. Bu jarayon to'g'riroq tartiblangan ma'lumotlar to'plamini olish uchun juda oson va qulay bo'lib chiqadi. Ammo, odatda, juda katta ma'lumotlar to'plamlarida ishlash bosqichi borasida qiyinliklar keltiradi.
2. **Chelaksaralash (Selection Sort):** Chelaksaralash algoritmi, ma'lumotlar to'plamida eng kichik elementni tanlash bilan boshlanadi. Eng kichik elementni chap tomoniga joylashtiradi, keyin qoldiq to'plamda qayta eng kichik elementni qidirib uni chap tomoniga joylashtiradi. Bu jarayon to'g'riroq tartiblangan ma'lumotlar to'plamini olish uchun yaxshi bo'lib chiqadi, ammo katta ma'lumotlar to'plamlari uchun juda katta xatolik soni bor.
3. **Chaqirish (Insertion Sort):** Chaqirish algoritmi, qadam darajali usulda elementni joylashtiradi. Har bir yangi element kelib tushganda, uni ma'lumotlar to'plamining to'g'ri joyiga joylashtiradi. Bu algoritmda har bir yangi element bironta joyga joylashtiriladi. Ammo, uning barcha elementlarni katta to'plamlarda joylashtirish va o'zgarishlarini tekshirish uchun yuzlab vaqt sarflanadi.
4. **Birlashtirish (Merge Sort):** Birlashtirish algoritmi, ma'lumotlar to'plamini ikki yarimka bo'lib bo'lish bilan boshlanadi va keyin har ikki yarimni birlashtiradi. Har bir yarimda faqatgina bittasi qolishadi. Keyin qoldiq yarimlar ham birlashtiriladi. Birlashtirish algoritmi saralash jarayonlarida juda samarali bo'lib chiqadi, ammo o'zgarishlar uchun qo'shimcha xotira zarur bo'ladi.
5. **Bajrsaralash (Quick Sort):** Bajrsaralash algoritmi, birlashtirishning ifodalashtirilgan varianti hisoblanadi. Elementlar orasida bo'lgan burchakni tanlash, to'plamni uch qismga bo'lish, qisqa burchakni saralash va boshqalar orasida taqsimlash va qoldiq elementlarni birlashtirish jarayonlaridan iborat. Bu algoritm saralash amalini ikkita usulda bajara olishni bajaradi.
6. **Tim saralash (Tim Sort):** Tim saralash algoritmi, birlashtirish va chaqirish saralashlari orasida ishlaydi. Keyincha, juda samarali va oddiy algoritm hisoblanadi.

Farqlar esa quyidagicha:

- **Samimiyati:** Ba'zi algoritmlar odatda boshqalardan yaxshi bo'lsa, boshqalar esa ma'lumotlar to'plamlari turi va o'lchami bo'yicha afzalliklarga ega bo'lishi mumkin.
- **Effektivligi:** Har bir algoritmning ma'lumotlar to'plami turi va o'lchami bo'yicha amalga oshirilishining effektivligi farq qiladi. Masalan, chiziqsaralash

algoritmi oddiy ma'lumotlar to'plamlarini saralashda samarali bo'lsa, birlashtirish algoritmi juda katta ma'lumotlar to'plamlarini saralashda yaxshi bo'ladi.

- **Zaruratlar:** Har bir algoritmnning xususiy zarurlari va cheklari mavjud. Bunday cheklar, algoritmning ma'lumotlar turi, o'lchami, katta elementlarning soni, xotira miqdori, vaqtni qanchalik qo'llashining chegarasi va boshqalar bilan bog'liq bo'ladi.
- **Qulaylik:** Ba'zi algoritmlar boshqa algoritmlardan ko'proq qulay va sodda bo'lishi mumkin. Ammo, qulaylik doirasida, ularning samarali bo'lishi kerak.

Saralash algoritmlari o'rtasida farqlar va o'z xususiyatlari mavjudligi tufayli, qanday ma'lumotlar to'plamini saralash kerakligiga va qanday vaqt va xotira miqdorini ajratish kerakligiga qaror qilish juda muhimdir.

## S247 Asimtotik taxlil natijalarini oling va tushuntiring

Asimtotik tahlil algoritm samaradorligini baholash uchun qo'llaniladi. Asosiy asimtotik tahlil natijalarini quyidagicha tushuntirish mumkin:

### 1. **O-notation (Big-O notation):**

- Algoritm bajarilish vaqtining yuqori chegarasini ifodalaydi.
- $O(f(n))$  - algoritm bajarilish vaqti  $f(n)$  funksiyasi bilan cheklanadi.
- Masalan,  $O(n)$  - chiziqli,  $O(n^2)$  - kvadratik,  $O(\log n)$  - logarifmik murakkablikka ega algoritm.
- Bu eng muhim asimtotik tahlil natijasidir, algoritm samaradorligini baholash uchun eng ko'p qo'llaniladi.

### 2. **$\Omega$ -notation (Big-Omega notation):**

- Algoritm bajarilish vaqtining quyi chegarasini ifodalaydi.
- $\Omega(f(n))$  - algoritm bajarilish vaqti  $f(n)$  funksiyasi bilan cheklanadi.
- Masalan,  $\Omega(n)$  - chiziqli,  $\Omega(n \log n)$  -  $n \log n$  murakkablikka ega algoritm.
- Asimptotik quyi chegarani aniqlash uchun qo'llaniladi.

### 3. **$\Theta$ -notation (Big-Theta notation):**

- Algoritm bajarilish vaqtini aniq chegaralaydi.
- $\Theta(f(n))$  - algoritm bajarilish vaqti  $f(n)$  funksiyasi bilan cheklanadi.
- Masalan,  $\Theta(n^2)$  - kvadratik,  $\Theta(n \log n)$  -  $n \log n$  murakkablikka ega algoritm.
- Asimptotik yuqori va quyi chegaralarni aniqlash uchun qo'llaniladi.

Bu asimtotik tahlil natijalarini quyidagicha tushunish mumkin:

- O-notation algoritm bajarilish vaqtining eng yomon holati (yuqori chegara)ni ko'rsatadi.
- $\Omega$ -notation algoritm bajarilish vaqtining eng yaxshi holati (quyi chegara)ni ko'rsatadi.
- $\Theta$ -notation algoritm bajarilish vaqtining aniq asimptotik chegarasini ko'rsatadi.

Asimtotik tahlil natijalari algoritm samaradorligini baholash, taqqoslash va tanlab olish uchun muhim ahamiyatga ega. Ular algoritmlar murakkabligini aniq tavsiflay oladi va algoritm tanlash, optimallashtirish va murakkab masalalarni hal qilishda qo'llaniladi.

Asimptotik taxlil, algoritmlarning bajarilish vaqtini o'sishining miqdori haqida ma'lumot beradi. Bu taxlil, algoritmning kattaligi bilan bog'liq bo'lib, algoritmdagi barcha bajariladigan amallar soni bo'yicha kirish, chiqish va boshqa amallar sonini hisoblashga asoslangan. Quyidagi algoritmlar uchun asimptotik taxlil natijalarni oling:

### 1. **Chiziqsaralash (Bubble Sort):**

- Asimptotik taxlil:  $O(n^2)$



- Tushuntirish: Chiziqsaralash algoritmi o'nlik matritsaning elementlarini qiyosiy ikki ikki qo'shiladigan har bir elementni almashtirib ko'rib, agar ularni almashtirib bo'lish kerak bo'lsa, ularni almashtiradi. Eng kichik elementlar oxirga to'g'ri tartiblashga joylashtiriladi. Shu sababli, algoritmdagi bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

## 2. Chelaksaralash (Selection Sort):

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chelaksaralash algoritmi har bir yangi element qoldiq elementlar orasidan eng kichik elementni tanlaydi va uni chap tomoniga joylashtiradi. Keyin, qoldiq elementlarning ichidan esa qayta eng kichik element tanlanadi. Bunda, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

## 3. Chaqirish (Insertion Sort):

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chaqirish algoritmi yangi elementni mavjud tartibda to'g'ri joyga joylashtiradi. Har bir yangi element bironta joyga joylashtiriladi. Bunda, qoldiq elementlarning joyini o'zgartirish va ularni tekshirish uchun matritsaning barcha elementlarini qayta ko'rib chiqish kerak. Shu sababli, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

## 4. Birlashtirish (Merge Sort):

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Birlashtirish algoritmi birlashgan saralash algoritmlari orasida eng samarali bo'lib, bajariladigan amallar soni o'nlik matritsaning logaritmiga tengdir. Bu algoritm quyidagi tartibda ishlaydi:
  - Ma'lumotlar to'plamini ikki teng qismlarga bo'lib bo'lish;
  - Har bir yarimni birlashtirish;
  - Qoldiq yarimlarni ham birlashtirish.

## 5. Bajsaralash (Quick Sort):

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Bajsaralash algoritmi o'nlik matritsada boshqa xisoblagichlar orasida o'nlikning logaritmiga teng elementlarni ajratib olish bilan boshlanadi. Elementlar orasidagi bo'lsa, ularni to'g'ri joyga joylashtiriladi. Bunda, o'nlikning logaritmiga teng bo'lgan bajariladigan amallar soni o'nlik matritsaning logaritmiga tengdir.

## 6. Tim saralash (Tim Sort):

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Tim saralash algoritmi birlashtirish va chaqirish saralashlari orasida ishlaydi. Asimptotik taxlili bu algoritmda o'nlik matritsaning logaritmiga tengdir.

Asimptotik taxlil natijalari algoritmlar uchun juda muhimdir, chunki ular algoritmlarning ishlayish vaqtini, katta ma'lumotlar to'plamlarida ham, qulayroq aniqlashga yordam beradi. Bunday taxlil bajariladigan amallar soni yoki kattaligi bilan bog'liq emas, balki algoritmdagi o'z xususiyatlarga bog'liqdir.

Asimptotik taxlil, algoritmlarning bajarilish vaqtini o'sishining miqdori haqida ma'lumot beradi. Quyidagi algoritmlar uchun asimptotik taxlil natijalarni ko'ramiz:

**1. Chiziqsaralash (Bubble Sort):**

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chiziqsaralash algoritmi har bir elementni qiyosiy barcha qolgan elementlar bilan solishtirib, ularni to'g'ri tartibda joylashtiradi. Bunda, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

**2. Chelaksaralash (Selection Sort):**

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chelaksaralash algoritmi har bir elementni qoldiq elementlar orasidan eng kichik elementni tanlaydi va uni joylashtiradi. Keyin qoldiq elementlarning ichidan esa qayta eng kichik element tanlanadi. Bunda, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

**3. Chaqirish (Insertion Sort):**

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chaqirish algoritmi yangi elementni mavjud tartibda to'g'ri joyga joylashtiradi. Har bir yangi element bironta joyga joylashtiriladi. Bunda, qoldiq elementlarning joyini o'zgartirish va ularni tekshirish uchun matritsaning barcha elementlarini qayta ko'rib chiqish kerak. Shu sababli, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

**4. Birlashtirish (Merge Sort):**

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Birlashtirish algoritmi birlashgan saralash algoritmlari orasida eng samarali bo'lib, bajariladigan amallar soni o'nlik matritsaning logaritmiga tengdir. Bu algoritm quyidagi tartibda ishlaydi:
  - Ma'lumotlar to'plamini ikki teng qismlarga bo'lib bo'lish;
  - Har bir yarimni birlashtirish;
  - Qoldiq yarimlarni ham birlashtirish.

**5. Bajsaralash (Quick Sort):**

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Bajsaralash algoritmi o'nlik matritsada boshqa xisoblagichlar orasida o'nlikning logaritmiga teng elementlarni ajratib olish bilan boshlanadi. Elementlar orasidagi bo'lsa, ularni to'g'ri joyga joylashtiriladi. Bunda, o'nlikning logaritmiga teng bo'lgan bajariladigan amallar soni o'nlik matritsaning logaritmiga tengdir.

## 6. Tim saralash (Tim Sort):

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Tim saralash algoritmi birlashtirish va chaqirish saralashlari orasida ishlaydi. Asimptotik taxlili bu algoritmda o'nlik matritsaning logaritmiga tengdir.

Asimptotik taxlil natijalari algoritmlar uchun juda muhimdir, chunki ular algoritmlarning ishlayish vaqtini, katta ma'lumotlar to'plamlarida ham, qulayroq aniqlashga yordam beradi. Bunday taxlil bajariladigan amallar soni yoki kattaligi bilan bog'liq emas, balki algoritmdagi o'z xususiyatlarga bog'liqdir.

Asimptotik taxlil, algoritmlarning bajarilish vaqtini o'sishining miqdori haqida ma'lumot beradi. Quyidagi algoritmlar uchun asimptotik taxlil natijalarni ko'ramiz:

### 1. Chiziqsaralash (Bubble Sort):

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chiziqsaralash algoritmi har bir elementni qiyosiy barcha qolgan elementlar bilan solishtirib, ularni to'g'ri tartibda joylashtiradi. Bunda, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

### 2. Chelaksaralash (Selection Sort):

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chelaksaralash algoritmi har bir elementni qoldiq elementlar orasidan eng kichik elementni tanlaydi va uni joylashtiradi. Keyin qoldiq elementlarning ichidan esa qayta eng kichik element tanlanadi. Bunda, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

### 3. Chaqirish (Insertion Sort):

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chaqirish algoritmi yangi elementni mavjud tartibda to'g'ri joyga joylashtiradi. Har bir yangi element bironta joyga joylashtiriladi. Bunda, qoldiq elementlarning joyini o'zgartirish va ularni tekshirish uchun matritsaning barcha elementlarini qayta ko'rib chiqish kerak. Shu sababli, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

### 4. Birlashtirish (Merge Sort):

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Birlashtirish algoritmi birlashgan saralash algoritmlari orasida eng samarali bo'lib, bajariladigan amallar soni o'nlik matritsaning logaritmiga tengdir. Bu algoritm quyidagi tartibda ishlaydi:
  - Ma'lumotlar to'plamini ikki teng qismlarga bo'lib bo'lish;
  - Har bir yarimni birlashtirish;
  - Qoldiq yarimlarni ham birlashtirish.

### 5. Bajsaralash (Quick Sort):

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Bajsaralash algoritmi o'nlik matritsada boshqa xisoblagichlar orasida o'nlikning logaritimga teng elementlarni ajratib olish bilan boshlanadi. Elementlar orasidagi bo'lsa, ularni to'g'ri joyga joylashtiriladi. Bunda, o'nlikning logaritimga teng bo'lgan bajariladigan amallar soni o'nlik matritsaning logaritmiga tengdir.

#### 6. **Tim saralash (Tim Sort):**

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Tim saralash algoritmi birlashtirish va chaqirish saralashlari orasida ishlaydi. Asimptotik taxlili bu algoritmda o'nlik matritsaning logaritmiga tengdir.

Asimptotik taxlil natijalari algoritmlar uchun juda muhimdir, chunki ular algoritmlarning ishlayish vaqtini, katta ma'lumotlar to'plamlarida ham, qulayroq aniqlashga yordam beradi. Bunday taxlil bajariladigan amallar soni yoki kattaligi bilan bog'liq emas, balki algoritmdagi o'z xususiyatlarga bog'liqdir.

Asimptotik taxlil, algoritmlarning bajarilish vaqtini o'sishining miqdori haqida ma'lumot beradi. Quyidagi algoritmlar uchun asimptotik taxlil natijalarni ko'ramiz:

#### 1. **Chiziqsaralash (Bubble Sort):**

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chiziqsaralash algoritmi har bir elementni qiyosiy barcha qolgan elementlar bilan solishtirib, ularni to'g'ri tartibda joylashtiradi. Bunda, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

#### 2. **Chelaksaralash (Selection Sort):**

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chelaksaralash algoritmi har bir elementni qoldiq elementlar orasidan eng kichik elementni tanlaydi va uni joylashtiradi. Keyin qoldiq elementlarning ichidan esa qayta eng kichik element tanlanadi. Bunda, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

#### 3. **Chaqirish (Insertion Sort):**

- Asimptotik taxlil:  $O(n^2)$
- Tushuntirish: Chaqirish algoritmi yangi elementni mavjud tartibda to'g'ri joyga joylashtiradi. Har bir yangi element bironta joyga joylashtiriladi. Bunda, qoldiq elementlarning joyini o'zgartirish va ularni tekshirish uchun matritsaning barcha elementlarini qayta ko'rib chiqish kerak. Shu sababli, bajariladigan amallar soni o'nlik matritsaning kvadratga tengdir.

#### 4. **Birlashtirish (Merge Sort):**

- Asimptotik taxlil:  $O(n \log n)$

- Tushuntirish: Birlashtirish algoritmi birlashgan saralash algoritmlari orasida eng samarali bo'lib, bajariladigan amallar soni o'nlik matritsaning logaritmiga tengdir. Bu algoritm quyidagi tartibda ishlaydi:
  - Ma'lumotlar to'plamini ikki teng qismlarga bo'lib bo'lish;
  - Har bir yarimni birlashtirish;
  - Qoldiq yarimlarni ham birlashtirish.

#### 5. Bajsaralash (Quick Sort):

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Bajsaralash algoritmi o'nlik matritsada boshqa xisoblagichlar orasida o'nlikning logaritimiga teng elementlarni ajratib olish bilan boshlanadi. Elementlar orasidagi bo'lsa, ularni to'g'ri joyga joylashtiriladi. Bunda, o'nlikning logaritimiga teng bo'lgan bajariladigan amallar soni o'nlik matritsaning logaritmiga tengdir.

#### 6. Tim saralash (Tim Sort):

- Asimptotik taxlil:  $O(n \log n)$
- Tushuntirish: Tim saralash algoritmi birlashtirish va chaqirish saralashlari orasida ishlaydi. Asimptotik taxlili bu algoritmda o'nlik matritsaning logaritmiga tengdir.

Asimptotik taxlil natijalari algoritmlar uchun juda muhimdir, chunki ular algoritmlarning ishlayish vaqtini, katta ma'lumotlar to'plamlarida ham, qulayroq aniqlashga yordam beradi. Bunday taxlil bajariladigan amallar soni yoki kattaligi bilan bog'liq emas, balki algoritmdagi o'z xususiyatlarga bog'liqdir.

## S248 Asimtotik taxlil tushuntiring va misollar keltiring

Asimtotik tahlil (asymptotic analysis) algoritm bajarilish vaqtini, xotira sarfini va boshqa resurs sarflarini baholash uchun qo'llaniladigan amaliy usuldir. Asimtotik tahlil algoritmlarning masshtablanuvchanligi va miqyosilanishi haqida ma'lumot beradi.

Asimtotik tahlilda algoritmning bajarilish vaqti va resurs sarflarini baholash uchun quyidagi asosiy notatsiyalar ishlatiladi:

1. **O-notation (Big-O notation)** - algoritm bajarilish vaqtining yuqori chegarasi:
  - $O(f(n)) = \{g(n) \mid \exists c > 0, n_0 \geq 0 : 0 \leq g(n) \leq c * f(n) \forall n \geq n_0\}$
  - Masalan,  $O(n)$  - chiziqli murakkablik,  $O(n^2)$  - kvadratik murakkablik.
2.  **$\Omega$ -notation (Big-Omega notation)** - algoritm bajarilish vaqtining quyi chegarasi:
  - $\Omega(f(n)) = \{g(n) \mid \exists c > 0, n_0 \geq 0 : 0 \leq c * f(n) \leq g(n) \forall n \geq n_0\}$
  - Masalan,  $\Omega(n)$  - chiziqli murakkablik,  $\Omega(n \log n)$  -  $n \log n$  murakkablik.
3.  **$\Theta$ -notation (Big-Theta notation)** - algoritm bajarilish vaqtining aniq chegarasi:
  - $\Theta(f(n)) = \{g(n) \mid \exists c_1, c_2 > 0, n_0 \geq 0 : 0 \leq c_1 * f(n) \leq g(n) \leq c_2 * f(n) \forall n \geq n_0\}$
  - Masalan,  $\Theta(n^2)$  - kvadratik murakkablik,  $\Theta(n \log n)$  -  $n \log n$  murakkablik.

### Misollar:

1. **Linear Search:**
  - Ilova massivdagi element qidiruvi
  - Vaqt murakkabligi:  $O(n)$
  - Quyi chegara:  $\Omega(1)$
  - Aniq chegara:  $\Theta(n)$
2. **Binary Search:**
  - Tartiblangan massivdagi element qidiruvi
  - Vaqt murakkabligi:  $O(\log n)$
  - Quyi chegara:  $\Omega(\log n)$
  - Aniq chegara:  $\Theta(\log n)$
3. **Sorting Algorithms:**
  - Puzircha saralash:  $O(n^2)$ ,  $\Omega(n)$ ,  $\Theta(n^2)$
  - Tezkor saralash:  $O(n \log n)$ ,  $\Omega(n \log n)$ ,  $\Theta(n \log n)$
  - Birlashtirish saralash:  $O(n \log n)$ ,  $\Omega(n \log n)$ ,  $\Theta(n \log n)$

Asimtotik tahlil algoritmlar samaradorligini baholash, solishtirish va tanlash uchun juda muhim vositadir. U algoritmning eng yaxshi, eng yomon va aniq chegaralarini hisoblab beradi.

Asimptotik taxlil, algoritmlarning bajarilish vaqtini o'sishining miqdori haqida ma'lumot beradi. Bu taxlil, algoritmning kattaligi bilan bog'liq bo'lib, algoritmdagi

barcha bajariladigan amallar soni bo'yicha kirish, chiqish va boshqa amallar sonini hisoblashga asoslangan. Asimptotik taxlilning uchta kategoriyasi mavjud:

1. **O'rtacha (Average):** Algoritmning barcha muammolar uchun o'rtacha bajariladigan amallar soni.
2. **Eng yomon (Worst Case):** Algoritmning eng kattaligi uchun bajariladigan amallar soni.
3. **Eng yaxshi (Best Case):** Algoritmning eng kam bajariladigan amallar soni.

Quyidagi misollar bilan asimptotik taxlilni tushunishni oshirishimiz mumkin:

## 1. Chiziqsaralash (Bubble Sort):

- **Asimptotik taxlil:**  $O(n^2)$  (eng yomon)

Bu algoritm quyidagi misolda 4 elementli ro'yxatni tartiblash uchun qo'llaniladi:

rust

Копировать код

Boshlang'ich ro'yxat: [5, 3, 8, 4]

○

1. Qadam: [3, 5, 4, 8]

○

2. Qadam: [3, 4, 5, 8]

Misol ko'rib turganizdek, har bir qadamda barcha elementlar tekshirilib, o'zgarishlar amalga oshiriladi. Agar  $n$  - ro'yxatdagi elementlar soni bo'lsa, algoritmdagi bajariladigan amallar soni  $n^2$  ga teng bo'ladi.

## 2. Chelaksaralash (Selection Sort):

- **Asimptotik taxlil:**  $O(n^2)$  (eng yomon)

Quyidagi misolda 5 elementli ro'yxatni tartiblash uchun chelaksaralash algoritmi qo'llaniladi:

rust

Копировать код

Boshlang'ich ro'yxat: [64, 25, 12, 22, 11]

- 1. Qadam: [11, 25, 12, 22, 64]
- 2. Qadam: [11, 12, 25, 22, 64]
- 3. Qadam: [11, 12, 22, 25, 64]
- 4. Qadam: [11, 12, 22, 25, 64]

Har bir qadamda chelaksaralash algoritmi ro'yxatni yuzlab tekshirib, eng kichik elementni topadi va joylashtiradi. Agar  $n$  - ro'yxatdagi elementlar soni bo'lsa, bajariladigan amallar soni  $n^2$  ga teng bo'ladi.

### 3. Chaqirish (Insertion Sort):

- **Asimptotik taxlil:**  $O(n^2)$  (eng yomon)

Misol uchun 6 elementli ro'yxatni tartiblash uchun chaqirish algoritmi quyidagi ko'rinishda qo'llaniladi:

rust

Копировать код

Boshlang'ich ro'yxat: [12, 11, 13, 5, 6]

- 1. Qadam: [11, 12, 13, 5, 6]
- 2. Qadam: [11, 12, 13, 5, 6]
- 3. Qadam: [5, 11, 12, 13, 6]
- 4. Qadam: [5, 6, 11, 12, 13]

Har bir qadamda yangi element ma'lumotlar to'plamining o'ng tomonida joylashishi kerak, keyin esa joylashtirish uchun bironta joydan boshlab barcha oldingi elementlarni tekshirish va joyini o'zgartirish kerak. Agar  $n$  - ro'yxatdagi



elementlar soni bo'lsa, algoritmdagi bajariladigan amallar soni  $n^2$  ga teng bo'ladi.

#### 4. Birlashtirish (Merge Sort):

- **Asimptotik taxlil:**  $O(n \log n)$  (eng yaxshi)

Misol uchun 8 elementli ro'yxatni tartiblash uchun birlashtirish algoritmi quyidagi ko'rinishda qo'llaniladi:

rust

Копировать код

Boshlang'ich ro'yxat: [38, 27, 43, 3, 9, 82, 10]

○

1. Qadam: [27, 38, 43, 3, 9, 10, 82]

○

2. Qadam: [3, 9, 10, 27, 38, 43, 82]

Birlashtirish algoritmi har bir qadamda ro'yxatni ikki qismga bo'lib, har bir qismini birlashtiradi. Agar  $n$  - ro'yxatdagi elementlar soni bo'lsa, bajariladigan amallar soni  $n \log(n)$  ga teng bo'ladi.

#### 5. Bajsaralash (Quick Sort):

- **Asimptotik taxlil:**  $O(n \log n)$  (eng yaxshi)

Misol uchun 7 elementli ro'yxatni tartiblash uchun bajsaralash algoritmi quyidagi ko'rinishda qo'llaniladi:

rust

Копировать код

Boshlang'ich ro'yxat: [10, 80, 30, 90, 40, 50, 70]

○ 1

## **S249 Ishlab chiqarish jarayonlarida algoritmlarni qo'llash, muammoning murakkablik darajasini aniqlash va yangi algoritmlarni ishlab chiqish va ularning samaradorligini tahlil qilish deganda nimani tushunasiz**

Ishlab chiqarish jarayonlarida algoritmlarni qo'llash, muammoning murakkablik darajasini aniqlash va yangi algoritmlarni ishlab chiqish hamda ularning samaradorligini tahlil qilish deganda quyidagilarni tushunish mumkin:

### **1. Algoritmlarni qo'llash:**

- Ishlab chiqarish jarayonlarini avtomatlashtirish, optimallashtirish va samaradorligini oshirish uchun algoritm va dasturiy ta'minotdan foydalanish.
- Masalan, optimal ish joylarini tanlash, ishlab chiqarish grafigini rejalashtirish, xomashyo va resurslarni boshqarish kabi jarayonlarni algoritmlash.

### **2. Muammoning murakkablik darajasini aniqlash:**

- Har bir ishlab chiqarish jarayoni yoki muammoning xususiyatlarini va murakkablik darajasini baholash.
- Murakkablik darajasi algoritm tanlash va ishlab chiqishda muhim rol o'ynaydi.
- Masalan, chiziqli, logarifmik, kvadratik, eksponensial kabi murakkablik darajalarini aniqlash.

### **3. Yangi algoritmlarni ishlab chiqish:**

- Mavjud algoritmlar yetarli emasligi yoki moslashmayotgani tushunilsa, yangi algoritmlar yaratish.
- Jarayonlarni tahlil qilish, optimallashtirish va yanada samarali echimlarni izlash.
- Masalan, yangi ishlab chiqarish algoritmlarini, yangi qidiruv yoki saralash algoritmlarini, yangi marshrutlashtirish yoki topshirish algoritmlarini ishlab chiqish.

### **4. Algoritmlar samaradorligini tahlil qilish:**

- Ishlab chiqilgan yangi algoritmlar yoki qo'llaniladigan mavjud algoritmlarning samaradorligini baholash.
- Asimptotik tahlil, empirik tajribalar, o'lchov va solishtirish orqali algoritmlar samaradorligini aniqlash.
- Masalan, algoritmning vaqt va xotira sarflarini  $O$ -notation,  $\Omega$ -notation,  $\Theta$ -notation orqali baholash.

Ushbu yondashuv ishlab chiqarishda samaradorlik, moslashuvchanlik va raqobatbardoshlikni ta'minlashga xizmat qiladi. Algoritmlar muammoning murakkabligini hisobga olgan holda qo'llaniladi va yangi, takomillashtirilgan algoritmlar yaratiladi.

Ishlab chiqarish jarayonlarida algoritmlarni qo'llash, muammoning murakkablik darajasini aniqlash va yangi algoritmlarni ishlab chiqish va ularning samaradorligini tahlil qilish, tizimni optimallashtirish va yaxshilashda kritik ahamiyatga ega. Keyingi bosqichlarni tushunish uchun bir qancha amalga oshiriladigan jarayonlarni ko'rib chiqamiz:

**1. Algoritmlarni qo'llash:**

- Ma'lumotlarni saqlash va ma'lumotlar ustida bajariladigan operatsiyalarni amalga oshirishda qo'llaniladigan algoritmlarni tanlash.
- Qo'llaniladigan algoritmlar optimallashtirilgan, ommalashgan va o'rganilganligini ta'minlash uchun etkazib berilishi kerak.

**2. Muammoning murakkablik darajasini aniqlash:**

- Algoritmning muammoning murakkablik darajasini (complexity) aniqlash uchun n-o'rinli jarayonlarni yaratish va natijalarni kuzatish talab etiladi.
- Algoritmlar tezlik va xotira sarflanishi orqali baholash uchun testlarni tuzish va bajarish zarur.

**3. Yangi algoritmlarni ishlab chiqish:**

- Muammoni hal qilish uchun yangi algoritmlarni ishlab chiqish va ularning samaradorligini hisoblashga qodir bo'lgan ilmiy va matematik vositalarini qo'llash.
- Yangi algoritmlarni boshqarish, murakkablik darajasini kamaytirish va ishlab chiqarish jarayonini osonlash uchun matematik model tuzish.

**4. Yangi algoritmlarni tahlil qilish:**

- Yangi algoritmlarni tahlil qilish, ularning samaradorligi, amal qilish darajasi, tezlik, va xotira sarflanishi to'g'risida o'rganish.
- Algoritmning qulayligi, optimalligi, va real yashirin odatlari (hidden habits) to'g'risida o'rganish uchun testlarni tuzish.

Yangi algoritmlar ishlab chiqilgan va ularga mo'ljallangan bo'lsa ham, ularni test qilish, optimallashtirish va amal qilish darajasini aniqlash lozim. Keyingi qadam, yangi algoritmlarni real dunyoda bajarish, va ularning samaradorligini o'rganish uchun mo'ljallangan ko'rsatkichlarga asoslangan testlarni tuzish va bajarish. Yangi algoritmlar tezligi, xotira sarflanishi, optimallashtirilishi va qo'llanish qulayligini oshirish uchun tez foydalanish mumkin bo'lgan vazifalarga duch kelishi kerak. Bu jarayonlarning bir qismini yoki hammasini amalga oshirish, algoritmlarni rivojlantirish va yangi muammolar uchun yechim topishda asosiy ahamiyatga ega.

## S250 “Ajrat va hukumronlik qil” tilidagi algoritmlar.

"Ajrat va hukumronlik qil" (Divide and Conquer) strategiyasi algoritmlarni ishlab chiqishda keng qo'llaniladi. Bu yondashuvning asosiy g'oyasi quyidagicha:

### 1. **Ajratish (Divide):**

- Katta muammoni yoki vazifani bir nechta kichik, mustaqil qismlar (submuammolar)ga bo'lish.
- Har bir submuammoni alohida yechish osonroq bo'ladi.

### 2. **Hukumronlik qilish (Conquer):**

- Har bir submuammoni mustaqil yechish (rekursiv yoki boshqa usullar orqali).
- Submuammolar yechimlarini birlashtirish orqali asosiy muammoning yechimiga erishish.

"Ajrat va hukumronlik qil" prinsipi algoritmlarning samaradorligini oshiradi. Chunki katta muammoni kichik, mustaqil qismlarga bo'lish va ularni alohida yechish, algoritmning umumiy murakkabligini sezilarli darajada kamaytirishga imkon beradi. Masalan, "Tezkor saralash" (Quicksort) algoritmi "Ajrat va hukumronlik qil" paradigmasi asosida ishlaydi:

### 1. **Ajratish:**

- Massivni ikki yoki undan ko'p qismga bo'lish (pivot element orqali).
- Har bir qismni alohida saralash uchun ajratish.

### 2. **Hukumronlik qilish:**

- Massivning har bir qismini mustaqil saralash (rekursiv chaqirish orqali).
- Saralangan qismlarni birlashtirish orqali butun massivni saralash.

Boshqa "Ajrat va hukumronlik qil" algoritmlariga misol qilib quyidagilarni keltirishimiz mumkin:

- **Birlashtirish saralash** (Merge Sort)
- **Ikkilamchi qayta-qidiruv** (Binary Search)
- **Eng katta qiymatni topish** (Maximum Subarray Problem)
- **Matritsalar ko'paytmasi** (Matrix Multiplication)

"Ajrat va hukumronlik qil" strategiyasi algoritmlarni yozishda va optimizatsiya qilishda, shuningdek, murakkab muammolarni yechishda keng qo'llaniladi. U algoritmlarning samaradorligini oshirish, masshtablanuvchanligi va takrorlanuvchanligi uchun muhim ahamiyatga ega.

"Ajrat va hukmronlik qil" ("Divide and Conquer") prinsipi algoritmik o'rganishning muhim qismidir. Ushbu prinsip orqali, muammolar ajratiladi, ular oddiy muammolar bo'lgan qismlariga bo'linadi, har bir qismini hal qilish uchun alohida hal qilinishi mumkin bo'lgan bo'lsa ham, muammolar ajratiladi. Shunday qilib, bu prinsip boshqa oddiy algoritmlarning ishini optimallashtirish va murakkab muammolarni hal qilishda juda samarali bo'ladi. Quyidagi misollar "Ajrat va hukmronlik qil" prinsipi bo'yicha algoritmlarni ko'rsatadi:

### 1. Birlashtirish (Merge Sort):

- Bu algoritim ro'yxatni ikki qismga bo'lib ajratadi, har bir qismini birlashtirib chiqadi. Keyin birlangan qismlar bitta tartibda joylashtiriladi. Bu algoritmda muammolar qo'shimcha xotira bo'lmaydi, lekin uning amaliyoti  $O(n \log n)$  darajada bo'lgan o'rniy ishni bajarish uchun juda samarali.

### 2. Bajsaralash (Quick Sort):

- Bu algoritim ro'yxatni o'rtasidan elementni ajratib olib, kichik va katta elementlarni ajratib olish va ularni qayta ajratib olish uchun ishlatiladi. Bu jarayonlar takrorlanib, ro'yxat birlashgan holda qayta yopiladi. Quick Sort algoritmi asimptotik ravishda o'nlik matritsaning logaritmiga teng bo'lgan samarali algoritmlardan biri hisoblanadi.

### 3. Binar qidiruv (Binary Search):

- Bu algoritim o'stirilgan ro'yxatda keyingi qidiruvni qulaylashtiradi. U o'stirilgan ro'yxatni ikki qismga bo'lib ajratadi va keyingi qidiruvni yana faqat bir qismda bajaradi. Bunday qilib, o'rtacha ish rejasi  $O(\log n)$  darajada bo'ladi.

"Ajrat va hukmronlik qil" prinsipi algoritmik yechimlarda o'ziga xos samarali va ishonchli usulni taklif etadi. Bu usul, murakkab muammolarni hal qilishda muhim asbob hisoblanadi.

"Ajrat va hukmronlik qil" ("Divide and Conquer") prinsipi algoritmik o'rganishning muhim qismidir. Ushbu prinsip orqali, muammolar ajratiladi, ular oddiy muammolar bo'lgan qismlariga bo'linadi, har bir qismini hal qilish uchun alohida hal qilinishi mumkin bo'lgan bo'lsa ham, muammolar ajratiladi. Shunday qilib, bu prinsip boshqa oddiy algoritmlarning ishini optimallashtirish va murakkab muammolarni hal qilishda juda samarali bo'ladi. Quyidagi misollar "Ajrat va hukmronlik qil" prinsipi bo'yicha algoritmlarni ko'rsatadi:

### 1. Birlashtirish (Merge Sort):

- Bu algoritim ro'yxatni ikki qismga bo'lib ajratadi, har bir qismini birlashtirib chiqadi. Keyin birlangan qismlar bitta tartibda joylashtiriladi. Bu algoritmda muammolar qo'shimcha xotira bo'lmaydi, lekin uning amaliyoti  $O(n \log n)$  darajada bo'lgan o'rniy ishni bajarish uchun juda samarali.

### 2. Bajsaralash (Quick Sort):

- Bu algoritim ro'yxatni o'rtasidan elementni ajratib olib, kichik va katta elementlarni ajratib olish va ularni qayta ajratib olish uchun ishlatiladi. Bu jarayonlar takrorlanib, ro'yxat birlashgan holda qayta yopiladi. Quick Sort algoritmi asimptotik ravishda o'nlik matritsaning logaritmiga teng bo'lgan samarali algoritmlardan biri hisoblanadi.

### 3. **Binar qidiruv (Binary Search):**

- Bu algoritim o'stirilgan ro'yxatda keyingi qidiruvni qulaylashtiradi. U o'stirilgan ro'yxatni ikki qismga bo'lib ajratadi va keyingi qidiruvni yana faqat bir qismda bajaradi. Bunday qilib, o'rtacha ish rejasi  $O(\log n)$  darajada bo'ladi.

"Ajrat va hukmronlik qil" prinsipi algoritmik yechimlarda o'ziga xos samarali va ishonchli usulni taklif etadi. Bu usul, murakkab muammolarni hal qilishda muhim asbob hisoblanadi