

O'zbekiston Respublikasi Oliy va O'rta maxsus ta'lim Vazirligi

**TERMIZ DAVLAT UNIVERSITETI
FIZIKA-MATEMATIKA FAKULTETI**

AMALIY MATEMATIKA VA INFORMATIKA KAFEDRASI



**«ALGORITMLAR NAZARIYASI»
фанидан**

MA'RUZA MATNLARI

Tuzuvchi : katta o'qituvchi Boboxo'jaeva N.M.

Annotasiya

Amaliy matematika ta'lim yo'nalishi b'yicha O'zbekiston Respublikasi oliy va o'rta maxsus ta'lim vazirligi tomonidan 2003 yilda tasdiqlangan o'quv dasturi asosida tuzildi. Ma'ruza matnlarida algoritmizasiya, algoritmlarni formal tasvirlash, ularning murakkablik darajasi, klassik algoritmlar, xususan, berilganlarni qayta ishlash algoritmlari, algoritmlarning berilish usullari, Tyuring, Post, Markovlarning formal algoritmik sxemalari, rekursiv funksiyalar, algoritmik echimsizlik tushunchasi, saralash, izlash va optimallashtirish algoritmlari o'rganiladi.

Tuzuvchi : k.o'q. N.Boboxo'jaeva

Taqrizchilar: F-m. Fanlari nomzodi M.Choriev

Kafedra mudiri:

200___ yil “___” _____

Termiz davlat universiteti fizika-matematika fakulteti “Amaliy matematika va informatika” kafedrası katta o’qituvchisi Boboxo’jaeva N.M ning 5480100 – Amaliy matematika ta’lim yo’nalishi 2-kurs talabalari uchun “Algoritmlar nazariyasi” fanidan tuzgan ma’ruza matnlariga

TAQRIZ

Mustaqil respublikamizda yuz berayotgan siyosiy, iqtisodiy, ilmiy-texnikaviy va madaniy o’zgarishlar Oliy ta’lim tizimida ham o’z aksini topmoqda. O’zbekistonda uzluksiz ta’lim-tarbiya tizimini yaratish, shu asosida ta’lim sifatini jaxon andozalari darajasiga etkazish ta’lim sistemasining eng dolzarb vazifasiga aylandi. Bu esa barcha mutaxassisliklar qatori kompyuter texnologiyalari bo’yicha kadrlar tayyorlash sifatini oshirishni ham taqozo etadi. Bu maqsad vazifalar ushbu fan dasturi mazmunini ham belgilaydi. Algoritm konsepsiyasining vujudga kelishi bilan algebra, sonlar nazariyasi, geometriya va matematikaning boshqa sohalariga tegishli bir qator muammolarning echimli yoki echimli emasligini aniqlashtirish imkonini berdi. Algoritmlar nazariyasi faoliyat sohasi EHMLar vujudga kelishi bilan yanada kengaydi . Yuqoridagi fikrlar «Algoritmlar nazariyasi va dasturlash texnologiyasi» fanining asosiy mazmunini belgilashga yordam beradi.

Bu ma’ruza matnlari «Algoritmlar nazariyasi i» fanini o’qitishda belgilangan reja asosida ma’ruza o’qish, auditoriya va kompyuter zallaridan foydalangan holda amalga oshiriladi. Bunda talabalar algoritmlar va ularning turlari, murakkablik belgilari, ishonshiligi, strukturaviy dasturlash, ma’lumotlar struktyralari, saralash, izlash va arxivlash va tabiiy fanlarga xos masalalarga doir ba’zi bir algoritmlar o’rganiladi.

1-Mavzu: Algoritmalar nazariyasiga kirish (2 soat)

Reja:

1. Tarixiy ma'lumotlar
2. Algoritmalar nazariyasi fani maqsadi va vazifalari
3. Algoritmalar nazariyasi fani yutuqlarining amaliyotda qo'llanilishi
4. Algoritm tushunchasini formallashtirish

Kalit so'zlar: Alan Tyuring, Aloiz Chyorch, Emil Post, Tyuring mashinasi, Post mashinasi, Chyorchning lyamda- hisoblanuvchanlik usuli, PqNP muammosi

Bizgacha etib kelgan intuitiv ma'nodagi algoritm eramizdan avvalgi III- asrda Evklid tomonidan taklif qilingan. Ushbu algoritm juda mashhur bo'lib, XX-asr boshlarigacha «algoritm» so'zining o'zi «Evklid algoritmi» ma'nosida ishlatilib kelindi. Boshqa matematika masalalarni bosqichli echishni tasvirlash uchun esa «usul» so'zidan foydalanilgan.

Zamonaviy algoritmalar nazariyasi rivojidagi boshlang'ich nuqta deb, nemis matematigi Kurt Gyodelning ilmiy ishini ko'rsatib o'tish mumkin (1931 y. Simvolik mantiqlarning to'lamasligi to'g'risidagi teorema). Ushbu ishda ba'zi matematik muammolarni qaysidir sinfga taalluqli algoritmalar yordamida hal etib bo'lmasligi ko'rsatib berilgan. ...

1936 yilda Algoritmalar nazariyasi bo'yicha birinchi fundamental ilmiy ishlar bir-biridan alohida tarzda Alan Tyuring, Aloiz CHyorch va Emil Postlar e'lon qildilar. Ular tomonidan taklif etilgan Tyuring mashinasi, Post mashinasi va CHyorchning lyamda-hisoblanuvchanlik usuli algoritm formalizmining ekvivalent shakllaridir. Ular tomonidan taklif etilgan tezislar algoritm intuitiv tushunchasi va formal tizimlarning ekvivalentligini ta'kidlab berdi. Algoritmik echimsiz muammolarning formulirovkasi va isboti ushbu ishlarning muhim natijasi bo'ldi.

1950- yillarda Algoritmalar nazariyasi rivojlanishiga rus matematiklari Kolmogorov va Markovlari z hissalarini qo'shdilar . 60-70-yillarga kelib Algoritmalar nazariyasi fanida quyidagi mustaqil yo'nalishlar ajralib chiqdi:

- Klassik algoritmalar nazariyasi(formal tillar terminlarida masalalarni ifodalash, echimli masala tushunchasi, 1965 yilda Edmonds tomonidan ta'riflangan PqNP muammosi, NP to'liq masalalar sinfining ochilishi va tekshirilishi);
- Algoritmalarining asimptotik analizi nazariyasi(asimptotik baholash usullari, algoritmalarining murakkabligi, algoritmalarini baholash kriteriyalari va h.k.z.). Ushbu yo'nalish rivojiga Knut, Axo, Xopkroft, Ulman, Karp kabi olimlar o'z hissalarini qo'shdilar;
- hisoblash algoritmalarining praktik analizi nazariyasi(algoritmalarining mehnattalabligi oshkor funksiyasini topish, funksiyalarining chegaraviy analizi, rasional algoritmalarini tanlash metodikasi).Ushbu yo'nalish rivojlanishiga sabab bo'lgan ilmiy ish D.Knutning "Isskustvo programmirovaniya dlya EVM" kitobidan iborat.

Algoritmalar nazariyasining maqsadi va vazifalari.Algoritmalar nazariyasi turli yo'nalishlarining yutuqlarini umumlashtirib, uning maqsadi va vazifalarini ko'rsatib o'tish mumkin:

- Algoritm tushunchasini formallashtirish va formal algoritmik tizimlarni tekshirish;
- Bir qator masalalarining algoritmik echimsizligini formal isbotlash;
- Masalalar klassifikatsiyasi, murakkablik sinflarini aniqlash va tekshirish;
- Algoritmalar murakkabligining asimptotik analizi;
- Rekursiv algoritmalarini tekshirish va analiz qilish;
- Algoritmalar qiyosiy analizi uchun mehnattalablik oshkor funksiyasini topish;
- Algoritmalar sifatini qiyosiy baholash kriteriyalarini ishlab chiqish;

Algoritmalar nazariyasi fani natijalarining amaliy qo'llanilishi. Algoritmalar nazariyasidan olingan nazariy natijalardan amalda anchayin keng foydalanilmoqda. Bunda ikki aspektni alohida ko'rsatib o'tish mumkin:

Nazariy aspekt: qandaydir masalani tekshirish natijasida “Ushbu masala prinsipial jihatdan algoritmik echimlimi?-, degan savolga javob berish imkoniyati mavjud. Algoritmik echimsiz masalalar Tyuring mashinasi to’xtashi masalasiga olib kelinishi mumkin. Algoritmik echimli masalalar uchun esa, ushbu masalaning NP to’liq masalalar sinfiga mansubligi muhim ikkinchi nazariy savol bo’lib hisoblanadi.

Amaliy aspekt: Algoritmilar nazariyasi usullari quyidagi vazifalarni bajarishga imkon beradi:

- Berilgan masalani echish algoritmilari to’plamidan eng rasional algoritmni tanlash;
- Murakkab masalalarni echish algoritmilarini vaqt jihatidan baholash;
- Kriptografik metodlar uchun muhim bo’lgan masala echimini ma’lum vaqt oralig’ida olib bo’lmasligini ishonchli baholash;
- Praktiki analiz asosida axborotlarni qayta ishlash sohasidagi masalalarni echish effektiv algoritmilarini ishlab chiqish va rivojlantirish;

Algoritm tushunchasini formallashtirish. Inson o’zining barcha faoliyat sohalarida, jumladan axborotlarni qayta ishlashda ham masalalarni echishning turli usul va vositalari bilan to’qnashadi. Ular pirovard natijaga erishish uchun bajariladigan xarakteristikalar tartibini aniqlaydi. Buni intuitiv ma’nodagi algoritm tushunchasi deb qarashimiz mumkin. Ushbu tushunchaga qo’yiladigan ba’zi talablar esa algoritmni noformal aniqlash imkonini beradi:

1. Algoritm - qaysidir tilda berilgan masalani echish uchun bajariladigan boshlang’ich berilganlar ustida bajariladigan amallarning chekli ketma-ketligi.

D- masalaning boshlang’ich berilganlar sohasi (to’plami), R – mumkin bo’lgan natijalar to’plami bo’lsin. Bu holda algoritm $D \rightarrow R$ akslantirishni bajaradi deb hisoblashimiz mumkin. Ushbu akslantirish to’la bo’lmasligi mumkin bo’lgani uchun quyidagi tushunchalarni kiritamiz:

Algoritm qisman deyiladi, agar natija faqat ba’zi D lar uchun olinishi mumkin bo’lsa, to’la algoritm deyiladi, gar barcha D lar uchun natija olinishi mumkin bo’lsa.

Olimlarning izchil faoliyatlariga qaramay, Algoritm tushunchasiga bitta konkret ta’rif berishning imkoni bo’lmadi. Algoritmilar nazariyasida algoritmning turli formal ta’riflari kiritilgan bo’lib, ularning ekvivalentligi isbotlangan.

A.N. Kolmogorov ta’rifi. Algoritm - bu qo’yilgan masala natijasiga qandaydir sondagi qadamlardan keyin olib keluvchi ma’lum qoidalar bo’yicha bajariluvchi har qanday hisoblash tizimi.

A.A. Markov ta’rifi. Algoritm – bu boshlang’ich berilganlardan izlangan natijaga olib keluvchi hisoblash jarayonini aniqlovchi aniq ko’rsatmalardir.

Algoritm tushunchasining turli ta’riflari bir qator talablarga javob berishi kerak:

- algoritm chekli sondagi elementar bajariluvchi ko’rsatmalardan iborat bo’lishi kerak;
- algoritm chekli sondagi qadamlardan iborat bo’lishi kerak;
- algoritm barcha boshlang’ich berilganlar uchun umumiy bo’lishi kerak;
- algoritm to’g’ri echimga olib kelishi kerak.

Nazorat savollari:

1. Algoritmilar nazariyasi faniga hissa qo’shgan olimlardan kimlarni bilasiz?
2. Algoritmilar nazariyasi fanining maqsadlari nimadan iborat?
3. Algoritmilar nazariyasi fanining vazifalari nimalardan iborat?
4. Algoritmilar nazariyasi fani qaysi yo’nalishlar bo’yicha rivojlanib keldi?
5. Algoritmilar nazariyasi fani yutuqlarining amaliy ahamiyati nimadan iborat?

Foydalanilgan adabiyotlar:

1. В.И.Игошин. Математическая логика и теория алгоритмов. Издательство Саратовского Университета, 1991. 209-211 с.
2. О.П.Кузнецов. Дискретная математика для инженера. М: Энергоатомиздат, 1982, 144-155 с.
3. Н.А.Криницкий, Г.А.Миронов, Г.Д. Фролов. Программирование и алгоритмические языки, М: Наука, 1979, 63-66 с.

4. Е.З. Любимский, В.В. Мартынюк, Н.П.Трифонов Программирование, М:Наука, 1980,13-17 с.
5. <http://structur.h1.ru/hash.htm>
6. <http://intsys.msu.ru/stuff/vnosov/theorald.htm#top>

2-Mavzu. Intuitiv algoritm tushunchasi.Intuitiv algoritm tushuchasini konkretlashtirish zarurati (2 soat)

Reja:

1. Intuitiv algoritm tushunchasi
2. Algoritm ob'ekti va uning tasviri
3. Algoritm alfaviti
4. Algoritmni konkretlashtirish zarururati

Kalit so'zlar: Algoritm, Ob'ekt, Tasvir, qadam, Alfavit, So'z.

Hisoblash mashinasining ishi algoritmni bajarishdan iborat buladi. SHuning uchun xisoblash mashinalarining umumiy imkoniyatlari kaysi muammo-masalalarni algoritm sifatida tasvirlash mumkinu, kaysilarini mumkin emasligiga boglik buladi.Matematikaning eng asosiy tushunchalaridan biri bulgan algoritm tushunchasi xisoblash masalalari paydo bulganidan ancha oldin vujudga kela boshlagan edi. Asrlar davomida kishilar intuitiv algoritm tushunchalaridan foydalanib kelgandlar. Bu tushunchani shunday ta'riflash mumkin:

Algoritm – bu koidalarning kat'iy va chekli sistemasi bulib, ba'zi ob'ektlar ustida bajariladigan amallarni aniklaydi va chekli kadamdan keyin kuyilgan maksadga olib kelishni ta'minlaydi.

Xususiyl xolda bunday koidalar sistemasi algoritm xisoblanadi, kachonki, ishning mazmuni bilan tanish bulmagan kishilarga uni kursatma sifatida berilganda , ularning barchasi bir xil xarakat kilsa.

Kadimgi Gresiyalik matematik Evklid 2 ta natural A va V sonlarning eng katta umumiy buluvchisini topish algoritmini taklif etdi. Uning ma'nosi kuyidagicha:

Katta sondan kichigini ayirish, natijani katta son urniga kuyish va ikkala son tenglashguncha bu amalni takrorlash. Ushbu teng sonlar izlangan natijadir.

Evklid algoritmida A va V sonlarning eng katta umumiy buluvchisi ushbu sonlar ayirmasining eng katta buluvchisi xamda ikkala A,V sonlarning xam umumiy eng katta buluvchisi bulish faktidan foydalanilgan.

Evklid algoritmining bu ifodasiga aniklik etishmaydi, shuning uchun uning konkretlashtirish zarur buladi.

Xakikiy Evklid algoritmi kuyidagicha:

1. A sonni birinchi son deb, V sonni ikkinchi son deb karalsin. 2-punktga utilsin.
2. Birinchi va ikkinchi sonlarni takkoslang. Agar ular teng bulsa, 5-punktga utilsin, aks xolda 3-punktga utilsin.
3. Agar birinchi son ikkinchi sondan kichik bulsa, ularning urni almashtirilsin. 4-punktga utilsin.
4. Birinchi sondan ikkinchi son ayirilsin va ayirma birinchi son deb xisoblansin. 2-punktga utilsin.
5. Birinchi sonni natija sifatida kabul kilinsin. Tamom.

Bu koidalar ketma-ketligi algoritmning tashkil etadi, chunki ularni bajargan ixtiyoriy ayirishni biladigan kishi ixtiyoriy sonlar jufti uchun eng katta umumiy buluvchini topa oladi.

Matematiklar uzok vaqtlar davomida algoritmning bunday ifodalaridan keng foydalanib turli xisblash algoritmilarini ishlab chikdilar.

Masalan, kvadrat va kubik tenglamalar ildizlarini topish algoritmilarini topildi. Asta-sekin olimlar kiyinroq masalalar ustida bosh kotirib, masalan, ixtiyoriy darajali algebraik tenglamalar ildizlarini topish algoritmilarini kidiradilar. Xatto, XVII –asrda Leybnis ixtiyoriy matematik masalani echin umumiy algoritmini tapishag urinib kurgan. Ammo bunga uxshash algoritmilarini kurishning iloji bulmagan va asta-sekin buning butunlay imkoni yuk degan xulosaga kelingan.

SHunday bulishiga karamay, algoritm tushunchasining anik tavsifi berilmagunga kadar, masalaning algoritmik echimsizligini isbotlash mumkin emas edi.

SHuning uchun juda dolzarb muammo – intuitiv algoritm tushunchasiga mos formal algoritm tushunchasini aniklash zaruriyati paydo buldi.

Intuitiv algoritm tushunchasi b'ekti sifatida ixtiyoriy narsa olinishi mumkin edi. Tabiiyki, ishni ob'ekt tushunchasini formallashtirishdan boshlash kerak edi.

Xisoblash algoritmilarida ish ob'ektlari sifatida sonlar olinadi. SHaxmat uyini algoritmda ob'ektlar bu – shaxmat figuralari va pozitsiyalaridir. Ishlab chikarish jarayonlarini algoritmiashtirishda lb'ektlar sifatida priborlar kursatishlari va boshkaruv tugmalari olinadi. Bunda klavishlarning shunday bosilish tartibi aniklanishi kerakki, ishlab chikarish jarayoni eng yaxshi bulsin.

Bu algoritmilar turli-tumanligiga bir necha misol xolos.

Ammo barcha xollarda real olam ob'ektlari bilan emas, ularning tasviri bilan ish kuradi.

Masalan, kushish algoritmi 26 va 57 sonli ob'ektlar bilan ish kurganda, u sonli nataja 83 ni beradi. Ammo biz algoritm lb'ekti deb, 5 ta simvoldan iborat tasvirni olishimiz mumkin:

$$26 + 57$$

natijani esa 2 ta belgidan iborat 83 tasviri bilan ifodalaymiz. Bunda 11 belgidan iborat tuplam mavjud deb olinadi.

$$\{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, + \}$$

Belgilarni xarflar deb, ularning tuplami esa alfavit deb ataladi. Umumiy xolda xarflar sifatida ixtiyoriy belgilar olinishi mumkin. Bunda ular uzaro turli xil bulishi va ularning chekliligi talab eliladi.

Demak, xarflar bu – ixtiyoriy belgilar; alfavit esa – uzaro turli bulgan xarflarning chekli tuplamidir.

Kandaydir alfavitdagi ixtiyoriy xarflarning chekli ketma-ketligi ushbu alfavitdagi suz deb ataladi. Masalan, kushishi algoritmidagi + belgisi bilan ajratilgan kushiluvchilarning tasvirlarini yigindini ifodalovchi tasvirga aylantiradi. Bunda suzlardagi xarflarning tartibi juda muxim buliB alfavitdagi tartibi esa muxim emasdir.

$\{A,B\}$ va $\{B,A\}$ alfavitlar bir xildir, ammo AV va VA suzlar xar xildir.

Suzlardagi xarflar soni suzning uzunligi deyiladi.

SHunday kilib, real olam ob'ektlarini turli alfavitlardagi suzlar bilan ifoda etish mumkin. Bu esa algoritmilarining ish ob'ektlari sifatida fakat suzlar kuooanilishi mumkin deb aytish imkonini beradi. Bundan shunday xulosaga kelish mumkin:

Algoritm bu – anik, chekli koidalarning shunday sistnmasiki: bu koidalar biror alfavit suzlarini, shu alfavitning boshka suzlariga almashtirsin.

Algoritm kulanilishi kerak bulgan suz – kirish suzi deb, algoritm kulanilishi natijasi bulgan suz esa – chikish suzi deyiladi. Algoritm kulanilishi kerak bulgan suzlar tuplami –

algoritmning kullanilish soxasi deb ataladi. Ammo barcha ob'ektlarni xam suzlar orkali ifoda etish mumkin emas.

Kushish algoritmini suzlar bilan ifoda etishni kurdik. Xuddi shuningdek, shaxmat uyini algoritmini xam suzlar bilan ifoda etish mumkin:

Oklar: Kpe5, Fd2, La7, If1

Koralar: KPb3, Ae4, Kb2, Kb4,nc2

Bunda shaxmat algoritmi natijasi figuralarning siljishi emas, tanlangan yurishning zuvi buladi:

$$Fd2 - e3 +$$

Ma'lumki, bunday belgilash bilan ixtiyoriy shaxmat situasiasini ifoda etish mumkin va yozuvlar asosida shaxmat doskasidagi donalar xolatini tiklash mumkin.

Shu va boshka kua misollar xar bir algoritm uchun mos alfavit tanlash mumkinligini isbotlaydi.

Ixtiyoriy alfavitni boshkasiga almashtirish mumkin. Bunday almashtirish kodlash deb ataladi. Masalan, telegrammalar Morze alifbosida uzatilgan. Bu alifbo 2 ta : «nukta» va «chizikcha» belgilaridan iborat. YAna bir alfavitga misrl : {0,1}.

Agar xar bir alfavitdan 2 ta belgili alfavitga utish va kodlangan belgilarni suzlarga aylantirish imkoni bulsa, ixtiyoriy algoritmi 0 va 1 lar ustidagi amallar ketma-ketligiga keltirish mumkin buladi.

Algoritm tushunchasi juda kadim zamonlardan shakllanib kelgan. SHunga karamay, asrimizning yarmiga kdar matematiklar bu ob'ekt xakida intuitiv karashlarga kanoatlanib kelganlar. Algoritm termini matematiklar tomonidan fakat konkret masalalarni echish bilan boglik xolda olinar edi.

XX asr boshida matematika asoslarida vujudga kelgan karama-karshiliklar va muammolar ularni xal etishga karatilgan turli konsepsiyalar va okimlarning vujudga kelishiga olib keldi. 20-yillarga kelib, effektiv xisoblash masalalari kundalang buldi.

Algoritm tushunchasining uzi matematik tadkikotlar ob'ekti bulib kolganligi uchun anik va kat'iy ta'rifga muxtij edi. Bundan tashkari EXM lar asrini yakinlashtiruvchi fizika va texnikaning rivojlanishi xam shuni takozo etar edi.

XX asr boshlarida matematiklar ba'zi ommaviy masalalar algoritmik echimga ega emas degan xulosaga kela boshladilar.

Biror bir ob'ektning mavjud emasligi ni kat'iy isbotlash uchun esa, ushbu ob'ektning anik ta'rifiga ega bulish kerak edi.

Uzluksizlik, egri chizik, sirt, uzunlik, yuza, xajm va boshka shu kabi tushunchalarni konkretlashtirish zarurati tugilganda xuddi ana shunday xolat vujudga kelgan edi.

Algoritm tushunchasini konkretlashtirish va urganish, ya'ni algoritmlar nazariyasi buyicha eng birinchi tadkikotlar 1936-37 yillarda A.Tyuring, E.Post, E.Erbran, K. Gedel, A.Markov, A.Church tomonidan bajarildi.

Algoritm tushunchasi buyicha bir kancha ta'riflar ishlab chikildi. Amma keyinchalik ularning teng kuchliligi aniklandi.

Nazorat uchun savollar:

1. Algoritm nima?
2. Intuitiv algorim formal algoritmdan fark kiladimi?
3. Algoritm ob'ekti nimadan iborat buladi?
4. Algoritm alfaviti deganda nimani tushunasiz?

Foydalanilgan adabiyotlar:

7. В.И.Игошин. Математическая логика и теория алгоритмов. Издательство Саратовского Университета, 1991. 209-211с.

8. О.П.Кузнецов. Дискретнауа математика длуа инженера. М:Энергоатомиздат, 1982, 144-155 с.
9. Н.А.Криницкий, Г.А.Миронов, Г.Д. Фролов. Программирование и алгоритмические узыки, М:Наука, 1979, 63-66 с.
10. Е.З. Любимский, В.В. Мартынюк, Н.П.Трифонов Программирование, М:Наука, 1980, 13-17 с.
11. <http://structur.h1.ru/hash.htm>
12. <http://intsys.msu.ru/stuff/vnosov/theorald.htm#top>

3-Mavzu.Algoritmlar, ularning xossalari. Berilish usullari va strukturalari (2 soat)

1. Reja:
2. Algoritmning asosiy xossalari
3. Algoritmning tasvirlash usullari
4. Chiziqli algoritmlar
5. Tarmoqlanuvchi algoritmlar
6. Takrorlanuvchi algoritmlar
7. Algoritm ijrosini tekshirish

Kalit so'zlar: Aniqlik, Diskretlik, Ommaviylik, Tushunarlilik, Natijaviylik, Blok-sxema, Algoritmik notasiya

Yuqorida qayd qilganimizdek, qo'yilgan biror masalani EHMda yechish uchun, avval uning matematik modelini, keyin algoritmini va programmasini tuzish kerak bo'ladi. Bu uchlikda algoritm bloki muhim ahamiyatga ega. Endi algoritm tushunchasining ta'rifi va xossalarini bayon qilamiz.

Algoritm bu oldimizga qo'yilgan masalani yechish zarur bo'lgan amallar ketma-ketligidir.

Algoritm so'zi va tushunchasi IX asrda yashab ijod etgan buyuk alloma Muhammad al-Xorazmiy nomi bilan uzviy bog'liq. Algoritm so'zi Al-Xorazmiy nomini Yevropa olimlari tomonidan buzib talaffuz qilinishidan yuzaga kelgan. Al-Xorazmiy birinchi bo'lib o'nlik sanoq sistemasining tamoyillarini va undagi to'rtta amallarni bajarish qoidalarini asoslab bergan.

Algoritmning asosiy xossalari. Algoritmning 5-ta asosiy xossasi bor:

Diskretlilik (Cheklilik). Bu xossaning mazmuni algoritmlarni doimo chekli qadamlardan iborat qilib bo'laklash imkoniyati mavjudligida. Ya'ni uni chekli sondagi oddiy ko'rsatmalar ketma-ketligi shaklida ifodalash mumkin. Agar kuzatilayotgan jarayonni chekli qadamlardan iborat qilib qo'llay olmasak, uni algoritm deb bo'lmaydi.

Tushunarlilik. Biz kundalik hayotimizda berilgan algoritmlar bilan ishlayotgan elektron soatlar, mashinalar, dastgohlar, kompyuterlar, turli avtomatik va mexanik qurilmalarni kuzatamiz.

Ijrochiga tavsiya etilayotgan ko'rsatmalar, uning uchun tushinarli mazmunda bo'lishi shart, aks holda ijrochi oddiygina amalni ham bajara olmaydi. Undan tashqari, ijrochi har qanday amalni bajara olmasligi ham mumkin.

Har bir ijrochining bajarishi mumkin bo'lgan ko'rsatmalar yoki buyruqlar majmuasi mavjud, u ijrochining ko'rsatmalar tizimi (sistemi) deyiladi. Demak, ijrochi uchun berilayotgan har bir ko'rsatma ijrochining ko'rsatmalar tizimiga mansub bo'lishi lozim.

Ko'rsatmalarni ijrochining ko'rsatmalar tizimiga tegishli bo'ladigan qilib ifodalay bilishimiz muhim ahamiyatga ega. Masalan, quyi sinfning a'lochi o'quvchisi "son kvadratga oshirilsin" degan ko'rsatmani tushinmasligi natijasida bajara olmaydi, lekin "son o'zini o'ziga ko'paytirilsin" shaklidagi ko'rsatmani bemalol bajaradi, chunki u ko'rsatma mazmunidan ko'paytirish amalini bajarish kerakligini anglaydi.

Aniqlik. Ijrochiga berilayotgan ko'rsatmalar aniq mazmunda bo'lishi zarur. Chunki ko'rsatmadagi noaniqliklar mo'ljaldagi maqsadga erishishga olib kelmaydi. Odam uchun tushinarli bo'lgan "3-4 marta silkitilsin", "5-10 daqiqa qizdirilsin", "1-2 qoshiq solinsin", "tenglamalardan biri yechilsin" kabi noaniq ko'rsatmalar robot yoki kompyuterni qiyin ahvolga solib qo'yadi.

Bundan tashqari, ko'rsatmalarning qaysi ketma-ketlikda bajarilishi ham muhim ahamiyatga ega. Demak, ko'rsatmalar aniq berilishi va faqat algoritmda ko'rsatilgan tartibda bajarilishi shart ekan.

Ommaviylik. Har bir algoritm mazmuniga ko'ra bir turdagi masalalarning barchasi uchun ham o'rinli bo'lishi kerak. YA'ni masaladagi boshlang'ich ma'lumotlar qanday bo'lishidan qat'iy nazar algoritm shu xildagi har qanday masalani yechishga yaroqli bo'lishi kerak. Masalan, ikki oddiy kasrning umumiy mahrajini topish algoritmi, kasrlarni turlicha o'zgartirib bersangiz ham ularning umumiy mahrajlarini aniqlab beraveradi. Yoki uchburchakning yuzini topish algoritmi, uchburchakning qanday bo'lishidan qat'iy nazar, uning yuzini hisoblab beraveradi.

Natijaviylik. Har bir algoritm chekli sondagi qadamlardan so'ng albatta natija berishi shart. Bajariladigan amallar ko'p bo'lsa ham baribir natijaga olib kelishi kerak. Chekli qadamdan so'ng qo'yilgan masala yechimga ega emasligini aniqlash ham natija hisoblanadi. Agar ko'rilayotgan jarayon cheksiz davom etib natija bermasa, uni algoritm deb atay olmaymiz.

Algoritmning tasvirlash usullari. Yuqorida ko'rilgan *misollarda* odatda biz masalani yechish algoritmini so'zlar va matematik formulalar orqali ifodaladik. Lekin algoritm boshqa ko'rinishlarda ham berilishi mumkin. Biz endi algoritmning eng ko'p uchraydigan turlari bilan tanishamiz.

Algoritmning so'zlar orqali ifodalanishi. Bu usulda ijrochi uchun beriladigan har bir ko'rsatma jumlar, so'zlar orqali buyruq shaklida beriladi.


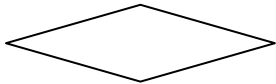
Algoritmning formulalar bilan berilish usulidan matematika, fizika, kimyo kabi aniq fanlardagi formulalarni o'rganishda foydalaniladi. Bu usulni ba'zan analitik ifodalash deyiladi.

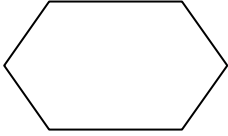

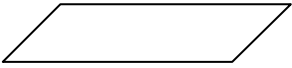
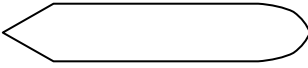



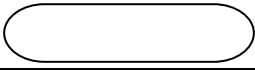
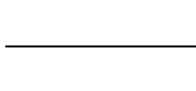
3. Algoritmning grafik shaklida tasvirlanishida algoritmlar maxsus geometrik figuralar yordamida tasvirlanadi va bu grafik ko'rinishi blok-sxema deyiladi.

4. Algoritmning jadval ko'rinishda berilishi. Algoritmning bu tarzda tasvirlanishdan ham ko'p foydalanamiz. Masalan, maktabda qo'llanib kelinayotgan to'rt xonali matematik jadvallar yoki turli xil lotereyalar jadvallari. Funksiyalarning grafiklarini chizishda ham algoritmning qiymatlari jadvali ko'rinishlaridan foydalanamiz. Bu kabi jadvallardan foydalanish algoritmлари sodda bo'lgan tufayli ularni o'zlashtirib olish oson.

Yuqorida ko'rilgan algoritmning tasvirlash usullarining asosiy maqsadi, qo'yilgan masalani yechish uchun zarur bo'lgan amallar ketma-ketligining eng qulay holatini aniqlash va shu bilan odam tomonidan programma yozishni yanada osonlashtirishdan iborat. Aslida programma ham algoritmning boshqa bir ko'rinishi bo'lib, u insonning kompyuter bilan muloqotini qulayroq amalga oshirish uchun mo'ljallangan.

Blok-sxemalarni tuzishda foydalaniladigan asosiy sodda geometrik figuralar quyidagilardan iborat:

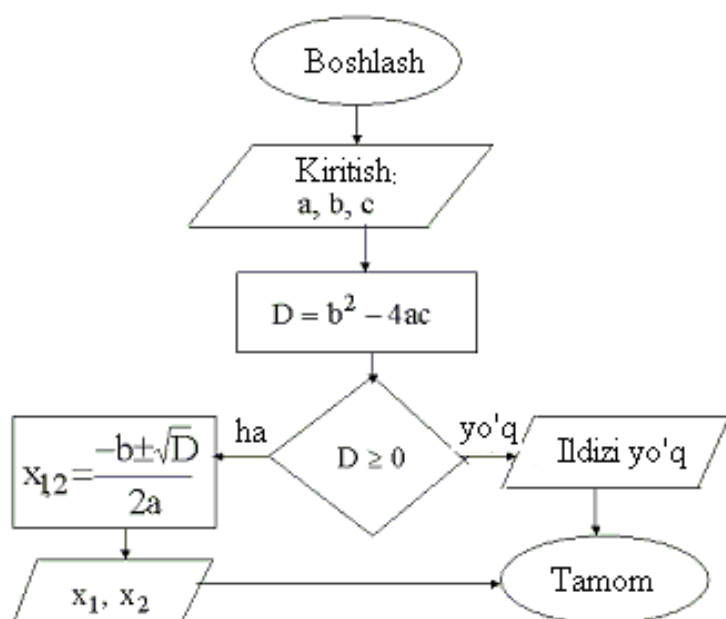
Nomi	Belgilanishi	Bajaradigan vazifasi
Jarayon		Bir yoki bir nechta amallarni bajarilishi natijasida ma'lumotlarning uzgarishi
Karor		Biror shartga bog'liq ravishda algoritmning bajarilish yunalishini tanlash

SHakl uzgartirish		Dasturni uzgartiruvchi buyruk yoki buyruklar turkumini uzgartirish amalini bajarish
Avval aniklangan jarayon		Oldindan ishlab chikilgan dastur yoki algoritmdan foydalanish
Kiritish CHikarish		Axborotlarni kayta ishlash mumkin bulgan shaklga utkazish yoki olingan natijani tasvirlash
Display		EXMga ulangan displaydan axborotlarni kiritish yoki chikarish
Xujjat		Axborotlarni kogozga chikarish yoki kogozdan kiritish
Axborotlar okimi chizigi		Bloklar orasidagi boglanishlarni tasvirlash
Boglagich		Uzilib kolgan axborot okimlarini ulash belgisi
Boshlash Tugatish		Axborotni kayta ishlashni boshlash, vaktincha yoki butunlay tuxtatish
Izox		Bloklarga tegishli turli xildagi tushuntirishlar

Blok-sxemalar bilan ishlashni yaxshilab o'zlashtirib olish zarur, chunki bu usul algoritmlarni ifodalashning qulay vositalaridan biri bo'lib programma tuzishni osonlashtiradi, programmalash qobiliyatini mustahkamlaydi. Algoritmik tillarda blok - sxemaning asosiy strukturalariga maxsus operatorlar mos keladi.

Shuni aytish kerakni, blok-sxemalardagi yozuvlar odatdagi yozuvlardan katta farq qilmaydi.

Misol sifatida $ax^2+bx+c=0$ kvadrat tenglamani yechish algoritmining blok-sxemasi quyida keltirilgan.



1-rasm. Kvadrat tenglamani yechish algoritmi

Chiziqli algoritmlar. Har qanday murakkab algoritmnı ham uchta asosiy struktura yordamida tasvirlash mumkin. Bular ketma-ketlik, ayri va takrorlash strukturalaridir. Bu strukturalar asosida chiziqli, tarmoqlanuvchi va takrorlanuvchi hisoblash jarayonlarining algoritmlarini tuzish mumkin. Umuman olganda, algoritmlarni shartli ravishda quyidagi turlarga ajratish mumkin:

chiziqli algoritmlar;

tarmoqlanuvchi algoritmlar;

takrorlanuvchi yoki siklik algoritmlar;

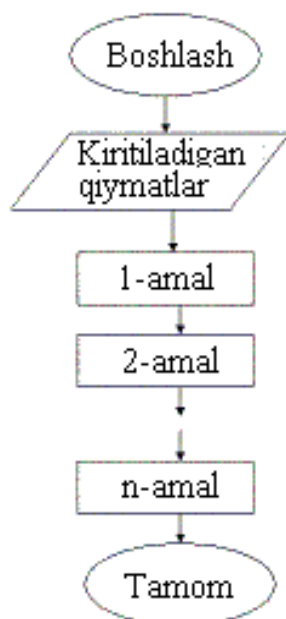
ichma-ich joylashgan siklik algoritmlar;

rekurrent algoritmlar;

takrorlanishlar soni oldindan no'malum algoritmlar;

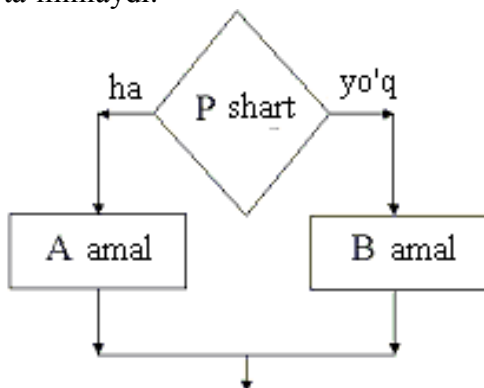
ketma-ket yaqinlashuvchi algoritmlar.

Faqat ketma-ket bajariladigan amallardan tashkil topgan algoritmlarga-chiziqli algoritmlar deyiladi. Bunday algoritmnı ifodalash uchun ketma-ketlik strukturasi ishlatiladi. Strukturada bajariladigan amal mos keluvchi shakl bilan ko'rsatiladi. Chiziqli algoritmlar blok-sxemasining umumiy strukturasi quyidagi ko'rinishda ifodalash mumkin:



2-rasm. Chiziqli algoritmlar blok - sxemasining umumiy strukturası

Tarmoqlanuvchi algoritmlar. Agar hisoblash jarayoni biror bir berilgan shartning bajarilishiga qarab turli tarmoqlar bo'yicha davom ettirilsa va hisoblash jarayonida har bir tarmoq faqat bir marta bajarilsa, bunday hisoblash jarayonlariga tarmoqlanuvchi algoritmlar deyiladi. Tarmoqlanuvchi algoritmlar uchun ayri strukturasi ishlatiladi. Tarmoqlanuvchi strukturasi berilgan shartning bajarilishiga qarab ko'rsatilgan tarmoqdan faqat bittasining bajarilishini ta'minlaydi.



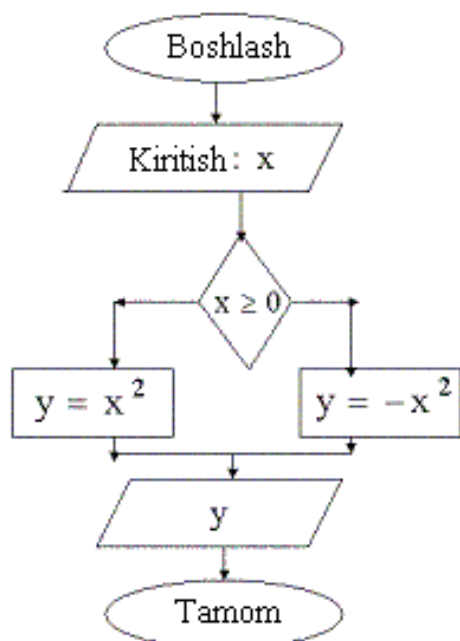
3-rasm. Tarmoqlanishning umumiy ko'rinishi

Berilgan shart romb orqali ifodalanadi, r-berilgan shart. Agar shart bajarilsa, "ha" tarmoq bo'yicha a amal, shart bajarilmasa "yo'q" tarmoq bo'yicha b amal bajariladi.

Tarmoqlanuvchi algoritmga tipik *misol* sifatida quyidagi sodda *misol*ni qaraylik.

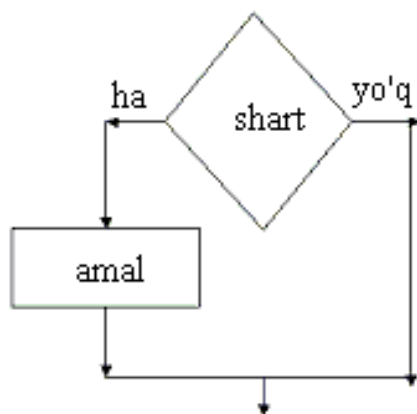
1- Misol:
$$Y = \begin{cases} x^2 & \text{agar } x \geq 0 \\ -x^2 & \text{agar } x < 0 \end{cases}$$

Berilgan x ning qiymatiga bog'lik holda, agar u musbat bo'lsa «ha» tarmoq bo'yicha $y = x^2$ funksiyaning qiymati, aks holda $y = -x^2$ funksiyaning qiymati hisoblanadi.



4-rasm. Interval ko'rinishidagi funksiya qiymatini hisoblash algoritmi

Ko'pgina masalalarni yechishda, shart asosida tarmoqlanuvchi algoritmlarning ikkita tarmog'idan bittasining, ya'ni yoki «ha» yoki «yo'q» ning bajarilishi yetarli bo'ladi. Bu holat tarmoqlanuvchi algoritmnıng xususiy holi sifatida aylanish strukturasi deb atash mumkin. Aylanish strukturasi quyidagi ko'rinishga ega:



5-rasm. Aylanish strukturasi umumiy ko'rinishi

Takrorlanuvchi algoritmlar. Agar biror masalani yechish uchun tuzilgan zarur bo'lgan amallar ketma-ketligining ma'lum bir qismi biror parametrga bog'liq ko'p marta qayta bajarilsa, bunday algoritmlar takrorlanuvchi algoritmlar yoki siklik algoritmlar deyiladi. Takrorlanuvchi algoritmlarga tipik *misol* sifatida odatda qatorlarning yig'indisi yoki ko'patmasini hisoblash jarayonlarini qarash mumkin. Quyidagi yig'indini hisoblash algoritmini tuzaylik.

$$S = 1 + 2 + 3 + \dots + N = \sum_{i=1}^N i$$

Bu yig'indini hisoblash uchun $i=0$ da $S=0$ deb olamiz va $i=i+1$ da $S=S+i$ ni hisoblaymiz. Bu yerda birinchi va ikkinchi qadamlar uchun yig'indi hisoblandi va keyingi qadamda i parametriga yana bittaga orttiriladi va navbatdagi raqam avvalgi yig'indi S ning ustiga qo'shiladi va bu jarayon shu tartibda to $i < N$ sharti bajarilmaguncha davom ettiriladi va natijada izlangan yig'indiga ega bo'lamiz. Bu fikrlarni quyidagi algoritmlar sifatida ifodalash mumkin:

N –berilgan bo'lsin,

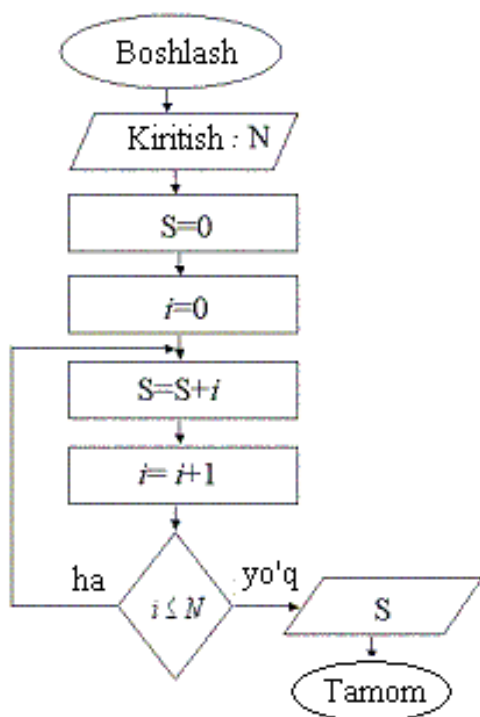
$i=0$ berilsin,

$S=0$ berilsin,

$i=i+1$ hisoblansin,

$S=S+i$ hisoblansin,

$i < N$ tekshirilsin va bu shart bajarilsa, 4-satrga qaytilsin, aks holda keyingi qatorga o'tilsin, S ning qiymati chop etilsin.

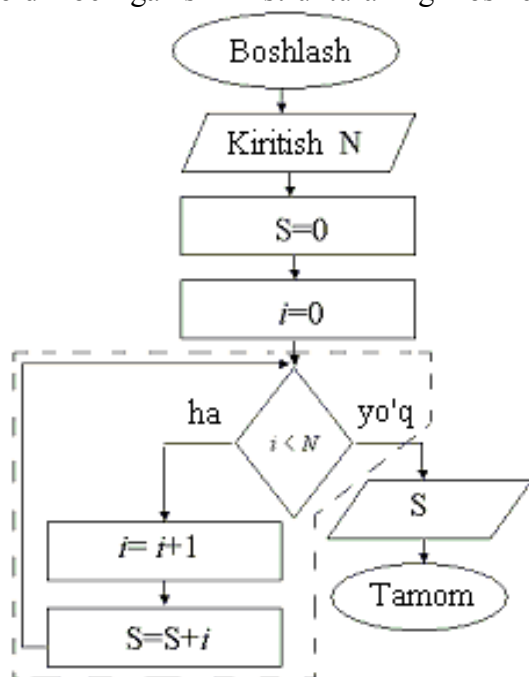


6-rasm. 1 dan n gacha bo'lgan sonlar yig'indisini hisoblash algoritmi

Yuqorida keltirilgan algoritm va blok sxemadan ko'rinib turibdiki amallar ketma-ketligining ma'lum qismi parametr i ga nisbatan N marta takrorlanayapti.

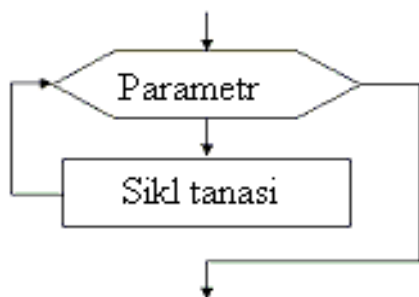
Yuqorida ko'rilgan yig'indi blok sxemalaridagi takrorlanuvchi qismlariga (aylana ichiga olingan) quyidagi sharti keyin berilgan siklik struktura mos kelishini ko'rish mumkin.

Yuqoridagi blok sxemalarda shartni oldin tekshiriladigan holatda chizish mumkin edi. Masalan, yig'indining algoritmini qaraylik. Bu blok sxemaning takrorlanuvchi qismiga quyidagi, sharti oldin berilgan siklik strukturaning mos kelishini ko'rish mumkin.



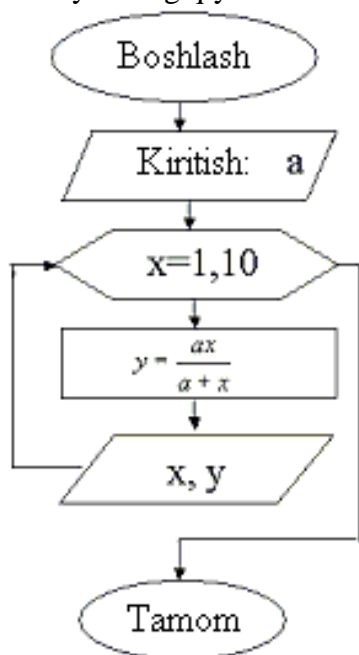
7-rasm. 1 dan n gacha bo'lgan sonlar yig'indisini hisoblash algoritmi

Blok sxemalarining takrorlanuvchi qismlarini, quyidagi parametrli takrorlash strukturasi ko'rinishida ham ifodalash mumkin.



8-rasm. Parametrli takrorlash operatorining umumiy ko‘rinishi

Parametrli takrorlash operatoriga *misol* sifatida berilgan $x=1,2,3,\dots,10$ larda $y = \frac{ax}{a+x}$ funksiyasining qiymatlarini hisoblash blok sxemasini qarash mumkin.



9-rasm. Parametrli takrorlash operatoriga doir algoritm

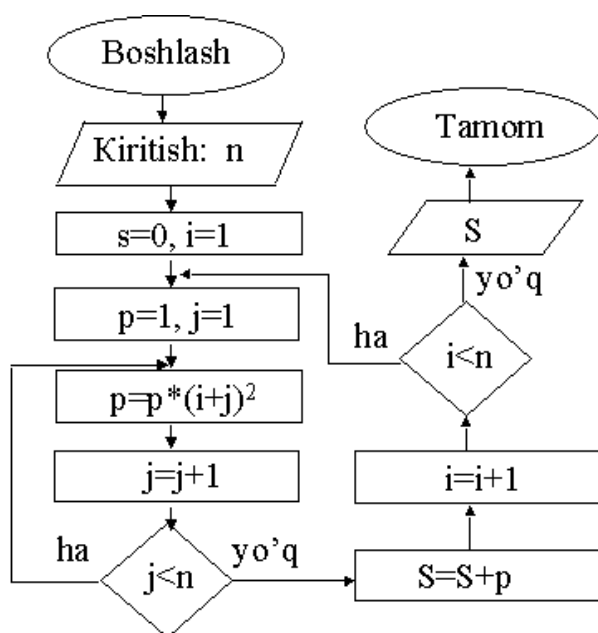
Ichma-ich joylashgan siklik algoritmlar. Ba‘zan, takrorlanuvchi algoritmlar bir nechta parametrlarga bog‘liq bo‘ladi. Odatda bunday algoritmlarni ichma-ich joylashgan algortmlar deb ataladi.

Misol sifati berilgan $n \times m$ o‘lchovli a_{ij} –matritsa elementlarining yig‘indisini hisoblash masalasini qaraylik.

$$S = \sum_{i=1}^n \prod_{j=1}^n (i+j)^2$$

Bu yig‘indi hisoblash uchun, i ning har bir qiymatida j bo‘yicha

ko‘paytmani hisoblab, avval yig‘indi ustiga ketma-ket qo‘shib borish kerak bo‘ladi. Bu jarayon quyidagi blok–sxemada aks ettirilgan. Bu yerda i -tashqi sikl - yig‘indi uchun, j -esa ichki sikl-ko‘paytmani hosil qilish uchun foydalanilgan.

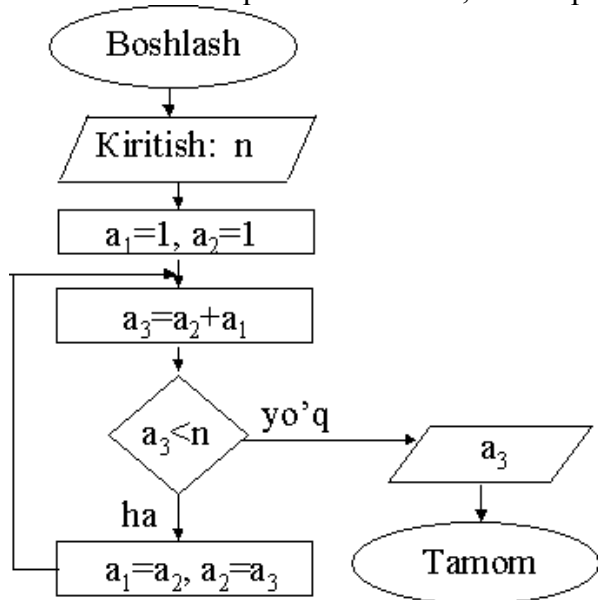


10-rasm. Ichma-ich joylashgan siklik algoritmgaga doir blok-sxema

Rekurrent algoritmlar. Hisoblash jarayonida ba'zi bir algoritmlarning o'ziga qayta murojaat qilishga to'g'ri keladi. O'ziga-o'zi murojaat qiladigan algoritmlarga rekkurent algoritmlar yoki rekursiya deb ataladi.

Bunday algoritmgaga *misol* sifatida Fibonachchi sonlarini keltirish mumkin. Ma'lumki, Fibonachchi sonlari quyidagicha aniqlangan.

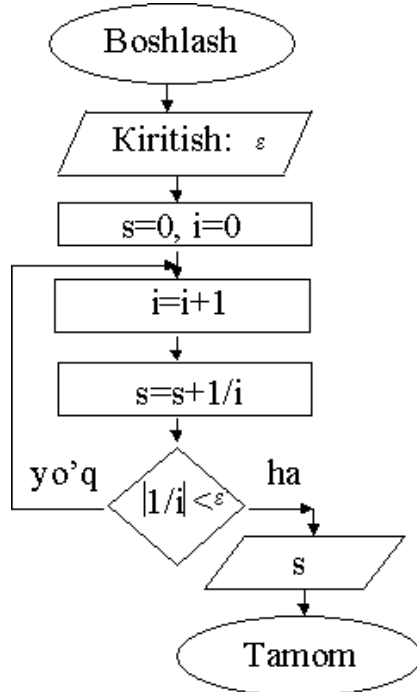
$a_0 q a_1 q 1, a_i q a_{i-1} + a_{i-2} \quad i q 2, 3, 4, \dots$ Bu rekurrent ifoda algoritmgaga mos keluvchi blok-sxema 2.15-rasmda keltirilgan. Eslatib o'tamiz formuladagi i -indeksga hojat yo'q, agar Fibonachchi sonining nomerini ham aniqlash zarur bo'lsa, birorta parametr-kalit kiritish kerak bo'ladi.



11-rasm. Fibonachchi sonlarining n - hadini hisoblash algoritmi.

Amalda shunday bir masalalar uchraydiki, ularda takrorlanishlar soni oldindan berilmagan-noma'lum bo'ladi. Ammo, bu jarayonni tugatish uchun biror bir shart berilgan bo'ladi.

Masalan, quyidagi $S = 1 + \frac{1}{2} + \frac{1}{3} + \dots = \sum_{i=1}^{\infty} \frac{1}{i}$ qatorda nechta had bilan chegaralanish berilmagan. Lekin qatorni ε aniqlikda hisoblash zarur bo'ladi. Buning uchun $\left| \frac{1}{i} \right| < \varepsilon$ shartni olish mumkin.



12-rasm. Takrorlanishlar soni oldindan no'malum bo'lgan algoritmlarga doir blok-sxema.

Ketma-ket yaqinlashuvchi yoki iteratsion algoritmlar. Yuqori tartibli algebrayik va transsendent tenglamalarni yechish usullari yoki algoritmlari ketma-ket yaqinlashuvchi – iteratsion algoritmlarga *misollar* bo'la oladi. Ma'lumki, transsendent tenglamalarni yechishning quyidagi asosiy usullari mavjud:

- Urinmalar usuli (Nyuton usuli),
- Ketma-ket yaqinlashishi usuli,
- Vatarlar usuli,
- Teng ikkiga bo'lish usuli.

Bizga

$$f(x)=0 \quad (1)$$

transsendent tenglama berilgan bo'lsin. Faraz qilaylik bu tenglama $[a,b]$ oraliqda uzluksiz va $f(a) \cdot f(b) < 0$ shartni qanoatlantirsin. Ma'lumki, bu holda berilgan tenglama $[a,b]$ orilaqda kamida bitta ildizga ega bo'ladi va u quyidagi formula orqali topiladi.

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)} \quad n = 0, 1, 2, \dots \quad (2)$$

Boshlang'ich X_0 qiymat $f(x_0) \cdot f''(x_0) < 0$ shart asosida tanlab olinsa, (2) iteratsion albatta yaqinlashadi. Ketma-ketlik

$$|X_{n+1} - X_n| < \varepsilon$$

shart bajarilgunga davom ettiriladi.

Berilgan musbat a haqiqiy sondan kvadrat ildiz chiqarish algoritmi tuzilsin.

Bu masalani yechish uchun kvadrat ildizni x deb belgilab olib,

$$\sqrt{a} = x \quad (3)$$

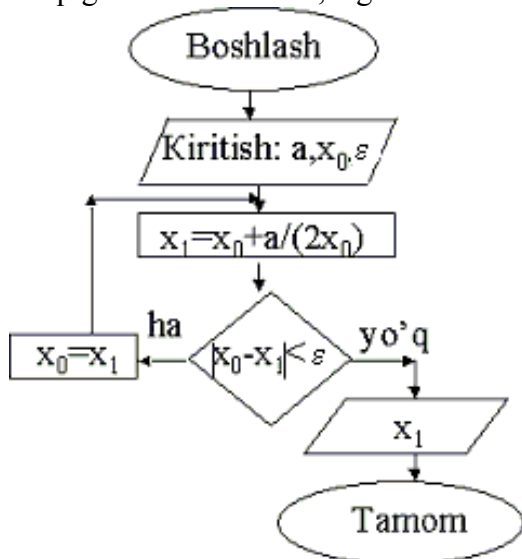
ifodalash yozib olamiz. U holda (1) tenglamaga asosan

$$f(x) = x^2 - a \quad (4)$$

ekanligini topish mumkin (4) ifodani (2) ga qo'yib, quyidagi rekurrent formulani topish mumkin:

$$X_{n+1} = \frac{1}{2} \left(X_n + \frac{a}{2X_n} \right) \quad (5)$$

Bu formulaga mos blok-sxema 2.18-rasmda keltirilgan. ε - kvadrat ildizni topishning berilgan aniqligi. Eslatib o'tamiz, algoritmda indeksli o'zgaruvchilarga zarurat yo'q.

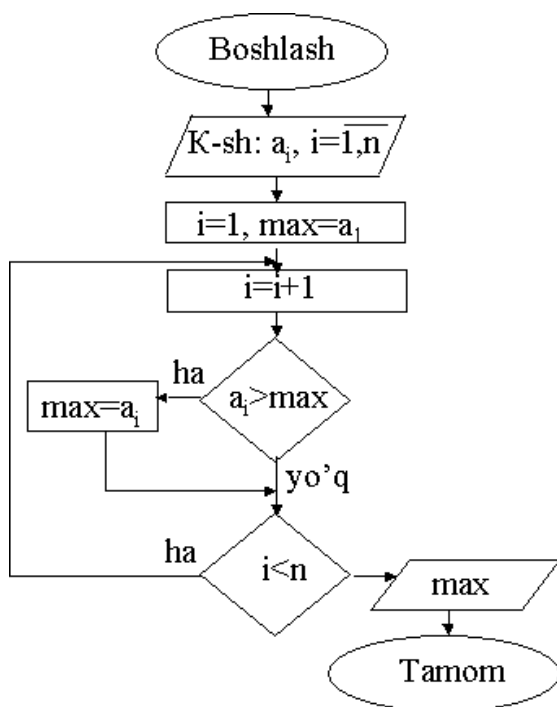


13-rasm. Berilgan musbat a haqiqiy sondan kvadrat ildiz chiqarish algoritmi (iteratsion algoritmgacha doir blok-sxema).

Algoritm ijrosini tekshirish. Kompyuter uchun tuzilgan algoritm ijrochisi-bu kompyuterdir. Biror programmalash tilida yozilgan algoritm kodlashtirilgan oddiy ko'rsatmalar ketma-ketligiga o'tadi va mashina tomonidan avtomatik ravishda bajariladi. Metodik nuqtayi-nazardan qaraganda algoritmning birinchi ijrochisi sifatida o'quvchining o'zini olish muhim ahamiyatga ega. O'quvchi tomonidan biror masalani yechish algoritmi tuzilganda bu algoritmni to'g'ri natija berishini tekshirish juda muhimdir. Buning yagona usuli o'quvchi tomonidan algoritmni turli boshlang'ich ma'lumotlarda qadamba - qadam bajarib (ijro etib) ko'rishdir. Algoritmni bajarish natijasida xatolar aniqlanadi va to'g'rilanadi. Ikkinchi tomonidan, masalani yechishga qiynalayotgan o'quvchi uchun tayyor algoritmni bajarish – masalani yechish yo'llarini tushunishga xizmat qiladi.

Algoritm ijrosini quyidagi *misolda* ko'raylik.

Berilgan a_i , $i = 1, n$ sonlarning eng kattasini topish algoritmini tuzaylik. Buning uchun, berilgan sonlardan birinchisi a_1 ni $i = 1$ eng katta qiymat deb faraz qilaylik va uni max nomli yangi o'zgaruvchiga uzataylik: $max = a_1$. Parametr i ning qiymatini bittaga oshirib, ya'ni $i = i + 1$ a_1 ni a_2 bilan taqqoslaymiz va qaysi biri katta bo'lsa uni max o'zgaruvchisiga uzatamiz va jarayonni shu tarzda to $i = n$ bo'lguncha davom ettiramiz. Bu fikrlar quyidagi blok-sxemada o'z aksini topgan.



14-rasm. Vektor elementlarining eng kattasini topish algoritmi.

Endi bu blok-sxema yoki algoritmnining ijrosini $n = 3$ $a_1 = 3$, $a_2 = 5$, $a_3 = 1$ aniq sonlarda ko'rib o'taylik:

$i=1$ da $max=3$ bo'ladi.

$i=i+1=2$ ni topamiz,

$a_2 > max$, ya'ni $5 > 3$ ni tekshiramiz, shart bajarilsa, $max=5$ bo'ladi.

$i < n$, ya'ni $2 < 3$ ni tekshiramiz. Shart bajarilsa, i ni yana bittaga oshiramiz, va $i=3$ bo'ladi, va

$a_3 > max$, ya'ni $1 > 5$, ni tekshiramiz. Shart bajarilmadi, demak, keyingi

$i < n$ shartni, ya'ni $3 < 3$ ni tekshiramiz. Shart bajarilmadi. Demak $max=5$ chop etiladi. Biz blok-sxemani tahlil qilish davomida uning to'g'riligiga ishonch hosil qildik. Endi ixtiyoriy n lar uchun bu blok-sxema bo'yicha eng katta elementni topish mumkin.

Nazorat savollari:

1. Intuitiv algoritm tushunishining mohiyati nimada?
2. Algoritmnining intuitiv ta'riflaridan birini keltiring?
3. Formal algoritm deganda nimani tushunamiz?
4. Algoritm ob'ekti deganda nimani tushunamiz?
5. Algoritm ob'ektining tasviri deganda nimani tushunamiz?

Foydalanilgan adabiyotlar:

1. Informatika va programmalash. O'quv qo'llanma. Mualliflar: A.A.Xaldjigitov, Sh.F.Madraximov, U.E.Adambayev, O'zMU, 2005 yil, 33-52 b.
2. Pascal tilida programmalash bo'yicha masalalar to'plami. O'quv qo'llanma. Mualliflar: A.A.Xaldjigitov, Sh.F.Madraximov, A.M.Ikromov, S.I.Rasulov, O'zMU, 2005 yil, 23-45 b.
3. Amaliy matematikadan kirish leksiyalari. A.НТихонов, Д.П.Костомаров. Toshkent. O'qituvchi, 1987. 27-36 b.
4. Вычислительная техника и программирование. А.В.Петров. М.: Просвещение, 1991. 34-58 с.

4-Mavzu. Tyuring mashinasi tushunchasi (2 soat)

Reja:

- 1. Tyuring mashinasi va uning sxemasi**
- 2. Tyuring mashinasi xolatlari va uning dasturi**
- 3. Algoritmga berilgan Tyuring ta'rifi**
- 4. Sonni 1 taga oshirib beruvchi Tyuring mashinasi.**
- 5. Shtrixlar sonini hisoblab, ular o'rniga yig'indini yozadigan Tyuring mashinasi.**

Kalit so'zlar: Abstrakt mashina, Kod, Kirish, CHiqish, Lenta, Avtomat, Son, Yacheyka, Dastur, Holatlar

Asrimizning 30-40-yillariga kelib, algoritmning formal ta'riflari keltirila boladi. Algoritmni formal ta'riflagan eng birinchi matematiklardan biri ingliz olimi A.Tyuring buldi. U 1936 yilda uziga xos abstrakt mashina sxemasini takdim etib, ushbu mashina bajarishi mumkin bulgan narsalarni – algoritm deb atash kerak, deb taklif kildi. Bu ta'rifdan Tyuring mashinasi bajara olmaydigan narsalarning algoritm emasligi kelib chikadi. Boshkacha aytganda, Tyuring amallar bajarilishi koidalarini konstruksiya ishini tasvirlash yordamida formallashtiridi.

Xisoblash mashinalari xam algoritmlarni bajaruvchi konstruksiyalardir, ammo ular Tyuring mashinasidan farkli real konstruksiyalardir. Tyuring mashinasi abstrakt bulib, u xech kachon amalda bulmagan.

Shuning uchun Tyuring mashinasi urniga algoritmlarni bajara oladigan maxsus usullra topishimizga tugri keladi.

Masalan, mashina urniga uning vazifalarini odam bajarsin deb faraz kilaylik. Tyuring mashinasi tushunchasidan foydalanishning maksadi shuki, ushbu xayoliy mashina xakida gapirib, biz turli masalarning echimini algoritmi bor-yukligini aniklashimiz mumkin. SHundan kelib chikib, Tyuring iloji boricha soddarak, ammo universal bulgan algoritmik sxemani kidirdi.

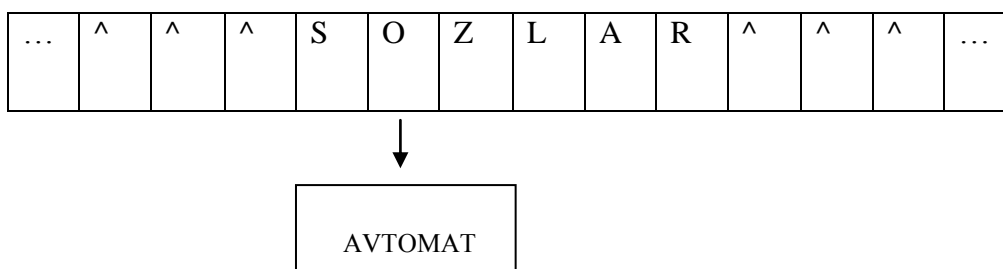
Xisoblash mashinasi xakida gap ketganda esa, aksincha bizga uning kulayligi, imkoniyatlarining boyligi muximrokdir; odamga u bilan mulokot kilish oson bulishi talab etiladi.

Tyuring mashinasining xisoblash mashinalaridan prinsipial farki shundaki, uning xotira kurulmasi kanchalik ulkan bulmasin, baribir u cheklidir. Tyuring mashinasini uning cheksiz xotirasi tufayli fizik reallashtirishning iloji yuk. Bu ma'noda Tyuring mashinasi xar kaday xisoblash mashinasidan kudratlirokdir.

Tyuring mashinasining lentasi yacheykalarga bulingandir. Xar bir yacheykada kadaydir alfavitning 1 ta xarfi joylashadi. Agar yacheyka bush bulsa, biz uni \wedge belgisi bilan belgilaymiz. Alfavitlar turli-tuman bulishi mumkin. Ammo xar bir Tyuring mashinasi uchun bitta alfavit tanlanadi. Tyuring mashinasida lentadan tashkari maxsus avtomat kurulma mavjud bulib, bu vavtomat lenta buylab xarakatlanib, navbat bilan yacheyka ichidagilarni «kuzdan kechirishi» mumkin.

«Kirish suzi» lentaning ketma-ket joylashgan yacheykalarida xarma-xarf joylashadi va chekli sondagi yacheykalarni egallaydi. Kirish suzidan chapda va ungda bush yacheykalar joylashadi. Kuyida Tyuri ng mashinasining sxematik tasvirini keltiramiz:

Cheksiz lenta



Shunday kilib, avtomat xar bir yurishda bitta yacheykani «kuradi». Bundan tashkari, u bir necha xolatlarining birini kabul qilishi mumkin: $q_1, q_2, q_3, \dots, q_k$. Avtomat knday (q_i) xolatda bulganligiga, xamda kaysi S_i yacheykani tekshirib turganligiga boglik xolda (S_i, q_i) turli amallar bajarishi mumkin:

- tekshirilayotgan yacheykaga yangi xarf kiritish;
- lenta buylab bir yacheykaga ungga siljish;
- yangi xolatga utish;

Tyuring mashinasining uch iurdagi amallari xr bir navbatdagi (S_i, q_i) uchun xar bittasidan bittadan bajariladi.

Uning ishlashi uchun barcha vazifalarni kuyidagi kurinishdagi dastur yordamida ifodalash mumkin:

	^	S1	...	S _i	...						
--	---	----	-----	----------------	-----	--	--	--	--	--	--

Q1

Q2

...

Q_i

...

Q_k

L
Si N Qm
P

Dasturining xar bir katagida berilgan xolatda turib, berilgan xarfni «ukiganda» nima ish kilinishi kerakligi xakidagi axborot berilishi kerak. Umumiy xolda q_i xolatda turib, S_i xarfni «kurib» Tyuring mashinasi shu yacheykaga berilgan $S1$ xarfni kiritadi, sungra u lenta buylab xarakat kilib, bir kadam chapga siljiydi (bunda u ungga siljishi yoki kuzgalmasligi xam mumkin). Ushbu uch xolatni belgilash uchun katakka L, N, P xarflaridan biri kiritiladi. SHunday sung avtomat q_m xolatga utadi. Endi avtomatning keyingi vazifasi tavsifini dasturining m – satridan kidirish kerak buladi. Axborotlar m - satr bilan avtomat siljigandan keyin aniklangan xarfga mos keluvchi ustun bilan kesishgan katakda joylashadi.

SHunday kilib, avtomat lenta buylab ungga yoki chapga siljiydi, yoki uz urnida koladi va xar safar nimani yozish, kaerga xarakat kilish va kaysi xolatni kabul qilishi xal etadi. Agar avtomatga e'tibor bermay, fakat lentani kuzatsak, undagi xarflar doimo uzgarib turganligini kuramiz. Ba'zi bush yacheykalarda xarflar paydo buladi, ba'zi yacheykalar bushaydi.

Uz ining sodda tuzilishiga karamay, Tyuring mashinasi anchagina murakkab urin almashtirishlarni bajarishi mumkin.

Jarayon boshida lenta kirish suziga ega bulganda , avtomat kaysidir yacheykaning tugrisida turadi va kaysidir xolatda buladi.

Bu yacheykaning kaysi ekanligini aniklash juda muximdir. Boshlangich yacheykaning tanlanishiga karab, xar xil natijalarga ega bulish mumkin.

Boshlangich xolatga kelsak, doimo kuday bulishi uchun q1 tanlanadi.

Tyuring mashinasi ishi davomida dasturning kataklari buylab «sakrab»yuradi. Bu jarayon avtomat uz xolatini uzgartirmaslik, joyida kolish, yacheykadagi axborotni uzgartirmaslik xakidagi buyruk kiritilgan katakka etib borgunga kadar davom etadi. Masalan, bu katak q4 satr va S7 ustunlar kesishmasida joylashsin va unda S7,N,q4 axborot joylashgan bulsin. Bu katakka etib Tyuring mashinasi tuxtaydi.

Kirish suzi, deb, dastur bajarilishi boshlangunga kadar lentada bulgan suzga aytiladi. Tyuring mashinasi tuxtaganda lentada xosil bulgan suz chikish suzi deb ataladi. Dasturda bunday kataklar bulmasligi xam mumkin. Bunday kataklar bulsa xam , avtomat xech kachon ulargacha etib kela olmasligi mumkin. CHunki avtomatning utishlari kirish suziga va dasturga boglik buladi. Agar Tyuring mashinasi xech kachon tuxtamasa, u berilgan kirish suziga «kullanilmas» deb ataladi.

Tyuring mashinasi berilgan kirish suziga «kullaniluvchan» deyiladi, kachonki, u shu suz ustida ish boshlab, ertami-kechmi tuxtash katagiga etib kelsa.

Lentada yozilgan songa 1 sonini kushib beradigan Tyuring mashinasini kurib utaylik. Kirish suzi lentada joylashgan sondan iborat buladi. U ketma-ket joylashgan yacheykalarda yozilgan buladi. Boshlangich momentda avtomat eng ungda joylashgan yacheyka tugrisida turadi. Mashina oxirgi rakamga 1 ni kushadi, agar bu rakam 9 bulsa, uni 0 bilan almashtiradi. Sungra undan oldin turgan rakam bilan shu amal bajariladi.

Kuyidagi dastur berilgan bulsin:

Mashina Xolatlari	\wedge	0	1	...	8	9
Q1	1,N1,q2	1,N1,q2	2,N1,q2		9,N1,q2	0,L,q1
Q2	\wedge ,N,q2	0,N1,q2	1,N1,q2,		8,N1,q2	9,N1,q2

Bu erda Q1 - rakamlar uzgarishi xolati, q2 esa tuxtash xolati. Sxemaning 2- satri tuxtash kataklari bilan tuldirilgan. Agar q1 xolatda avtomat 0 rakamini ukisa, yoki 1 ni ukisa, va x.k.z. 8 ni ukisa, u xolda uni 1 ga, 2 ga, va x.k.z. 9 ga almashtiradi va q2 xolatga utadi, ya'ni mashina ishi tuxtatiladi. Agar avtomat 9 ni ukisa, uni 0 ga almashtiradiva keyingi rakamga siljiydi, bunda u q1 xolatda koladi. Bu jarayon 9 dan kichik sonlar uchragunga kadar davom etadi. Agar barcha rakamlar 9 lardan iborat bulsa, avtomat ularning barchasini 0 ga aylantiradi, keyin u yana chapga siljib, bush yacheykani uchratadi, u erga 1 ni kiritadi va q2 xolatni kabul kiladi, ya'ni tuxtaydi. Masalan, Tyuring mashinasi 999 ni 1000 ga almashtiradi.

Tyuring mashinalari uchun dasturlarni kiskarok va kulay yozish uchun ba'zi belgilashlar kiritilgan:

- 1) Tuxtash xolatiga utish uchun kursatilgan g' belgisi ifoda etiladi;
- 2) Dasturda R xarfi tushirib koldiriladi;
- 3) Agar yacheykadagi xarf saklanib kolishi kerak bulsa, u katakka yozilmaydt;

Ushbu belgilarni xisobga olib, yukoridagi dasturni kuyidagicha yozish mumkin:

	\wedge	0	1	...	8	9
--	----------	---	---	-----	---	---

Q1	1,!	1,!	2,!		9,!	0,L,q1
----	-----	-----	-----	--	-----	--------

Endi murakkabrok tuzilgan mashinani kurib utaylik. U Tyuring mashinasi lentada joylashgan shtrixlarni sanash ishini bajarsin. Bu shtrixlar mashina uchun «kirish suzi» dan iborat bulsin. Mashina lentadagi barcha shtrixlarni uchirib, lentada shtrixlar sonini unli sanok sistemasidagi ifodasini yozsin. Bu sonni lentada shtrixlardan chapda xosil kilish kerak. Boshlangich momentda Tyuring mashinasi ixtiyoriy shtrixni ukisin va q1 xolatda bulsin.

Kurilayotgan masala uchun algoritm sxemasi kuyidagicha bulishi mumkin:

- 1) Lentadagi suzning birinchi chekasi topilsin;
- 2) Agar suz shtrix bilan tugasa, bu shtrix uchirilsin, aks xolda mashina tuxtatilsin;
- 3) Songa 1 kushilsin va 1) ga utilsin;

Xar safar eng ungda joylashgan shtrix uchiriladi va songa 1 kushiladi. Ushbu 3 ta punktning bajarilishi oxirgi shtrix uchirilgunga kadar davom etadi va 2) punktga asosan, mashina tuxtaydi.

Xar bir punkt Tyuring mashinasining 1 ta xolati bilan bajariladi. SHunday kilib, bizga mashinaning 3 xolati kerak buladi:

- Q1 xolatda mashina suzning ung chetini kidiradi;
- Q2 xolatda shtrixlarni uchiradi;
- Q3 xolatda songa 1 ni kushadi;

Kuyida ushbu Tyuring mashinasi uchun dastur keltiramiz:

	^	0	1	2	...	8	9	/
Q1	L,q2	P,q1	P,q1	P,q1		P,q1	P,q1	
Q2	!	!	!	!		!	!	!
Q3	1,P,q1	1,P,q1	2,P,q1	3,P,q1		9,P,q1	0,L,q3	/,P,q3

Mashina lentadagi rakamlarni ukiydi; q1 xolatida suzning ung chetiga etish belgisi, bu bush katakdir. Bunda avtomat lenta buylab chapga siljiydi va q2 xolatiga utadi. Bunda shtrixni «kurib», avtomat uni uchiradi, yana uchiradi, yana chapga siljiydi va q3 xolatiga utadi. Bunda u srnga 1 ni kushadi . Agar q2 xolatda rakamga duch kelsa, mashina tuxtaydi, chunki bu barcha shtrixlar uchirilgandan dalolat beradiyu q3 xolatda avtomat lenta buylab to songa etgunga kadar charga siljiydi va unga 1 ni kushadi.

Masalan, kirish suzi 3 ta shtrixdan iborat bulsin va avtomat utradagi shtrixning ruparasida tursin:

^ / / / ^
Q1

Ishni boshlab, avtomat 2 marta q1 xolatida ungga siljiydi, bunda kuyidagicha xolat paydo buladi:

^ / / / ^
Q1

Bu momentda avtomat chapga siljiydi va q2 xolatga utadi. Bunda ukilgan shtrixlar uchiriladi, keyin chapga siljib, q3 xolatga utiladi:

$$\begin{array}{c} \wedge \quad / \quad / \quad \wedge \quad \wedge \\ Q3 \end{array}$$

Sungra u yana chapga siljib, Q3 xolatda koladi, bu jarayon avtomat bush yacheykaga duch kelgunga kadar davom etadi. Bu yacheykaga 1 ni yozadi, sungra ungga siljib, q1 xolatiga utadi:

$$\begin{array}{c} 1 \quad / \quad / \quad \wedge \quad \wedge \\ Q1 \end{array}$$

Keyin avtomat 1- bush yacheykaga ungga siljiydi, chapga siljib q2 xolatga utadi. YAna bir shtrix uchiriladi va chapga siljishni bajarib, q3 xolatga utiladi:

$$\begin{array}{c} 1 \quad / \quad / \quad \wedge \quad \wedge \\ Q2 \end{array} \qquad \begin{array}{c} 1 \quad / \quad \wedge \quad \wedge \quad \wedge \\ Q3 \end{array}$$

YAna bir yacheykaga chapga siljib, avtomat 1 ning urniga 2 ni yozadi, sungra ungga siljib, q1 xolatga utiladi:

$$\begin{array}{c} 2 \quad / \quad \wedge \quad \wedge \quad \wedge \\ Q1 \end{array}$$

Jarayon shu tarzda davom etib, natijaga erishiladi:

$$\begin{array}{c} 3 \quad \wedge \quad \wedge \quad \wedge \\ Q1 \end{array}$$

Oxirgi kadamda avtomat q2 xolatga utadi va Tyuring mashinasi tuxtaydi.

Tyuring mashinasi imkoniyatlari. Algoritmlar nazariyasi asosiy gepotezasi. Tyuring mashinasi imkoniyatlarining rang-barangligi shunda kurinadiki, agar A va V algoritmlar Tyuring mashinasi tomonidan realizasiya kilinsa, A va V algoritmlar kompozisiyalarini amalda ijro etuvchi Tyuring mashinasi uchun dasturlar tuzish mumkindir. Masalan, «A bajarilsin, keyin V bajarilsin» yoki «A bajarilsin. Agar natijada «Xa» suzi paydo bulsa, V bajarilsin. Aks xolda V bajarilsin, «yoki A,V navbat bilan bajarilsin, toki V natijani berguncha».

Intuitiv ma'noda bunday kompozisiyalar algoritmlardir. SHuning uchunularning realizasiyasi Tyuring konstruksiyasining universalligini asoslashning yullaridan biri bulib xisoblanadi.

Bunday algoritmlarning bajarilishi konkret algoritmlar A va V larga boglik bulmagan umumiy xolda isbotlanadi. Isbot shundan iborat buladiki, bunda A va V dasturlardan kerakli kompozisiya dasturini kurish usuli kursatiladi. Masalan, A va V algoritmlarning ketma-ket bajarilishiga ekvivalent bulgan AV mashinasini kurish kerak bulsin.

A mashina q1,q2,...,qm ta xolatga ega bulsin. V mashina q1,q2,...,qk k ta xolatga ega bulsin. V mashinaning xolat nomlarini uzgartirib chikamiz: q1 ni qm+1 ga, q2 ni qm+2 ga va x.k.z. qk ni qk+m ga uzgartiramiz. Bu algoritm dasturidagi barcha xolatlar uzgartiriladi. A dasturda xamma joyda g' belgisini qm+1 xolatga utish bilan almashtiramiz. Olingan A dasturni yozib, undan keyin esa kayta nomlangan xolatlari bilan V dastur keltiriladi. Bu ikki dastur birgalikda istalgan AV dasturni xosil kiladi. A algoritm bajarilganda AV dasturning birinchi kismi bajariladi. A algoritm tugagandan keyin tuxtash urniga V kism ishlay boshlaydi.

Masalan, agar A lentadagi shtrixlarni sanash algoritmi, V esa lentadagi songa 1 ni kushishi algoritmi bulsa, oldni kurib utilgan Tyuring mashinalari dasturlaridan foydalanishimiz mumkin.

Bunda $m=3$; $k=1$; A dasturdan AV dasturning 1-3-satrini xosil kilamiz:

oxirgi 4-satr V dasturdan olinadi. Olingan AV dastur oldin lentadagi shtrixlar sonini sanaydigan, ularni uchrib, shu songa 1 ni kushadigan buladi.

Turli algoritmlarni tasvirlab, turli algoritm kompoztsiyalarining Tyuring mashinadari tamonidan bajariluvchi ekanligini kursatish mumkin.

Bu bilan Tyuring uzi taklif etgan konstruksiyaning rang-barang imkoniyatlarga ega ekanligini kursatib beradi. Bu esa unga kuyidagi tezis bilan chikish imkonini beradi:

Ixtiyoriy algoritm mos Tyuring mashinasi tomonidan bajariladi.

Bu Tyuring taklif etgan algoritmlar nazariyasining asosiy gipotezasidir. SHu bilan birgalikda bu tezis – algoritmning formal ta'riflaridan biridir.

Endi algoritmlarning mavjud yoki mavjud emasligini mos Tyuring mashinasini ta'riflash yoki uning kurilishi bilan isbotlash mumkin buldi.

Agar echimni izlash jarayoni biror tusikka duch kelsa, ushbu tusikdan echimni mavjud emasligini isbotlash uchun foydalanishga urinamiz (Algoritmlar nazariyasi asosiy gipotezasiga asoslanib).

Tyuring tezisini isbotlash mumkin emas, chunki undagi «ixtiyoriy algoritm» degan tushuncha aniklanmagan. Uni fakat Tyuring mashinalari misolida asoslash mumkin. Tesisni yana shu bilan asoslash mumkinki, keyinchalik algoritm tushunchasini ta'riflovchi bir necha sxemalar taklif etildi, ular boshkacha kurinishga ega bulsa xam, Tyuring mashinasiga ekvivalentligi isbotlandi.

Shu paytgacha biz konkret algoritmlarni bajarishag muljallangan maxsus Tyuring mashinalari bilan tanishib chikdik. Amma, biz kurib chikkan Tyuring mashinasi ishining interpretatsiyasi umumiy usuli xam algoritmdir. Bundan kelib chikadiki, unga xam kandaydir Tyuring mashinasi tanlanishi mumkin. Bunday mashina universal deb ataladi, chunki u ixtiyoriy Tyuring mashinasi uchun muljallangan ishlarni bajara oladi.

Nazorat uchun savollar:

1. Tyuring mashinasi imkoniyatlari kanday?
2. Mashina avtomati kanday xarakatlanadi?
3. Avtomat nima ishlarni bajara oladi?
4. Universal Tyuring mashinasi nima?
5. Tyuring mashinasi sxemasini taklif etishdan maksad
6. Tyuring mashinasi nima uchun abstrakt mashina deb ataladi?
7. Tyuring mashinasi lentasi tushunchasi?
8. Tyuring mashinasi avtomati nima vazifani bajaradi?
9. Tyuring mashinasi dasturi kanda tuziladi?
10. Algoritmlar nazariyasi asosiy gipotezasida nima deyilgan?

Foydalanilgan adabiyotlar:

1. О.П.Кузнецов. Дискретнауа математика длуа инженера. М:Энергоатомиздат, 1982,155-178 с
2. Е.З. Любимский, В.В. Мартынюк, Н.П.Трифонов Программирование, М:Наука, 1980,17-25 с.
3. В.И.Игошин. Математическауа логика и теориуа алгоритмов. Издательство Саратовского Университета,1991.219-225с.
4. Ю.Л.Ершов,Е.А.Палютин. Математическауа логика, М:Наука,1987г.241-251 с.
5. <http://intsys.msu.ru/stuff/vnosov/theorald.htm#top>
6. www.de.uspu.ru/Informatics/metodes/DPP/F/08/1/Index.htm

5-Mavzu. Hisoblanuvchi funksiyalar va Tyuring tezi.
Tyuring mashinalari va EHMLar (2 soat)

Reja:

- 1. Hisoblanuvchi funksiyalar va sanoqli to'plamlar**
- 2. Hisoblanuvchi funksiyalarga oid Tyuring tezi**
- 3. Tyuring mashinalari va EHMLar**

Kalit suzlar: Hisoblanuvchi funksiyalar, Tezis, EHM.

$\{0,1,2,\dots,n,\dots\}$ Natural sonlar to'plamida berilgan bir yoki bir necha argumentli f funksiyalarni ko'rib o'taylik Hamda ushbu funksiyalar shu to'plamda qiymatlar qabul qiladi deb hisoblaylik. f funksiyaning aniqlanish sohasi $D_f N^n = N \times N \times \dots \times N$ to'plamning qism to'plami bo'lsin.

$$D_f = \{(x_1, x_2, \dots, x_p) : f(x_1, x_2, \dots, x_n) \text{ aniqlangan}\}$$

f ning qiymatlar sohasi N to'plamning qism to'plami bo'ladi:

$$I_f = \{y \in N : \exists x_1, x_2, \dots, x_n (f(x_1, x_2, \dots, x_n) = y)\}$$

1-Ta'rif. $f(x_1, x_2, \dots, x_n)$ Funksiyaning qiymatlarini o'zi aniqlangan argumentlar naborlari uchun hisoblovchi algoritm mavjud bo'lsa, hamda ushbu argumentlar nabori uchun funksiya aniqlanmagan bo'lgan holda ushbu algoritm cheksiz bajarilsa, funksiya hisoblanuvchi deyiladi.

$$M \subseteq N^n = N \times N \times \dots \times N \text{ bo'lsin.}$$

2-Ta'rif. M to'plam echimli deyiladi, agar ixtiyoriy elementni unga tegishli yoki tegishli emasligini aniqlab beruvchi algoritm mavjud bo'lsa.

M to'plamning xarakteristik funksiyasi degan tushunchani kiritaylik. X_M funksiya to'plamning xarakteristik funksiyasi deb ataladi, qachonki X_M M to'plamda berilgan bo'lsa, hamda 2 elementli $\{0,1\}$ to'plamda qiymatlar qabul qilib, quyidagicha aniqlansa:

$$X_M(x_1, x_2, \dots, x_n) = \begin{cases} 0, & \text{agar } (x_1, x_2, \dots, x_n) \notin M \\ 1, & \text{agar } (x_1, x_2, \dots, x_n) \in M \end{cases}$$

Bundan kelib chiqadiki, M to'plam echimli bo'ladi, qachonki uning xarakteristik funksiyasi X_M hisoblanuvchi bo'lsa.

$$M \subseteq N \text{ bo'lsin}$$

3-Ta'rif

$M \subseteq N$ to'plam algoritmik (Rekursiv) sanoqli deyiladi, qachonki, M bo'sh to'plam bo'lsa, yoki qandaydir hisoblanuvchi funksiyaning aniqlanish sohasi mavjud bo'lib, shunday algoritm topiladiki, M to'plamning barcha elementlarini hosil qilsa.

Misol. $M_q \{1,4,9,16,25,36,\dots\}$ to'plam berilgan bo'lsin. Bu to'plam natural sonlarning kvadratlari to'plamidir. Uning elementlarini hosil qilish uchun $1,2,\dots$ sonlarni kvadratga ko'tarish kerak. Boshqacha aytganda M to'plam $f(x) = x^2 : M \rightarrow M$ funksiyaning qiymatlar sohasidir. Bundan tashqari bu to'plam echimlidir ham. Buni isbotlash mumkin. Ta'rifdan kelib chiqqan xolda, ixtiyoriy elementni to'plamga tegishli ekanligini tekshirish

uchun uni tub ko'paytuvchilarga ajratib, biror sonning aniq kvadrati ekanligini, yoki aksinchaligini tekshirib ko'rish mumkin.

1-Teorema. Agar M va L to'plamlar sanoqli bo'lsa, $M \cap L$ va MYL to'plamlar ham sanoqlidir. Isbot; Agar M va L to'plamlar elementlarini hosil qiluvchi algoritmlar mavjud bo'lsa, MYL to'plam elementlari berilgan algoritmlarning bir vaqtda qo'llanilishi orqali hosil qilinadi.

Q_M algoritm M to'plam elementlarini hosil qilsin;

Q_L algoritm esa L to'plam elementlarini hosil qilsin;

U holda $M \cap L$ tuplam elementlarni hosil qiluvchi algoritmning mohiyati quyidagicha bo'ladi: Q_M, Q_L algoritmlar yordamida navbat bilan $m_1, l_1, m_2, l_2, \dots$ elementlar hosil qilinadi. Har bir hosil qilingan m_i element oldin hosil qilingan l_i elementlar bilan taqqoslanadi. Agar m_i birorta l_i bilan mos tushib qolsa, u $M \cap L$ to'plamga kiritiladi. Aks holda l_i element hosil qilinib, uni oldin hosil qilingan m_i lar bilan taqqoslash kerak bo'ladi va h.z. Ushbu jarayon $M \cap L$ to'plamning barcha elementlarni sanab o'tish imkonini beradi. Bu esa teoremaning isbotidan iborat.

2-Teorema. $M \subseteq N$ bo'lsin. M to'plam echimli buladi faqat va faqat uning o'zi va uning kengaytmasi sanoqli bo'lsa.

Isbot. Zaruriyligi. M echimli to'plam bo'lsin. M to'plam bo'sh bo'lmasin, ya'ni shunday a element mavjud bo'lsinki, $a \in M$ bo'lsin. U holda uning xarakteristik funksiyasi X_m hisoblanuvchi, ya'ni uni hisoblovchi algoritm mavjud.

M to'plamni hosil qiluvchi algoritm ko'raylik:

$$x, \text{ agar } X_m(x)=1$$

$$F(x)=$$

$$a, \text{ agar } X_m(x)=0$$

Bundan $M_q\{f(0), f(1), \dots\}$, ya'ni M to'plam f funksiyaning qiymatlar to'plami ekanligi kelib chiqadi. f funksiyaning X_m ning hisoblanuvchi ekanligidan hisoblanuvchi ekanligi kelib chiqadi. Bundan ko'rinadiki, M to'plam sanoqlidir.

Xudi shuningdek, $b \in M$ element tanlab,

$$b, \text{ agar } X_m(x) \neq 1$$

Eslatib utishimiz kerakki, algoritmlarning asosiy xususiyatlaridan biri shuki, u kandaydir bir tipdagi masalalarning cheksiz tuplamidan xar bir masalani echish usulidan iborat bulib, bu usul ushbu masala echimini chekli kadamda topish imkonini beradi. Ammo algoritm tushunchasiga boshka nuqtai nazardan xam karash mumkin. SHu cheksiz bir tipdagi masalalar tuplamidan olinadigan xar bir masalani kandaydir alfavitning kaysidir suzi bilan ifodalash (kodlash) mumkin. Uning echimini esa shu alfavitning boshka kandaydir suzi bilan ifodalash mumkin. Natijada tanlangan alfavitning barcha suzlari tuplaminin biror kism tuplamida aniklangan funksiyaga ega bulamiz. Bu funksiyaning kiymatlar soxasi shu alfavit barcha suzlari tuplamidan iborat buladi.

Bundan shu kelib chikadiki, biror masalaning echimini topish deganda, uni bu funksiyaning berilgan masalani kodlovchi suzdagi qiymatini topish tushuniladi.

Berilgan masalalar sinfini echish algoritmgaga ega bulish deganda esa kurilgan funksiya argumentlarining funksiya aniklanish soxasidan olingan ixtiyoriy qiymatlari uchun funksiya qiymatini chekli kadamda xisoblovchi usulga ega bulish demakdir.

SHu tarzda algoritmik muammo bu – kandaydir alfavitda berilgan funksiyaning xisoblash muammosidir degan xulosaga kelish mumkin.

Endi funksiya qiymatini xisoblash deganda nimani tushunmok kerak?-degan savolga javob berishimiz kerak.

Bu savolga shunday javob berishimiz mumkin: funksiya qiymatini mos Turing mashinasi yordamida topish.

Kanday funksiyalarni Turing buyicha xisoblanuvchi deya olamiz? Olimlarning tadqiqotlari, juda kur amaliy ishlar bunday funksiyalar sinfining ulkan ekanligini kursatdi. Qiymatini xisoblovchi algoritmgaga ega bulgan xar bir funksiya Turing buyicha xisoblanuvchi bulib chikdi.

Bu xulosalar Turinga algoritmlar nazariyasi asosiy gipotezasi deb ataluvchi kuyidagi uezisni taklif etish imkonini berdi.:

Turing tezi: Kandaydir alfavitda berilgan funksiya qiymatini xisoblovchi algoritmi fakat va fakat funksiya Turing buyicha xisoblanuvchi bulganda, ya'ni uni mos Turing mashinasi yordamida xisoblash mumkin bulgandagina mavjud buladi.

Bu tezis amalda aksioma, postulat bulib, prinsipial jixatdan matematik usullar bilan isbotlanishi mumkin emas. Uning kanchalik tugriligi fakat amaliy usullar bilan tekshirilishi mumkin.

Ammo Turing tezisining inkori etilishi imkoniyati xam prinsipial mavjud bulib, buning uchun kandaydir algoritmi bilan xisoblanuvchi, ammo xech kanday Turing mashinasi bilan xisoblanmaydigan funksiya kursatilishi kerak.

Turing mashinasi va EXM lar. Turing mashinalarini urganish va ular uchun dasturlar tuzish algoritmik fikrlash fundamentini xosil kiladi. Uning mazmuni shundan iboratki, u yoki bu jarayonni elementar tashkil kiluvchi kadamlarga ajrata olishdir. Turing mashinasida xisoblash jarayonini kislmlarga bulish (analiz) ning eng oxirgi imkoniyatlari xam amalda kullatilgan. Bular: berilgan birlik informatsiya simvollarini «ukish», elementar simvolda kuzatish ob'ektlarini uzgartirish; berilgan axborotlarni uzgartirishdan iborat. Albatta, xisoblash jarayonini bunday mayda bulaklash uni anchagina uzaytirishi tabiiy. Ammo shu bilan birga bunday elementar kadamlarga bulingan jarayonning mantiqiy strukturasi anchagina soddlashadi va nazariy tadqiqotlar uchun kulay shaklni oladi.

SHunday kilib, Turing mashinasi tushunchasi algoritmik jarayon analizining nazariy kuroli bulib, bundan esa ushbu tushunchaning algoritmik fikrlash moxiyati analizining nazariy kuroli sifatida maydonag keldi, deb xisoblash mumkin.

Xozirgi zamon EXM larida algoritmik jarayon Turing mashinalaridagidek unchalik mayda tshkil etuvchilarga bulinmaydi.

Aksincha, EXM yaratuvchilari mashina tomonidan bajariladigan amallarni imkon boricha kattalashtirishga intiladilar (bu yulda ma'lum chegaralarga rioya etiladi). Jumladan, Turing mashinasida «kushish» amali butun bir dasturni tashkil etadi., xozirgi zamon EXMlarida esa bu amal elementar xisoblanadi.

Bundan tashkari Turing mashinasi cheksiz tashkili xotiraga ega (ikki tomondan chegaralanmagan yacheykalarga bulingan lenta). Ammo biror real mavjud bulgan mashinadi cheksiz xotira bulishi mumkin emas. Turing mashinasi xozirgi zamon EXMlarining xotirasining cheksiz kattalashtirilib borilishi potentsial imkoniyatini aks ettiradi.

Va nixoyat, Turing mashinalari bilan xozirgi zamon EXMlari ishining kiyosiy analizini utkazish mumkin.

Kupchilik EXMlarda uch adresli buyruklar sistemasi kabul kilingan, bu xotiraning birdaniga uchta yacheykasi ichidagilari katnashuvchi binar amallarning bajarilishi bilan belgilanadi. Maslan, a yacheykadagi son v yacheykadagi songa kupaytiriladi, natija s yacheykaga junatiladi.

Ikki adresli va bir adresli EXMlar xam mavjud.

Bir adresli EXMlar kuyidagicha ishlaydi: summatorga a yacheykadagi son chakiriladi; summatorda , masalan, v yacheykadagi songa ushbu son kupaytiriladi, natija summatoridan s yacheykaga uzatiladi.

Tyuring mashinasini bir adresli mashina deb, xisoblash mumkin. Bu erda bir adresli buyruklar yanada soddalashtirilgan: mashina ishining xar bir kadamida kurilayotgan yacheykadagi birgina belgi uzgartiriladi, adres esa fakat 1 ga uzgarishi mumkin(chapdan yoki ungdan kushni yacheykaga utish) yoki umuman uzgarmasligi mumkin. Bu jarayonni anchagina chuzadi, ammo uni standartlashtiradi.

Xulosa kilib shuni aytish mumkinki, xozirgi zamon EXMlari Tyuring mashinalarining real fizik modellari bulib, xisoblash jarayonlarining realizasiyasi uchun yaratilgandir. SHu bilan birga Tyuring mashinalari tushunchasi va bu mashinadar nazariyasi xozirgi zamon EXM larining nazariy fundamenti bulib xisoblanadi.

Nazorat savollari:

- 1. Xisoblanuvchi funksiya nima?**
- 2. Tyuring buyicha xisoblanuvchi funksiyalar deganda nimani tushunamiz?**
- 3. Sanoqli to'plam deb nimaga aytiladi?**
- 4. Echimli to'plam deb nimaga aytiladi?**
- 5. Tyuring mashinasi va EXMlar orasida qanday o'xshash va farqli tomonlar bor?**

Foydalanilgan adabiyotlar:

- 1. О.П.Кузнецов. Дискретная математика для инженера. М:Энергоатомиздат, 1982,201-217 с**
- 2.В.И.Игошин. Математическая логика и теория алгоритмов. Издательство Саратовского Университета,1991.226-232с.**
- 3. <http://intsys.msu.ru/stuff/vnosov/theorald.htm#top>**
- 4. www.de.uspu.ru/Informatics/metodes/DPP/F/08/1/Index.htm**

6-Mavzu. Post mashinasi (2 soat)

Reja:

1. Asosiy tushunchalar va amallar.
2. Post mashinasining tuzilishi.
3. 1-Finit jarayon tushunchasi.
4. Muammoning berilish usuli va 1-Formulirovka

Kalit so'zlar: Post mashinasi, Abstrakt mashina, Kod, Kirish, Shiqish, Lenta, Son, Yasneyka, Avtomat, Dastur, Holatlar

1936 yilda «Simvolicheskaya logika» jurnalining sentyabr sonida Emil Postning «Finit kombinator jarayonlar.1-Formulirovka» nomli maqolasi e'lon qilindi. Ushbu fundamental maqola natijalari algoritmlar nazariyasi faniga asos solgan ilmiy ishlardan biri bo'lib hisoblanadi.

Emil Post konkret muammolar to'plamidan tashkil topgan umumiy muammo maslasini ko'rib chiqadi. Bunda umumiy muammoning echimi konkret muammolarning har birini hal etishi mumkinligi g'oyasi ilgari surilgan. Masalan, $3 \cdot X + 9q0$ tenglama konkret muammolardan biri bo'lsa, $a \cdot X + bq0$ tenglama esa umumiy muammo o'rnida keladi.

Post algoritmik formalizmining asosiy tushunchalari sifatida konkret muammo qo'yiladigan va natija olinadigan simvollar sohasi (L tili)ni va simvollar sohasi uchun berilgan amallar (ko'rsatmalar) to'plamini qarash kerak. Postning simvollar sohasi – bu yacheykalarining cheksiz lentasidir:

–	V	–	V	–	–	V	V	–	–
---	---	---	---	---	---	---	---	---	---

har bir yacheyka belgilangan yoki belgilanmagan bo'lishi mumkin. Konkret muammo «tashqi kuch» (Post termini) tomonidan qo'yiladi. Bu chekli sondagi yacheykalarni belgilash orqali ifoda etiladi. Bunda har qanday konfiguratsiya (masalaning qo'yilishi) belgilangan yacheyka bilan boshlanib, belgilangan yacheyka bilan tugallanadi. o'yilgan konkret masalaga qandaydir ko'rsatmalar to'plamini (ketma-ketligini) qo'llash jarayonidan keyin ushbu berilgan masalaning echimiga ega bo'linadi. Ushbu echim yangi belgilangan va belgilanmagan yacheykalardan iborat bo'ladi. Olingan echim «tashqi kuch» tomonidan qayta ishlanadi. Post o'z abstrakt mashinasi uchun elementar ko'rsatmalar to'plamini ishlab chiqdi:

1. Yacheyka bo'sh bo'lsa, uni belgilash;
2. Yacheyka bo'sh bo'lmasa, belgini o'chirish;
3. Chap tomonga bitta yacheykaga siljish;
4. O'ng tomonga bitta yacheykaga siljish;
5. Yacheyka belgilanganmi-yo'qligini tekshirish, tekshirish natijasiga qarab berilgan ikkita ko'rsatmadan birini bajarish;
6. To'xtash.

Bunda 1-2- ko'rsatma noto'g'ri holatlardan himoya qiladi.

Post mashinasi dasturi ko'rsatmalarining nomlangan ketma-ketligidan iborat bo'ladi.

1-Finit jarayon. Post mashinasi dasturi (Post terminlariga ko'ra ko'rsatmalar nabori) barcha muammolar (masalalar) uchun umumiydir. Bu bilan Post o'z abstrakt mashinasiga universallik talabini qo'yadi. So'ngra Post quyidagi tushunchalarni ilgari suradi:

- Ko'rsatmalar to'plami umumiy muammoga qo'llanuvchan bo'ladi, agar 1-2 ko'rsatmada muammoga duch kelinmasa;
- Ko'rsatmalar to'plami tugaydi, agar 6-ko'rsatma bajarilsa;

- Ko'rsatmalar to'plami 1-Finit jarayonni beradi, qachonki, to'plam muammoga qo'llaniluvchan bo'lib, har bir konkret muammo uchun chekli qadamda tugasa;
- 1-Finit jarayon umumiy muammo uchun echim bo'ladi, agar har bir konkret muammo uchun olingan javob to'g'ri bo'lsa (bu tashqi kuch tomonidan aniqlanadi).

Muammoning berilish usuli va 1-Formulirovka. Post bo'yicha muammo tashqi kuch tomonidan lentadagi chekli yacheykalarni belgilash yo'li bilan beriladi. Post mashinasi «birlik sanoqsistemi» da ishlaydi, ya'ni ($0qV, 1qVV, 2qVVV, \dots$), bitta belgilangan yacheyka, qolgan butun sonlar esa qiymatidan bitta ko'p sondagi sondagi belgilangan yacheykalar orqali ifodalanadi. Umumiy muammoni tashkil qiluvchi konkret muammolar to'plami bilan butun musbat sonlar N to'plami orasida o'zaro bir qiymatli moslik o'rnatish mumkin.

Post bo'yicha 1-umumiy muammo berilgan deyiladi, agar shunday 1-Finit jarayon mavjud bo'lsaki, yacheykalarning boshlang'ich konfiguratsiyasi sifatida n ga qo'llaniluvchan bo'lib, n -konkret muammoni belgilangan yacheykalar ko'rinishida bersa.

Agar 1-Umumiy muammo berilgan bo'lib, echimli bo'lsa, muammoning berilishi va echimi bo'yicha ko'rsatmalar guruhlarini birlashtirib muammo nomeri bo'yicha javobga ega bo'lamiz. Ushbu jumla postning 1-Formulirovkasi deyiladi.

Emil Post o'z maqolasini quyidagi jumla bilan tugatgan: «Muallif ushbu formulirovkani Gyodel-CHyorch ma'nosidagi rekursivlikka mantiqiy ekvivalent bo'ladi deb hisoblaydi. Formulirovkaning maqsadi ma'lum mantiqiy kuchgagina emas, balki psixologik ishonchlilikka ham ega bo'lgan tizimni taqdim etishdan iborat».

SHunday qilib, Post gepotezasi lentadagi simvollar alfaviti, ko'rsatmalar to'plamlari, konkret muammolar tasvirlari va interpretatsiyalari ma'nosidagi ixtiyoriy kengroq formulirovkalar 1-Formulirovkaga keladi, degan fikrdan iborat. Bundan kelib chiqadiki, agargipoteza to'g'ri bo'lsa, qandaydir algoritmlar sinfini aniqlovchi ixtiyoriy boshqa formal tizimlar Emil Postning 1-Formulirovkasi bilan ifodalanuvchi algoritmlar sinfiga ekvivalentdir.

Nazorat savollari:

1. Post mashinasining qanday tuzilgan?
2. Post mashinasi qanday ishlarni bajara oladi?
3. Postning 1-Formulirovkasi mohiyati nimada?
4. Post gipotezasining mohiyati nimada?
5. Post formal algoritmik tizimining mohiyati nimada?

Foydalanilgan adabiyotlar:

1. О.П.Кузнецов. Дискретнауа математика длуа инженера. М:Энергоатомиздат, 1982,144-215 с.
2. Е.З. Любимский, В.В. Мартынюк, Н.П.Трифонов Программирование, М:Наука, 1980,13-40 с.
3. В.И.Игошин. Математическауа логика и теориуа алгоритмов. Издательство Саратовского Университета,1991.209-250с.

7-Mavzu: Markovning Normal algoritmlari (2 soat)

Reja:

1. Normal algoritm tushunchasi
2. Normal algoritmning bajarilish koidasi
3. Normal algoritmda Suz va kism Suz tushunchasi
4. Markovning normalizasiya prinsipi
5. Normal xisoblanuvchi funksiyalar

Kalit so'zlar: Normal algoritm, Normal Algoritm takti, So'zlar jufti, so'z, qism so'z, Normalizasiya prinsipi

1954 yilda ruch matematigi A.A. Markov Tyuring mashinasidagi kabi suzlarni kayta ishlovchi algoritmik sxemani taklif etdi. Bu sxema asosini butunlay boshka prinsiplar tashkil etadi. Bu erda lenta tushunchasi mavjud emas va kayta ishlanuvchi suzning turli kiismlariga bevosit murojaat etish kuzda tutiladi.

A.A. Markov Ushbu algoritmik sxemani normal algoritm deb atadi:

Suzlarni katta xarflar Bilan belgilab (kandaydir alfavitda) Normal algoritmni kuyidagicha ifodalash mumkin:

$A_1 \longrightarrow B_1$

$A_2 \longrightarrow B_2$

...

$A_i \longrightarrow B_i$

...

$A_n \mid \longrightarrow B_n$

Shunday kilib, normal algoritm deganda bir-biri Bilan strelka Bilan birlashtirilgan tartiblangan suzlar juftliklarini tushunish mumkin. Ushbu algoritmlar suzlarni kandaydir alfavitda kayta ishlashning koidalarini ifoda etadi. Bunda berilgan ma'lumotlar va izlangan natijalar algoritmlar uchun kaysidir alfavitdagi suzlardan iborat buladi.

Alfavit deb ixtiyoriy bush bulmagan tuplamga aytiladi. Uning elementlari xarflar deb ataladi, bunday xarflarning ixtiyoriy ketma-ketligi berilgan alfavitdagi suzlar deb ataladi. Bitta Suz ikkinchi suzning kism suzi xam bulishi mumkin. Masalan, agar A rus xarflari alfaviti bulsa, u xolda kuyidagi suzlarni kurib chikish mumkin:

R_1 = paragraf ; R_2 = graf; $R_3 = R_a$; R_2 suz R_1 suzning kism suzidir. R_3 esa R_1 va R_2 larning kism suzidir.

Markov algoritmidagi xar bir suzlar jufti kayta ishlanuvchi suzdagi kism suzlarni almashtiruvchi formulani ifodalaydi. Normal algoritmlarning bajarilishi taktlarga (boskichlarga) bulinadi. Xaar bir takt tartib buyicha birinchi formulani kidirish va uni kullashni uz ichiga oladi. Birinchi takt A_1 suzining KIRISH suzining kismi ekanligini tekshiradi. Masalan MAKAR suzida MA kism suzi bor, ammo MK kism suzi yuk. Agar kism Suz mavjud bulsa, u suzlar

juftining ung kismiga , ya'ni V_1 suz Bilan almashtiriladi. SHu tarzda KIRISH suzining kism suzlar Bilan almashtirilishi amalga oshiriladi. Keyingi taktda uzgartirilgan suzda YAna kism suzlar kidiriladi, agar kism Suz topilmasa keiyngi juftga utiladi va x.k.z. Agar formulani kullashda bir nechta bir xil kism Suz topilsa, doimo chapdan birinchisi almashtiriladi. Normal algoritm bajarilish jarayoni ikki xolatdan birida tuxtaydi:

- barcha formulalar bajarilmaydigan bulib chikadi, ya'ni xech bir formulada kayta ishlanuchi suzning kism suzlari mavjud emas;

- ikkinchi xolda tugallovchi formula kulaniladi;

Bu ikki xolatda xam Normal algoritm berilgan KIRISH suziga kulaniluvchi bulib xisoblanadi. Agar Normal algoritmning bajarilish jarayonida tugallanmaydigan formulalar cheksiz marta kulanilsa, algoritm berilgan KIRISH suziga kulanilmas deb ataladi. Kayta kurish formulalarining ung va chap tomonlari bush suzlardan iborat bulishi xam mumkin.

1-MISOL. Kuyidagi jadvalda Markov Normal algoritmlariga misollar keltirilgan.

Kayta ishlanuvchi Suz	Markov algoritmi	Natija
138578926	(85789,00)	130026
Tararam	(ara,^)	Tram
Tram	(ra,ar)	Tarm
Funksiya	(^,A-)	A-Funksiya
Logika	(ika, ^)	Log
Kniga	(^,^)	Kniga
Polyana	(kor,t)	kulanilmas

2-MISOL. {A,V,S} alfavitdan olingan suzlarni 0 va 1 belgilari Bilan kodlashning normal formulasini karaymiz:

a \longrightarrow 101

v \longrightarrow 1001

s \longrightarrow 10001

Ushbu algorimning saav kirish suziga kulanilishini kurib utaylik:

Kirish suzi a xarfini ikki marta kullaydi. Bizning xolatda birinch a xarfi 101 ga aylantiriladi va kuyidagi uzgargan suzga ega bulamiz: s101av;

Keyingi taktda s101101v natijaga ega bulamiz; 3-taktda 1- formula kulanilmas bulib koladi va 2-formulaga utiladiyu bunda natija: s1011011001; Oxirgi boskichda 100011011011001 natija olinadi. Endi bu suzga xech bir formulani kullab bulmaydi, demak algoritm tuxtaydi.

3-MISOL.

a \longrightarrow

v \longrightarrow

s \longrightarrow

algitm KIRISH suzida a, v, s xarflarini uchirib beradi;

4-MISOL.

→ A algoritm bush kism suzlarni A ga almashtiradi va KIRISH suziga chap tomondan cheksiz marta A larni kushib yozadi, shu sababli bu algoritm xech bir KIRISH suziga kulaniluvchi bulmaydi.

Normal algoritmnning barcha KIRISH suzlariga kulaniluvchi bulishining ETARLILIK ALOMATLARI kuyidagilardan iborat:

1. Barcha almashtirish formulalarida chap kismlar bush emas, ung kismlarida esa chap kismlarida mavjud xarflar yuk;
2. Xar bir almashtirish koidasida ung tomon chap tomondan kiskarokdir;

Birinchi alomat cheksiz takrorlashlar xalkasidan saklaydi. Agar ikkinchi alomat bajarilsa, xar bir almashtirish formulasi bajarilgandan keyin, suzning uzunligi kiskaradi. SHuning uchun almashtirishlar soni KIRISH suzi uzunligidan oshib ket maydi. Bundan 2- alomatga buysinuvchi algoritmlar chap kismi bush bulgan formulaga ega bulla olmasligi kelib chikadi.

5-MISOL.

{a,v,s} alfavitdan olingan ixtiyoriy suzning ung tomonidan a xarfini yozuvchi Normal algoritm tashkil kilishga xarakat kilamiz. Tyuring mashinasidan farkli Normal algoritm KIRISH suzining ung chekasiga tugridan- tugri murojaat etolmaydi. Ammo bu murojaatni tashkil etish uchun alfavitga kushimcha maxsus * belgisini kiritamiz. Istalgan Normal algoritm kuyidagi sxema buyicha kuruladi:

1. * xarfi KIRISH suzining chap tarafiga yozilsin;
2. Agar * xarfi suz oxirida bulmasa, uning keyingi xarf bilan joyini almashritib, yana 2-punkt bajarilsin;
3. * xarfni a xarfiga almashtirilsin va algoritm tuxtatilsin.

Normal algoritm KIRISH suzining chap chekkasiga bevosita murojaatga ega, * xarfni suzning chap chekkasiga yozish uchun kuyidagi formulani kullash etarli: → * ;
ikkinchi punktni bajarish uchun uchta formula kerak buladi:

$$\begin{aligned} * a &\longrightarrow a * \\ * v &\longrightarrow v * \\ * s &\longrightarrow s * \end{aligned}$$

* belgini a xarfiga almashtirish uchukuyidagi formula ishlatiladi: * → a

oxirgi belgi algoritm tugaganligini bildiradi. Endi bizga kerakli normal algoritmni xosil kilish uchun shu beshta urinlashtirish formulalarini tartiblashtirish kerak buladi:

$$\begin{aligned} * a &\longrightarrow a * & (1) \\ * v &\longrightarrow v * & (2) \\ * s &\longrightarrow s * & (3) \\ * &\longrightarrow a & (4) \\ &\longrightarrow * & \end{aligned}$$

Agar KIRISH suzi a, v, s xarflaridan iborat bulsa, u xolda 1-4 formulalar bu KIRISH suziga kulanilmasdir. Algoritm 5- formuladan boshlanadi, bu esa suzning chap chekasiga * belgisini yozilishiga olib keladi. Sungra suzdagi xarflarning tartibiga boglik xolda 1-3 formulalar kulaniladi va xar safar * bir pozisiyaga ungga siljiydi. Bu jarayon * belgisi suzning ung chekkasiga etib borguncha davom etadi. Bu 1-3 formulalarning kulanilmas ekanligini kursatadi, ya'ni * belgisining ung tomonida xarf mavjud bulmaydi. U xolda 4- tugallovchi formula kulaniladi, natijada suzning ung chekasida * belgisi a xarfiga almashtiriladi va algoritm tugallanadi.

Masalan, KIRISH suzi aavsa dan iborat bulsin. U xolda algoritm kuyidagi ketma-ketlikda bajariladi:

*aavsa (5)
a*avsa (1)
aa*vsa (1)
aav*sa (2)
aavs*a (3)
aavsa* (1)
aavsaa (4)

6-MISOL.

Berilgan funksiyani xisoblovchi Normal algoritm :

$$F(111\dots 1) = \begin{cases} 1, & \text{agar } n \text{ ga bulinsa} \\ \wedge, & \text{agar } n \text{ ga bulinmasa} \end{cases}$$

Aq{1} alfavitdagi normal algoritm sxemasini kurib chikamiz:

$$\begin{aligned} 111 &\rightarrow \wedge \\ 11 &\rightarrow \cdot \wedge \\ 1 &\rightarrow \cdot \wedge \\ \wedge &\rightarrow \cdot \wedge \end{aligned}$$

Ushbu algoritm kuyidagi prinsipga asoslanadi: 1 lar soni 3 dan kichik bulmaganda, algoritm tartib bilan 3 tadan xarfni uchiradi, 3 dan kichik 0 dan katta bulganda 2 tadan yoki 1 tadan uchiradi. Xarflar soni 0 ga teng bulsa, 1 ga aylantiriladi. Masalan,

$$\begin{aligned} 1111111 &\rightarrow 1111 \rightarrow 1 \rightarrow \wedge; \\ 11111111 &\rightarrow 111111 \rightarrow 111 \rightarrow \wedge \rightarrow 1 \end{aligned}$$

SHunday klib, Ushbu algoritm berilgan funksiyani xisoblaydi.

TA'RIF. F funksiya A alfavitda berilgan bulsa, u Normal xisoblanuvchi deyiladi, kachonki A alfavitning shunday V kengaytmasi va shu V tuplamda algoritm topilsinki, A dan olingan xar bir R suzni F(R) suzga aylantirsa.

7-MISOL.

$F(X)qX+1$ funksiyani xisoblovchi Normal algoritmni kurib utamiz.

Alfavit sifatida arab rakamlaridan iborat A tuplamni olamiz: $Aq\{0,1,2,3,4,5,6,7,8,9\}$. Normal algoritmni uning kengaytmasi bulgan V tuplamda kuramiz: $VqA+\{a,v\}$

Normal algoritm sxemasi quyidagicha buladi:

$0v \rightarrow \cdot 1$	$a0 \rightarrow 0a$	$0a \rightarrow 0v$
$1v \rightarrow \cdot 2$	$a1 \rightarrow 1a$	$1a \rightarrow 1v$
$2v \rightarrow \cdot 3$	$a2 \rightarrow 2a$	$2a \rightarrow 2v$
$3v \rightarrow \cdot 4$	$a3 \rightarrow 3a$	$3a \rightarrow 3v$
$4v \rightarrow \cdot 5$	$a4 \rightarrow 4a$	$4a \rightarrow 4v$
$5v \rightarrow \cdot 6$	$a5 \rightarrow 5a$	$5a \rightarrow 5v$
$6v \rightarrow \cdot 7$	$a6 \rightarrow 6a$	$6a \rightarrow 6v$
$7v \rightarrow \cdot 8$	$a7 \rightarrow 7a$	$7a \rightarrow 7v$
$8v \rightarrow \cdot 9$	$a8 \rightarrow 8a$	$8a \rightarrow 8v$
$9v \rightarrow \cdot 0$	$a9 \rightarrow 9a$	$9a \rightarrow 9v$
$v \rightarrow \cdot 1$		$\wedge \rightarrow a$

Algoritmni bush suzga kullashga xarakat kilamiz. Bunda oxirgi formula kullaniladi, natijada cheksiz jarayon xosil buladi:

$$\wedge \rightarrow a \rightarrow aa \rightarrow aaa \rightarrow aaaa \rightarrow \dots$$

bundan chikdiki, bu algoritm bush suzga kullanilmas ekan.

Agar algoritmni 499 suziga kullasak, quyidagi ketma-ketlik xosil buladi:

$499 \rightarrow a$ (oxirgi formula) $\rightarrow 4a99$ (ikkinchi ustun urtasidagi formula) $\rightarrow 499v$ (oxiridan oldingi formula) $\rightarrow 49v0 \rightarrow 4v00$ (birinchi ustun oxiridan oldingi formula ikki marta kullangan) $\rightarrow 500$ (ikkinchi ustun urtasidagi formula). SHunday kilib, 499 suzi normal algoritm yordamida 500 suziga aylantiriladi.

Kurib utilgan misolda Normal algoritm V alfavitda kurilgan. V alfavit esa A ning kengaytmasidir. Ammo bu algoritm A alfavitdagi suzlarni YAna A alfavitdagi suzlarga almashtiradi. Bunday xolda algoritm A alfavit ustida berilgan deb ataladi.

Normal algoritmlar nazariyasining asoschisi A.A. Markov Markov Normalizasiya prinsipi deb ataluvchi gepotezasini taklif etdi.

Markovning normalizasiya prinsipi:

Biror alfavitda berilgan funksiyaning kiymatini xisoblovchi algoritm fakat va fakat funksiya normal xisoblanuvchi bulsa, mavjuddir.

Bu prinsip Tyuring tezisi kabi matematik usul Bilan isbotlanmaydi. Ushbu gepoteza amaliyotda uz tasdigini topgan.

Nazorat savollari:

1. Normal algoritm sxemasini taklif etishdan maqsad?
2. Normal algoritmning mohiyati nimada?
3. Normal algoritm takti deganda nimani tushunamiz?
4. Normal algoritmda so'z va qism so'z tushunsalari?
5. Markovning normalizasiya prinsipi nimadan iborat?

Foydalanilgan adabiyotlar:

1. Е.З. Любимский, В.В. Мартынюк, Н.П.Трифонов Программирование, М:Наука, 1980,29-34 с.
2. В.И.Игошин. Математическая логика и теория алгоритмов. Издательство Саратовского Университета,1991.234-239с.
3. Ю.Л.Ершов,Е.А.Палютин. Математическая логика, М:Наука,1987г,241-251 с.

8- Mavzu. Rekursiv funksiyalar (2 soat)

Reja:

1. Rekursiv funksiyalar nazariyasi hisoblanuvchi funksiyalar intuitiv tushunchasini matematik aniqlashtirish usuli sifatida.
2. Primitiv rekursiya operatori.
3. Minimizasiya operatori.
4. Chyorch tezisi.

Kalit so'zlar: Rekursiv funksiyalar,Chyorch tezisi, Primitiv rekursiya, Minimizasiya, Superpozisiya

Rekursiv funksiya tushunchasi hisoblanuvchi funksiya intuitiv tushunchasini konkretlashtirishning yana bi usulidir. Rekursiv funksiyalar sinfini qurishda birlamchi, qaysidir ma'noda eng sodda funksiyalar tanlanadi. So'ngra qoidalar sistemasi qabul qilinib, ushbu qoidalar asosida bor funksiyalardan yangi funksiyalardan yangi funksiyalar quriladi. Bunday qoidalar operatorlar deb ataladi. Demak, tanlangan operatorlar yordamida eng sodda funksiyalardan hosil qilinadigan funksiyalar to'plami qidirilgan funksiyalar sinfini tashkil etadi.

qabul qilingan prinsiplar asosida rekursiv funksiyalar sinfini qurishga xarakat qilamiz. Eslatib o'tishimiz kerakki, qurilayotgan funksiyalarning barchasi natural sonlar to'plamida aniqlangan va natural qiymatlarni qabul qiladi.

Eng sodda funksiyalar sifatida quyidagilarni tanlab olamiz:

$$S(x)=x+1;$$

$$Q(x)=0 \text{ (nol-funksiya); } I_m^n=(x_1,x_2,...,x_n)=x_m \text{ } 1 \leq m \leq n \text{ (proektor funksiyalar);}$$

Yangi funksiyalarni quradigan operatorlar sifatida quyidagi uchtasini tanlab olamiz:

- superpozisiya operatori;
- primitiv rekursiya operatori;
- minimizasiya operatori;

Superpozisiya operatori. n o'rinli ψ funksiya m o'rinli ϕ funksiya va n o'rinli $f_1, f_2, ..., f_m$ funksiyalardan superpozisiya operatori yordamida olindi deyiladi, qachonki, barcha $x_1, x_2, ..., x_n$ lar uchun quyidagi tenglik o'rinli bo'lsa:

$$\psi(x_1, x_2, ..., x_n) = \phi(f_1(x_1, x_2, ..., x_n), ..., f_m(x_1, x_2, ..., x_n))$$

Primitiv rekursiya operatori. $(n+1)$ o'rinli φ funksiya n o'rinli f funksiya va $n+z$ o'rinli g funksiya primitiv rekursiya operatori yordamida hosil qilindi deyiladi, qachonki, ixtiyoriy x_1, x_2, \dots, x_n, y lar uchun quyidagi tengliklar bajarilsa:

$$\varphi(x_1, x_2, \dots, x_n, 0) = f(x_1, x_2, \dots, x_n)$$

$$\varphi(f_1(x_1, x_2, \dots, x_n, y+1), \dots, f_n(x_1, x_2, \dots, x_n, y), \varphi(x_1, x_2, \dots, x_n, y))$$

Bu tengliklar primitiv rekursiya sxemasi deb ataladi.

Misol. Eng sodda funksiyalardan primitiv rekursiya yordamida quyidagi kesik ayirma funksiyasini hosil qilishni ko'rib o'tamiz:

$$X, Y = \begin{cases} x-u, & x \geq y \\ 0, & x < y \end{cases}$$

Birinchidan, X, Y funksiya 2-argumentni fiksirlash yuli bilan aniqlanadigan $X, 1$ funksiya quyidagi munosabatlarni qanoatlantiradi:

$$0, 1 = 0 = 0(X)$$

$$(X+1), 1 = X = I_2^1(x, y)$$

ya'ni eng sodda $0(x)$ va $I_2^1(x, y)$ funksiyalar primitiv rekursiya operatori yordamida hosil qilinadi.

Ikkinchidan, kesik ayirma qoidasidan kelib chiqib, quyidagilarni yozish mumkin:

$$X, 0 = x = I_2^1(x, y)$$

$$X, (y+1) = (x, y), 1 = h(x, y, (x, y)), \text{ bunda } x, y \text{ lar ixtiyoriy.}$$

Bu ayniyatlar 2 o'rinli x, y funksiya primitiv rekursiya operatori yordamida $I_2^1(x, u)$ va $h(x, y, z) = z, 1$ funksiyalardan olingan.

2-Misol. $S(x, y) = x+y$ funksiya primitiv rekursiya operatori yordamida birlamchi funksiyalardan olinganligini ko'ramiz; funksiya uchun quyidagilar o'rinli:

$$x+0=x$$

$$x+(y+1)=(x+y)+1$$

bularni quyidagicha yozish mumkin:

$$S(x, 0) = x$$

$$S(x, y+1) = S(x, y) + 1 \quad \text{Yoki} \quad S(x, 0) = I_2^1(x, y), \\ S(x, y+1) = S(x, y)$$

Bu esa (I_1^1, S) funksiyalar asosida qurilgan primitiv rekursiya sxemasidir.

3-Misol. qo'shish amaliga o'xshash ko'paytirish amali uchun ham quyidagilarni yozish mumkin:

$$\begin{aligned} p(x,0) &= x \quad 0=0=0(x) \\ p(x,y+1) &= xy+x=p(x,y)+x=p(x,y)+x+x=p(x,y)=S(x,p(x,y)) \end{aligned}$$

Bundan $p(x,y)=xy$ funksiya $0(x)$ va $S(x,y) = x+y$ funksiyalardan primitiv rekursiya operatori yordamida tashkil etilgani kelib chiqadi.

Minimizasiya operatori

Φ n o'rinli funksiya $(p+1)$ o'rinli f_1, f_2 funksiyalardan minimizasiya operatori yordamida hosil qilinadi deyiladi, qachonki, ixtiyoriy x_1, x_2, \dots, x_n lar uchun va u uchun $\Phi \Phi(x_1, x_2, \dots, x_n) = y$ tenglik faqat va faqat $f_i(x_1, x_2, \dots, x_n, 0), \dots, f_i(x_1, x_2, \dots, x_n, y-1)$ ($i=1, 2$) qiymatlar aniqlangan va juft-juft teng emas bo'lsa:

$$f_1(x_1, x_2, \dots, x_n, 0) \neq f_2(x_1, x_2, \dots, x_n, 0)$$

....

$$f_1(x_1, x_2, \dots, x_n, y-1) \neq f_2(x_1, x_2, \dots, x_n, y-1),$$

$$f_1(x_1, x_2, \dots, x_n, y) = f_2(x_1, x_2, \dots, x_n, y)$$

qisqacha qilib aytganda, $\Phi(x_1, x_2, \dots, x_n)$ kattalik y argumentning oxirgi tenglikni qanoatlantiruvchi eng kichik qiymatiga teng bo'ladi.

4-Misol. Minimizasiya operatori yordamida olinadigan quyidagi funksiyaning ko'rib chiqamiz:

$$d(x,y) = \mu_z [y+z=x] = \mu_z [S(I_2^3(x,y,z), I_3^3(x,y,z)) = I_1^3(x,y,z)]$$

Masalan, $d(7,2)$ ni hisoblaylik. Buning uchun $y=2$ deb olinib, z o'zgaruvchiga navbat bilan $0, 1, 2, \dots$ qiymatlar berilib, $y+z$ yig'indi hisoblanadi. Yig'indi 7 ga etishi bilan z ning qiymati $d(7,2)$ ga o'zlashtiriladi:

$$z=0, 2+0 \neq 7$$

$$z=1, 2+1 \neq 7,$$

$$z=2, 2+2 \neq 7$$

$$z=3, 2+3 \neq 7,$$

$$z=4, 2+4 \neq 7,$$

$$z=5, 2+5 = 7$$

Shunday qilib, $d(7,2)=5$.

Shu qoida bilan $d(3,4)$ hisoblab ko'ramiz: $z=0, 4+0=4 > 3$

$$z=1, 4+1=5 > 3$$

...

Bu jarayon cheksiz davom etadi, demak $d(3,4)$ aniqlanmagan. Shunday qilib, $d(x,y) = x-y$.

Shunga o'xshash minimizasiya operatori yordamida 2 natural son bo'linmasini ifodalovchi funksiyaning olishi mumkin:

$$x/y=q[<x,y)=\mu z[yz=x]=\mu z[p(I_2^3(x,y,z), I_3^3(x,y,z))=I_1^3(x,y,z)].$$

Eng sodda funksiyalardan primitiv rekursiya, superpozitsiya va minimizatsiya operatorlari yordamida hosil qilingan funksiyalar rekursiv funksiyalar deb ataladi. Agar bunday funksiya hamma joyda aniqlangan bo'lsa, umumrekursiv deb ataladi.

Tyuring tezisiga o'xshash tarzda yoki Markovning normalizatsiya prinsipi kabi rekursiv funksiyalar nazariyasida ham mos tabiiy-ilmiy gipoteza ilgari surilgan. Bu gipoteza Chyorch tezisi deb ataladi:

Sonli funksiya algoritmik echimli bo'ladi faqat va faqat rekursiv bo'lsa.

Intuitiv ma'noda hisoblanuvchi deb topilgan barcha funksiyalar rekursiv deb topilgan.

Nazorat uchun savollar:

1. Rekursiv funksiyalar deb qanday funksiyalarga aytiladi?
2. Rekursiya operatorlari deganda nimani tushunamiz?
3. Primitiv rekursiya operatorining mazmuni nimada?
4. Minimizatsiya operatorining mazmuni nimada?
5. Superpozitsiya operatorining mazmuni nimada?
6. Chyorch tezisining mazmuni nimada?

Foydalanilgan adabiyotlar:

7. О.П.Кузнецов. Дискретная математика для инженера. М:Энергоатомиздат, 1982,178-201 с
8. В.И.Игошин. Математическая логика и теория алгоритмов. Издательство Саратовского Университета,1991.239-243с.
9. Ю.Л.Ершов,Е.А.Палютин. Математическая логика, М:Наука,1987г.251-268 с.
10. <http://intsys.msu.ru/stuff/vnosov/theorald.htm#top>
11. www.de.uspu.ru/Informatics/metodes/DPP/F/08/1/Index.htm

9-Mavzu: Algoritmik echimsizlik tushunchasi (2 soat)

Reja:

1. Algoritmik echimsiz masalalar
2. Uz-uzigaullanuvchanlik muammosi
3. Tyuring mashinasining uz-uzigaullanuvchanligi

Kalit so'zlar: Algoritmik echimsizlik, Qo'llanuvchanlik,o'z-o'ziga qo'llanuvchanlik, Kirish so'zi, Chiqish so'zi

Algoritmlar nazariyasida shunday masalalar mavjudki, ushbularni echish algoritmlari mavjud emas. Bunday masalalar algoritmik echimsiz deb ataladi. Odatda YAngi masalalarning algoritmik echimsizligi ularni oldindan ma'lum algoritmik echimsizmasalalarga keltirish yuli bilan isbotlanadi.Shu bilan birga YAngi masalaning echimi mavjud bulganda oldindan echimsiz deb xisoblangan masalani xam echish mumkinligi isbotlanadi. Bunday masalalar katoriga uz-uzigaullanuvchanlik muammosi misol buladi.

Tyuring mashinasi xakida gapirganda ixtiyoriy Tyuring mashinasi sxemasini kodlangan xolda lentaga yozish mumkinligini bilamiz. Xuddi shuningdek ixtiyoriy Normal algoritmnin urinlashtirish formulalarini ajratish uchun biror belgidan foydalanib kodlash mumkin. U xolda Normal algoritmnin uzi suzga aylanadi va ixtiyoriy Normal algoritm uchun KIRISH suzi sifatida kullnilishi mumkin. Xususiy xolda Normal algoritm uz-uziga KIRISH suzi buladi.

Barcha algoritmlar ikki sinfga bulinadi: uz-uziga kullanuvchan va kullanilmas;

Uz-uziga kullanuvchan algoritmlar deb, Uzining ifodasi ustida ishlab, ertami-kechmi tuxtaydigan algoritmlarga aytiladi. Agar algoritm ishi cheksiz takrorlanuvchi bulsa, bunday algoritmlar uz-uziga kullnailmas deyiladi.

Shunday kilib, xakli savol tugiladi: Kanday kilib u yoki bu algoritmnin uz-uziga kullanuvchanligini aniklash mumkin? YA'ni, ixtiyoriy algoritm uchun yukoridagi savolga javob beruvchi umumiy algoritm topilishi kerak.

Ishni xech kaysi Tyuring mashinasi yordamida xisoblab bulmaydigan funksiya kurishdan boshlaymiz.

Hisoblanuvchi bulmagan funksiyaga misol. Buning uchun foydalanish mumkin bulgan barcha Tyuring mashinalarini ifoda etamiz: Tyuring mashinasidagi ichki xolatlarini cheksiz $q_0, q_1, q_2, \dots, q_s, \dots$ lar bilan belgilanadi. Ushbu mashinalar majmui alfavitlari $a_0, a_1, a_2, \dots, a_s, \dots$ lar bilan belgilandi. Ushbu cheksiz ketma-ketliklarning barcha simvollari ni standart $\{a_0, 1, q, U, CH\}$ alfavit suzlari bilan ifodalaymiz. Bunda kuyidagi koidalar kabul kilinadi:

q_0 q suzi bilan kodlansin;

q_1 qq suzi bilan kodlansin;

q_2 qqq suzi bilan kodlansin;

...

q_i $qq \dots q$ (q lar $i+1$ ta) suzi bilan kodlansin;

va k.k.z.

a_1 1 suzi bilan kodlansin;

a_2 11 suzi bilan kodlansin;

...

a_i $11 \dots 1$ (1 lar $i+1$ ta) suzi bilan kodlansin;

va k.k.z.

Standart alfavitda Tyuring mashinasi dasturini kuyidagi koidaga asosan SO'Z kurinishida ifodalash mumkin. Oldin dasturning barcha buyruklari uchiriladi. Buning uchun

$q_1 a_i \rightarrow q_i a_m x$, $x \in \{e, Ch, O'\}$ yozuvlarda « \rightarrow » belgisi tushirib koldiriladi. q_1, a_i, a_1, a_m xarflar standart alfavitning mos xarflariga almashtiriladi. Bundan keyin buyruk-suzlar ketma-ketligi bitta So'z shaklida yoziladi.

Shunday kilib, xar bir Tyuring mashinasini kandaydir chekli standart alfavitdagi chekli So;z bilan ifoda etish mumkin bular ekan.

Chekli alfavitdagi barcha chekli suzlar tuplami sanokli bulgani uchun, Tyuring mashinalari soni xam shunga mos buladi.

Endi barcha Tyuring mashinalarini nomerlaymiz. Buning uchun turli xil Tyuring mashinalari dasturlarini ifoda etuvchi standart alfavitdagi barcha suzlarni fiksirlangan sanokli cheksiz ketma-ketlik kurinishida joylashtiramiz. Bunda kuyidagi koidaga rioya etiladi. Oldin barcha bir xarfli suzlar yozib olinadi: a_0, a_1, \dots, a_k (bu ketma-ketlik chekli buladi). Sungra ikki xarfli suzlar terib olinadi: $a_{k+1}, a_{k+2}, \dots, a_l$ (bunday suzlar ketma-ketligi xam chekli buladi), keyin uch xarfli suzlar keladi va x.k.z. Natijada barcha Tyuring mashinalari dasturlari ketma-ketligiga ega bulamiz:

$$\alpha_0, \alpha_1, \dots, \alpha_l.$$

1 sonini Turing mashinasi nomeri deb kabul kilamiz.

Endi $A=\{1\}$ alfavitda berilgan suzlar tuplamidan kiymat kabul kiluvchi barcha funksiyalar tuplamini kurib chikamiz. Boshka tomondan, barcha mavjud bulishi mumkin bulgan Turing mashinalarini nomerlash yuli Bilan , ushbu mashinalar tuplamini sanokli ekanligini kursatdik. Bundan Turing buyicha xisoblanuvchi funksiyalar tuplamining xam sanokli ekanligi kelib chikadi . Yuqorida ifodalangan funksiyalar tuplami esa sanoklidir. Bundan Turing buyicha xisoblanuvchi funksiyalarning mavjudligi kelib chikadi. Xech bir Turing mashinasida xisoblanmaydigan konkret funksiya kursatsak, funksiyani xisoblovchi algoritm mavjud emasligini isbotlaydi.

$A_q\{1\}$ alfavitdan olingan suzlar uchun berilgan φ funksiyani kurib chikamiz. Ixtiyoriy k uzunlikdagi $A_q\{1\}$ alfavitdan olingan α Suz uchun :

$$\begin{aligned} & B_n 1, \text{ agar } A_q\{1\} \text{ alfavitda } n \text{ nomerli Turing} \\ & \text{ mashinasi } \alpha \text{ suzni } B_n \text{ suzga aylantirsa;} \\ \varphi(\alpha) = & \\ & 1, \text{ aks holda;} \end{aligned}$$

Teorema: $\varphi(\alpha)$ funksiya Turing mashinasi buyicha xisoblanuvchi emas.

Isbot: Aksini tugri deb xisoblaylik. YA'ni T Turing mashinasi mavjud bulib, uning standart alfaviti $\{a_0, 1, q, U, CH\}$ bulsin va Ushbu funksiyani xisoblasin. K- Ushbu Turing mashinasining nomeri bulsin. $\alpha q 1 1 \dots 1$ (1 lar soni k ta) bulganda $\varphi(\alpha) q \varphi(1 1 \dots 1)$ ga teng. Bunda Suz nimaga teng bulishini kuramiz. Faraz kilaylik T mashina $1 1 \dots 1$ suzni B_k Suzga almashtirsin. Bu B_k xam $A_q\{1\}$ dan olingan. Bundan $\varphi(1 1 \dots 1) q B_k$ ekanligi kelib chikadi.

Ikkinchi tomondan, $\varphi(\alpha)$ funksiyaning ifodasidan $\varphi(1 \dots 1) q B_k 1$ ekanligi ma'lum. Bu kelib chikkan ziddiyat $\varphi(\alpha)$ ni xisoblovchi Turing mashinasi mavjud emasligini isbotlaydi.

Algoritmik echimsizlik muammosiga Yana bir misol – uz-uziga kulllanuvchanlikni aniklashdir.

Faraz kilaylik Turing mashinasi lentasida uning uz funksional sxemasi yozilgan bulsin. Agar mashina Ushbu konfigurasiyaga kulllanuvchan bulsa, uni uz-uziga kulllanuvchi deb ataymiz., aks xolda esa kullanilmas buladi.

Teorema: Turing mashinalari uz-uziga kulllanuvchanligini aniklash muammosi algoritmik echimsizdir.

Isbot: Aksini farazkilaylik. Turing tezisiga asoslanib, Bunday algoritmni xal kiluvchi Turing mashinasi mavjud deb xisoblaymiz. T – shunday Turing mashinasi bulsin. Uning lentasiga mos usulda kodlangan u yoki bu Turing mashinasining dasturi kiritiladi. Bunda agar mashina uz-uziga kulllanuvchan bulsa, kiritilgan Suz mashina tomonidan uz-uziga kulllanuvchanlik xakidagi savolga tasdik javobini anglatuvchi S simvolga aylantiri ladi. Mashina uz-uziga kullanilmas bulsa, uning dasturini ifoda etuvchi KIRISH suzi yukoridagi savolga inkor ma'nosini anglatuvchi A simvolga aylantiriladi.

Endi T_1 Turing mashinasini kurib utsak, Ushbu mashina Ushbu mashina uz-uziga kullanilmas kodlarni A ga aylantirsa, uz-uziga kulllanuvchi kodlarga esa T_1 mashina kullanilmas bulsin. Bunday mashina T mashina yordamida kuruladi, agar uning dasturi kuyidagicha uzgartirilsa, ya'ni S simvol paydo bulgandan keyin , mashina tuxtash urniga , uni cheksiz marta takrorlasin. Demak, T_1 mashina xar kanday uz-uziga kullanilmas kodga kulllanuvchan (A simvol xosil kilinadi), amma uz-uziga kulllanuvchan kodlarga kullanilmasdir. Bu esa ziddiyatga olib keladi, chunki bunday mashina uz-uziga kulllanuvchan xam, kudllanilmas xam bulla olmaydi. Ushbu ziddiyat Teoremani isbotlaydi.

Ushbu isbotlangan teorema ba'zi boshka umumiy muammolarning xam echimsizligini isbotlaydi. Masalan, Turing mashinasi uchun kulllanuvchanlikni aniklash muammosi algoritmik echimsizdir. U kuyidagidan iborat: Kandaydir Turing mashinasi dasturi va konfigurasiyasi

berilgan. Ushbu konfigurasiyaga berilgan mashina kullanuvchanmi, yukmi, aniklash kerak. Bu masalani echish algoritmi mavjud bulganda, uning yordamida mashinaning uz dasturi kodiga kullanuvchan ekanligini aniklash mumkin bular edi. Ammo yukoridagi teorema asosan, bunday algoritmi mavjud emas.

Algoritmik echimsizlikka boshqa misollar. Matematikaning eng mashhur algoritmik muammolaridan Gilbertning 10- muammosini keltirish mumkin. Ushbu masalani Gilbert 1901 yilda Parijda bulib utgan Xalkaro matematiklar Kongressida e'lon kildi. Ushbu muammoning mazmuni quyidagidan iborat: ixtiyoriy Diofant tenglamasining butun echimga ega ekanligini aniklovchi algoritmi mavjudmi?

Diofant tenglamasi deganda, $F(x,y,...,z)=0$, bu erda $F(x,y,...,z)$ butun daraja kirsatkichlariga ega bulgan butun koeffitsientli kupsaddir.

Kup yillar davomida ushbu muammo echimsiz bulib keldi. Fakat 1970 yilga kelib, Rossiyalik matematik YU.V. Matiyasevich uning echimsizligini isbotladi.

Xulosa sifatida shuni kayd kilishimiz kerakki, algoritmik echimsizlikning ma'nosiga katta masalalar guruxiga yagona usul bilan echim kidirish nukta-i-nazaridan karash kerak. SHu vaktning uzida Ushbu guruxga kiruvchi individual masala uzining individual echilish uslubiga ega bulishi mumkin. Masalan, Diofant masalalar turkumiga kiruvchi

$$A_n x^n + A_{n-1} x^{n-1} + \dots + A_1 x + A_0 = 0$$

Tenglamaning butun ildizlari ozod hadning buluvchilari ichidan kidiriladi.

Nazorat uchun savollar:

1. Algoritmik echimsizlik nima?
2. Uz-uziga kullanuvchanlik nima?
3. Kanday algoritmik echimsiz muammolarni bilasiz?

Foydalanilgan adabiyotlar:

1. Е.З. Любимский, В.В. Мартынюк, Н.П.Трифонов Программирование, М:Наука, 1980,36-40 с.
2. В.И.Игошин. Математическая логика и теория алгоритмов. Издательство Саратовского Университета,1991.244-250с.
3. <http://intsys.msu.ru/stuff/vnosov/theorald.htm#top>
4. www.de.uspu.ru/Informatics/metodes/DPP/F/08/1/Index.htm

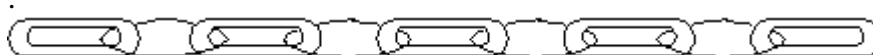
10-Mavzu: Berilganlarning dinamik strukturalari (2 soat)

Reja:

1. Ro'yxat dinamik strukturalari
2. Siklik ro'yxatlar
3. Steklar

Kalit so'zlar: Shiziqli Ro'yxat, Siklik ro'yxat, Stek, Daraxt, Binar Daraxt

Ro'yxat – bu berilganlarning ketma-ket tashkillashtirilgan strukturasidir. Chiziqli ro'yxatlarning massivlardan farqi shundaki, ular dastur bajarilishi jarayonida o'z xajmini o'zgartirish imkoniyatiga ega. Binobarin ro'yxatlarning xajmi oldindan aniqlanmaydi. Chiziqli ro'yxatni zanjir qismlari ko'rinishida tasvirlash mumkin:



Massivda elementlarni ketma-ket joylashtirish bevosita (indekslash orqali) amalga oshiriladi. Ro'yxat elementlari esa maxsus usulda joylashtirilib, uning elementlari axborotlar hamda keyingi qism adresini saqlovchi tugunlarda saqlanadi. Ushbu tugun va adresni quyidagicha e'lon qilish mumkin:

Type

Link = ^Node;

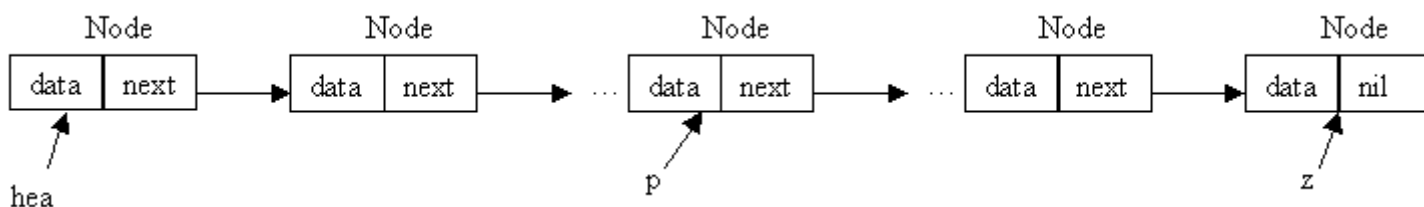
Node = record

Data: integer;

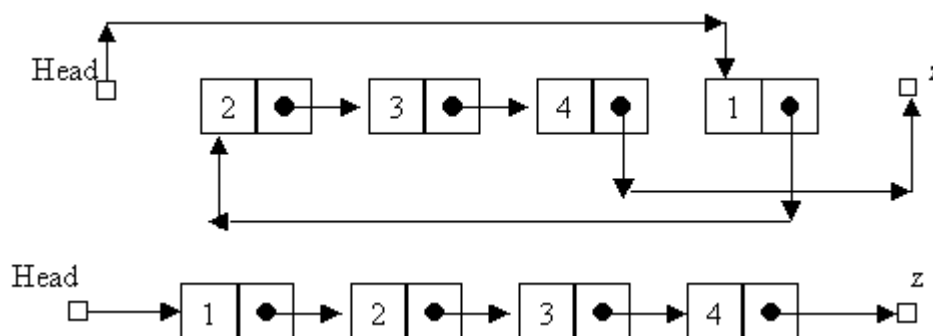
Next: Link;

End;

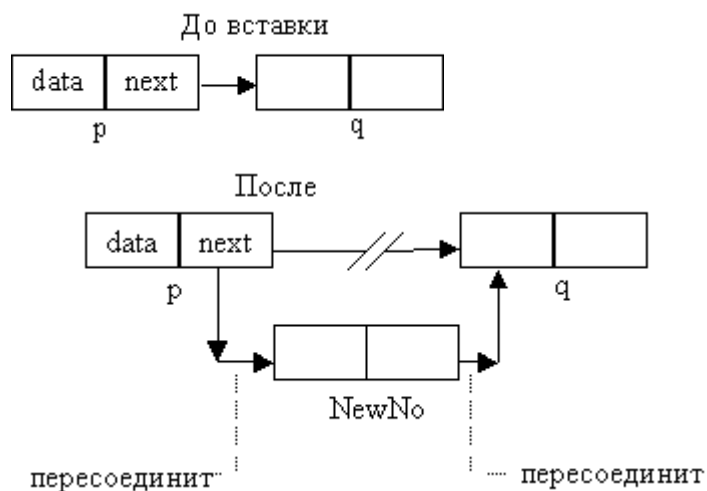
Ro'yxatni e'lon qilish uchun ikkita qo'ishimcha head va z tugunlaridan foydalanamiz. Head ro'yxatning birinchi elementini ko'rsatadi, z esa oxirgi elementini ko'rsatadi. Bunda ro'yxatni quyidagicha ifodalash mumkin bo'ladi:



Berilganlarning bunday strukturalari ma'lumotlar ustida amallar bajarishning massivlardan ko'ra ancha effektivroq usullarni qo'llashga imkon beradi. Masalan, agar 1-elementni ro'yxat boshidan oxiriga o'tqazmoqchi bo'lsak, massivning barcha elementlarini 1- elementga joy bo'shatish uchun 1 pozisiya o'ngga siljitishga to'g'ri keladi. Ro'yxatda esa shu amalni bajarish uchun faqat adreslar o'zgartirilishi kerak holos. Bunda 1-elementni saqlovchi tugun ko'rsatkichini 2-elementni saqlovchi tugunga o'rnatib, head bo'sh tugun ko'rsatkichini esa 1-elementni saqlovchi tugunga o'rnatamiz.

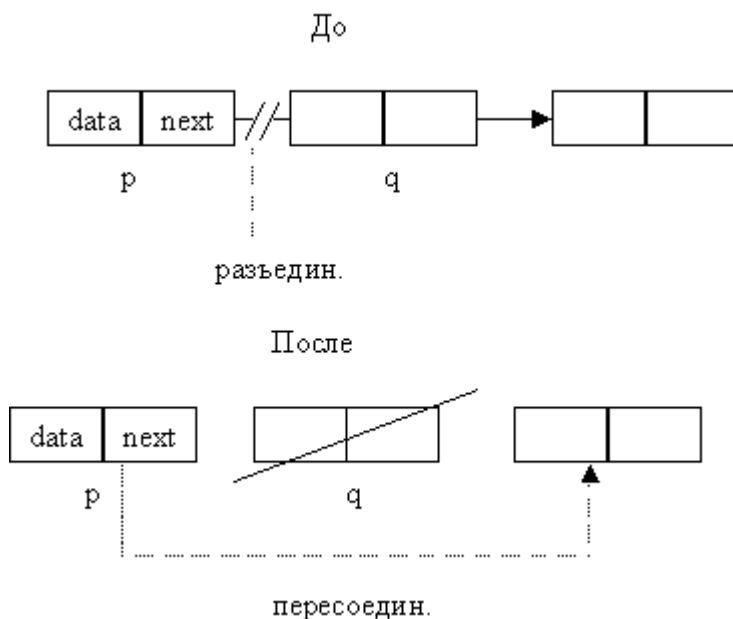


Bundan tashqari berilganlarning ro'yxat strukturasi ketma-ketlikka yangi element qo'shish imkoniyatini yaratadi. Bunda ro'yxat uzunligi bitta elementga uzayadi. Quyidagi rasmda ro'yxatga yangi element qo'shish prosedurasi ifodalangan:



Avvalo ushbu dinamik element yaratiladi, so'ngra yangi elementning ko'rsatkichi q tugunga to'g'irlanadi va oxirida R tugunning ko'rsatkichi yangi tugunga to'g'irlanadi.

Xuddi shuningdek, ro'yxatdan element olib tashlash prosedurasini ham oson bajarish mumkin. Bunda r elementning ko'rsatkichi q dan keyin keluvchi elementga to'g'irlanadi.



Boshqa tomondan qaraganda, shunday amallar borki, berilganlarning ro'yxat strukturasi ularni bajarishda ma'lum noqulayliklarni tug'diradi. Bunday proseduralarga misol sifatida k-elementni topish masalasini keltirish mumkin. Massivda bu prosedura $a[k]$ ga murojaat bilan oson hal etiladi. Ro'yxatda esa k ta adresni ko'rib chiqishga to'g'ri keladi. Xuddi shunday berilgan element oldidagi elementni topish ro'yxat uchun notabiiy amal bo'lib hisoblanadi. Bu muammoni hal etish uchun masalalarning formulirovkasi berilgan elementni olib tashlash, qo'shish o'rniga berilgan elementdan keyingi elementni olib tashlash yoki berilgan elementdan keyin element qo'shish shakliga almashtiriladi.

Paskal tilida chiziqli ro'yxatlarni yaratish va ular ustida amal bajarish vositalari mavjud bo'lib, quyidagi proseduralar yuqorida to'xtalib o'tilgan ro'yxatlar ustida bajariluvchi sodda amallarni realizasiya qiladi:

Type

```
Link = ^Node;  
Node = record  
    Data: integer;  
    Next: Link;  
End;
```

Var

```
    Head, z: link;  
procedure list_initialize;  
begin  
    new( head );  
new( z );  
    head^.next := z;  
z^.next := nil;  
end;
```

```
procedure insert_after( v : integer; t : link );
```

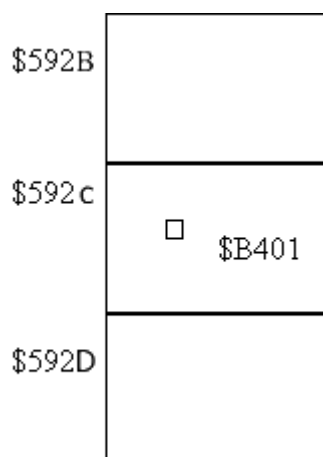
```
var  
x : link;  
begin  
    new(x);  
    x^.data := v;  
x^.next := t^.next;  
    t^.next := x;  
end;
```

```
procedure delete_next( t : link );
```

```
var  
    del: link;  
begin  
del := t^.next;  
t^.next := t^.next^.next;  
dispose(del);  
end;
```

Ushbu proseduralarni batafsilroq ko'rib o'taylik.

Link = ^Node; - bu erda yangi Link tipi yaratilib, u Node toifasidagi ko'rsatkichdan iboratdir. Ko'rsatkich bu- butun toifali o'zgaruvchi bo'lib, berilganlarning qandaydir elementini saqlovchi xotira bayti adresini saqlaydi. Ushbu terminning ma'nosiga alohida to'xtalamiz. Kompyuter xotirasini quyidagicha tasvirlash mumkin: xotira segment deb ataluvchi alohida bloklardan iborat. Dos da har segment nomeri maksimal 16 bitda iborat bo'lishi mumkin.



Ixtiyoriy segmant [0; \$FFFF] oralig'idagi nomerga ega bo'ladi. Bunda \$ belgi 16 lik sanoq sistemasidagi sonni ifodalaydi. \$592B, \$592C, \$592D segmentli xotira sohasini ko'rib chiqaylik. har bir segment ichidagi xotira yacheykalari ham o'z navbatida adreslarga ega. Bu adreslar ham [0; \$FFFF] oralig'idagi nomerlardan iborat. Masalan, \$592C segmentidagi xotira yacheykasining nomeri \$B401 bo'lsa, ushbu xotira yacheykasiga ko'rsatkichning qiymati \$592CB401 dan iborat bo'ladi.

(procedure list_initialize;

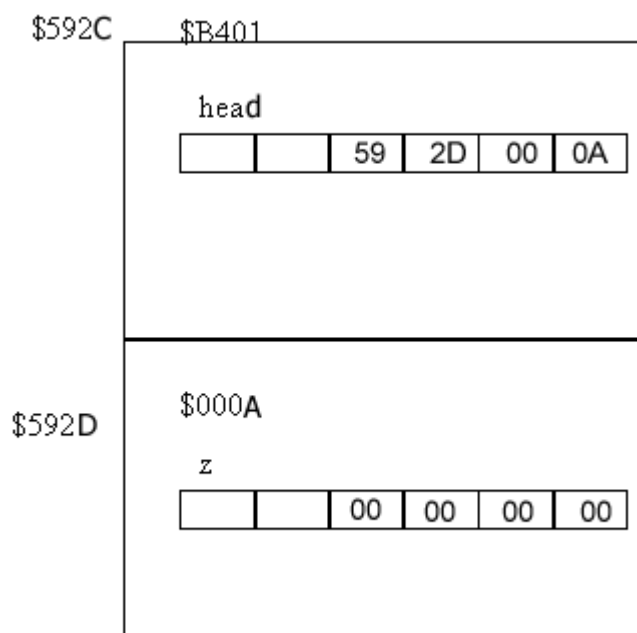
new(head); - new prosedurasi node toifasidagi yangi dinamik o'zgaruvchi yaratadi va head o'zgaruvchisining qiymatini yangi yaratilgan o'zgaruvchini ko'rsatadigan qilib belgilaydi, ya'ni dastur xotirada 6(\$6) bayt uzunlikdagi bo'sh joy qidirib topib, bu sohani band deb e'lon qiladi. So'ngra dastur head o'zgaruvchisiga ushbu rezervlangan joy adresini o'zlashtiradi. Faraz qilaylik, dastur \$592CB401 adresli xotira sohasini topib, bu nomerni head o'zgaruvchisiga o'zlashtirsin.

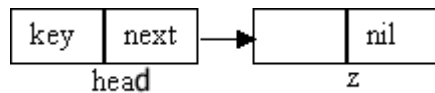
new(z) – dastur xotiradagi \$592D000A adresli sohani topib, uning adresini z ga o'zlashtirsin;

Xotira adresi mazmuniga murojaat qanday amalga oshiriladi, degan savolga quyidagi misol vositasida javob beramiz: Masalan, head^.key:=10; operatori bajarilganda \$592CB401 adresli xotira yacheykasining key deb nomlanuvchi 2 baytli qismiga 10 soni kiritiladi.

head^.next := z; - bu erda esa \$592CB401 yacheykasining Next deb ataluvchi qismiga \$592D000A sonini kiritamiz.

z^.next := nil - z bilan adreslanuvchi xotira yacheykasini nollar bilan to'ldiramiz.





head^.next^.key ifodasi ro'yxatning birinchi elementini, **head^. Next^.next^.key** – esa ikkinchi elementini beradi.

Quyida avtomobillar to'g'risidagi ma'lumotlarni qayta ishlovchi dastur matnini keltiramiz:

```

program car;
uses
crt;
type namestr = string[20];
Link = ^Node;
Node = record
name: namestr;
speed: integer;
next: link;
end;
var head, z: link;
namfind: namestr;
v: 0..4;
endmenu: boolean;
procedure list_initialize;
begin
new(head);
new(z);
head^.next:=z;
z^.next:=nil;
end;
procedure list_destroy;
begin
dispose(head);
dispose(z);
end;
procedure insert_after(name1: namestr; speed1: integer; t: link);
var x : link;
begin
new(x);
x^.name := name1;
x^.speed:= speed1;
x^.next := t^.next;
t^.next := x;
end;
procedure delete_next(t: link);
var
del: link;
begin
del := t^.next;
t^.next := t^.next^.next;
dispose(del);
end;
procedure InpAuto;
var
  
```

```

nam: namestr;
sp: integer;
begin
write('Avtomobil markasini kiriting: ');
readln(nam);
write('Maksimal tezlik: ');
readln(sp);
insert_after(nam, sp, head);
end;
procedure Mylist;
var
Curr: Link;
begin
Curr:=head^.next;
While Curr^.next <> nil do
begin
writeln('Marka: ', Curr^.name, ' Tezlik: ', Curr^.Speed);
curr:=curr^.next;
end;
write('Enterni bosing');
readln;
end;
function findname(fn: namestr): link;
var
Curr: Link;
begin
Curr:=head;
While Curr<>Nil do
if Curr^.name=fn then
begin
findname:=curr;
exit;
end
else
curr:=curr^.next;
findName:=Nil;
end;
begin
list_initialize;
endmenu:=false;
repeat
clrscr;
writeln(' Ishlardan biri tanlansin:');
writeln('1.Ro'yxatga birinch yuzish');
writeln('2. Ro'yxatdagi birinchi elementni o'chirish');
writeln('3. Butun ro'yxatni ko'rish');
writeln('4. Tanlangan elementdan keyingisini o'chirish');
writeln('0. Ishni tugatish');
readln(v);
case v of
1: inpauto;
2: delete_next(head);

```

```

3: mylist;
4: begin
writeln('Ro'yxatdan o'chiriladigan elementdan oldin keluvchi avtomobil markasi kiritilsin');
readln(NamFind);
delete_next(FindName(namfind));
end;
else
endmenu:=true;
end;
until endmenu;
list_destroy;
end.
end;

```

Siklik ro'yxatlar. Ro'yxatlarning ushbu turida oxirgi element ko'rsatkichi birinchi elementga to'g'irlanadi. Ro'yxatlarning bunday turi dasturga ro'yxat elementlarini qayta-qayta ko'rib chiqish imkoniyatini yaratadi. Misol sifatida Djozef masalasini ko'rib chiqamiz. Uning mazmuni quyidagidan iborat: Ommaviy o'zini o'ldirishga qaror qilgan N ta odam aylana shaklida turadi. Bunda har bir M- odam tartib bilan o'zini o'ldirib, aylana torayib borishi kerak. Maqsad odamlarning o'zini o'ldirish tartibini topish. Masalan, agar Nq9 i Mq5 bo'lsa, Odamlar 5, 1, 7, 4, 3, 6, 9, 2, 8 tartibda o'ladi. Ushbu dastur berilgan N va M uchun o'limlar tartibini chiqarib beradi.

Ushbu dasturda siklik ro'yxat strukturasidan foydalanilgan. Oldin 1 do N kalitli ro'yxat yaratiladi. Head o'zgaruvchisi ryxat boshini belgilaydi. So'ngra dastur siklik ro'yxatni ko'rib o'tib, har M-1 ta elementdan keyi keluvchi elementni o'chiradi. Jarayon ro'yxatda bitta element qolgunicha davom etadi.

```

Program Josef;
Type linkq^=ode;
node=record
data:word;
next:link;
end;
Var N,M:word;
head:link;
Procedure Init;
Var q,l:link;i:word;
Begin
Write('Enter N: ');
Readln(N);
New(head);
l:=head;
Head^.data:=1;
Head^.next:=head;
For i:=2 to n do
begin
New(q);
q^.data:=i;
l^.next:=q;
q^.next:=head;
l:=q;
end;
End;
Procedure Del;

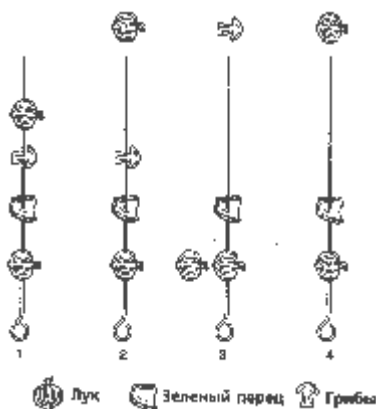
```

```

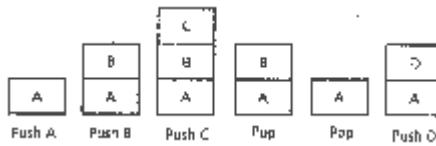
Var i,k:word;
q,p:link;
Begin
Write('Enter M: ');
Readln(M);
k:=0;
i:=1;
q:=head;
While k<n-1 do
begin
While i<m-1 do
begin
inc(i);
q:=q^.next;
end;
i:=0;
inc(k);
p:=q^.next;
q^.next:=q^.next^.next;
writeln(p^.data:4);
dispose(p);
end;
Writeln('The Last: ',q^.data);
End;
Init;
Del;
readln
End.

```

Стек .Стек –yangi element qo’shish va o’chirish jarayoni faqat bir uchidan bajarilishi mumkin bo’lgan dinamik berilganlar strukturasi.Стек ro’yxat boshidan murojaat qilish mumkin bo’lgan elementlar ni saqlash uchun ishlatiladi.Кабоб uchun tayyorlab qo’yilgan go’sht va sabzavotlarni ko’z oldimizga keltiraylik.Сixlar tayyor bo’lgandan so’ng bitta mexmon pomidor emasligini aytsa, uning uchun tayyorlangan sixndagi barcha maslliqlarni olib tashlab, boshqatdan tayyorlashga to’g’ri keladi.



Стек структурасида elementlarni qo’shish va olib tashlash amallar muhim ahamiyatga egadir. Push orerasiyasi стек boshiga element qo’shish, Pop amali esa стек boshidagi elementni olib tashlaydi.



```

type
link = ^node;
node = record
key: integer;
next: link;
end;
Var
head, z: link;
procedure stackinit;
begin
new(head);
new(z);
head^.next:=z;
z^.next:=z;
end;
procedure push(v: integer);
var
t: link;
begin
new(t);
t^.key:=v;
t^.next:=head^.next;
head^.next:=t;
end;
function pop: integer;
var
t: link;
begin
t:=head^.next;
pop:=t^.key;
head^.next:=t^.next;
dispose(t);
end;

```

Nazorat savollari:

1. Chiziqli ro'yxat strukturasi nima?
2. Siklik ro'yxat strukturasi nima?
3. Stek nima?

Foydalanilgan adabiyotlar:

1. Абрамов С.А,Зима Е.В.Начала программирования на языке Паскаль.М.:Наука,1987.87-110с.
2. <http://structur.h1.ru/hash.htm>

11-Mavzu. Optimallashtirish masalalari va ularni echish algoritmlari (2 soat)

Reja:

- 1. Eng yaxshi konserva bankasi haqida masala.**
- 2. Bir o'lchovli optimallashtirish masalalari.**
- 3. Bir o'lchovli masalalarini sonli echish.**
- 4. Ko'p o'lchovli optimallashtirish masalalari.**

Kalit so'zlar: Optimallashtirish masalasi, maqsad funksiyasi, Bir o'lsnovli optimallashtirish masalasi, ko'p o'lsnovli optimallashtirish masalasi, eng kichik qiymat, eng katta qiymat

Oshkor yoki oshkormas ravishda biz optimallashtirish bilan inson faoliyatining istalgan doirasida, shaxsiy ishlardan to eng yuksak umumdavlat ishlarigacha bo'lgan darajada uchrashamiz. Iqtisodiy planlashtirish, boshqarish, chegaralangan resurslarni taqsimlash, ishlab chiqarish jarayonlarini analiz qilish, murakkab ob'ektlarni loyihalash doim muljallangan maqsad nuqtai nazaridan eng yaxshi variantni izlashga qaratilgan bo'lishi lozim. Bu Fan-texnika taraqqiyotining muhim omilidir. Optimallashtirish masalalari o'zaro turli-tuman bo'lganidan ularni echishning umumiy metodlarini faqat matematika berishi mumkin. Ammo matematik apparatdan foydalanish uchun avval bizni qiziqtirgan problemani matematik masala kabi ta'riflash, mumkin bo'lgan variantlarni miqdoriy baholash zarur.

Ko'pgina optimallashtirish masalalari maqsad funksiyasi yoki sifat kriteriyasi deb ataladigan funksiyaning eng kichik (eng katta) qiymatini izlashga keltiriladi. Masalaning quyilishi va tekshirish metodlari maqsad funksiyasining xossalari, hamda echish jarayonida foydalanish mumkin bo'lgan informatsiyaga qat'iy bo'liq bo'ladi.

Matematik nuqtai nazardan maqsad funksiyasi oshkor formula bilan berilgan va differentsiallanuvchi funksiyadan iborat bulgan hol eng soddadir. Bu holda funksiyaning xossalari tekshirish, uning ushish va kamayish oraliqlarini aniqlash, lokal ekstremum nuqtalarini izlashda hosiladan foydalanish mumkin.

Oxirgi davrda fan-texnika taraqqiyoti sharoitida amaliyot tomonidan qo'yilgan optimallashtirish masalalari doirasi keskin kengaydi. Ularning ko'plarida maqsad funksiyasi formula bilan berilmaydi, uning qiymatlari murakkab hisoblar natijasida topilishi, eksperimentdan olinishi mumkin. Bunday masalalar ancha murakkab hisoblanadi, chunki ular maqsad funksiyasini hosil yordamida tekshirib bulmaydi. Shuni yana nazarda tutish lozimki, masalaning murakkabligi uning o'lchamiga, ya'ni maqsad funksiyasining argumentlari soniga jiddiy bog'langan.

Eng yaxshi konserva bankasi haqida masala.

Berilgan V hajmli odatdagi tug'ri doiraviy silindr formasidagi konserva bankasining eng yaxshi varianti ko'rsatilsin. Optimallashtirish maqsadlarining ikki variantini ko'ramiz:

1. Eng yaxshi banka eng kam S sirtga ega bo'lishi kerak (uni tayyorlashga eng kam tunuka sarflanadi);
2. Eng yaxshi banka choklarining uzunligi 1 eng kam bo'lishi kerak (choklarni payvandlashga ketadigan ish miqdori eng kam bo'lsin);

Bu masalani echish uchun banka hajmi, uning sirti va choklarining uzunligi formulalarini yozamiz:

$$V = \pi r^2 h, S = 2\pi r^2 + 2\pi r h, l = 4\pi r + h \quad (1)$$

Banka hajmi berilgan, bu R radius va h balandlik orasida bo'lanishni beradi. Balandlikni radius orqali belgilaymiz: $h = V / \pi r^2$ va topilgan ifodani sirt hamda choklar uzunligi formulalariga qo'yamiz:

$$S(r) = 2\pi r^2 + 2V/r \quad 0 < r < \infty \quad (2)$$

$$l(r) = 4\pi r + V/\pi r^2 \quad 0 < r < \infty \quad (3)$$

Shunday qilib, matematik nuqtai nazardan eng yaxshi banka haqidagi masalaning birinchi holda funksiya eng kichik qiymatiga, ikkinchi holda funksiya eng kichik qiymatiga erishadigan qiymatini topishga keltiriladi. Masalaning birinchi variantini kuramiz. Fuksiyaning hosilasini hisoblaymiz:

$$S'(r) = 4\pi r - 2V/r^2 = 2/r^2(2\pi r^3 - V) \quad (4)$$

Va uni ishoraga tekshiramiz. $0 < r < r_1 = \sqrt[3]{V/(2\pi)}$ bo'lganda hosila manfiy va $S(r)$ funksiya kamayadi, $r_1 < r < \infty$ bo'lganda hosila musbat va $S(r)$ funksiya o'sadi. Demak bu funksiya hosilasi 0 ga aylanadigan $r = r_1$ nuqtada o'zining eng kichik qiymatiga erishadi. Shunday qilib, bankaning $S(r)$ ning minimallik sharti nuqtai nazaridan eng yaxshi radiusi va balandligi ushbu formulalar bilan aniqlanadi:

$$r_1 = \sqrt[3]{\frac{V}{2\pi}}, \quad h_1 = 2r_1 \quad (5)$$

bunda

$$S(r_1) = 3\sqrt[3]{2\pi V^2} \leq S(r) \quad (6)$$

Endi ikkinchi qo'yilgan masalani ko'ramiz. $l(r)$ funksiyani differensiallaymiz:

$$l'(r) = 4\pi - 2V/\pi r^3 = 2/\pi r^3(2\pi^2 r^3 - V) \quad (7)$$

Avvalgidek $0 < r < r_2 = \sqrt[3]{V/(2\pi^2)}$ bo'lganda hosila manfiy va $l(r)$ kamayadi, $r_2 < r < \infty$ bo'lganda hosila musbat va $l(r)$ funksiya o'sadi. Demak bu funksiya hosilasi 0 ga aylanadigan $r = r_2$ nuqtada o'zining eng kichik qiymatiga erishadi. Shunday qilib, bankaning $l(r)$ ning minimalik sharti nuqtai nazaridan eng yaxshi radiusi va balandligi ushbu formulalar bilan aniqlanadi:

$$r_2 = \sqrt[3]{\frac{V}{2\pi^2}}, \quad h_2 = 2\pi r_2^2 \quad (8)$$

bunda

$$l(r_2) = 3\sqrt[3]{4\pi V} \leq l(r) \quad (9)$$

Demak, turli optimallashtirish kriteriyalar uchun turli javoblar kelib chiqadi. Birinchi holda bankaning eng yaxshi balandligi (5) uning diametriga teng, ikkinchi holda (8) uning diametridan π marta ko'p.

Bir ulchovli optimallashtirish masalalari.

Matematik nuqtai nazardan bir o'ldchovli umumiy optimallashtirish masalasining qo'yilishini qo'yidagicha ta'riflash mumkin:

X to'plamda berilgan $f(x)$ maqsad funksiyasining eng kichik (eng katta) qiymati topilsin. $x \in X$ o'zgaruvchining shu funktsiya ekstremal qiymatga erishadigan qiymati aniqlansin.

Matematik analizda kesmada uzluksiz bo'lgan funktsiyaning xossalari o'rganilganda quyidagi teorema isbotlanadi:

Veyershtrass teoremasi. $[a,b]$ kesmada uzluksiz bo'lgan har bir $f(x)$ funktsiya shu kesmada o'zining eng kichik va eng katta qiymatlariga erishadi, ya'ni $[a,b]$ kesmada shunday x_1, x_2 nuqtalar mavjudki, ixtiyoriy $x \in [a,b]$ uchun

$$f(x_1) \leq f(x) \leq f(x_2) \quad (10)$$

tengsizliklar bajariladi.

Xususan, eng kichik va eng katta qiymatga bir vaqtda bir necha nuqtalarda erishish mumkinligi inkor etilmaydi.

Berilgan holda Veyershtrass teoremasi mavjudlik teoremasi rolini o'ynaydi. SHu teoreмага ko'ra kesmada berilgan va uzluksiz $f(x)$ funktsiya uchun optimallashtirish masalasi doim echimga ega.

Quyida biz eng yaxshi konserva bankasi haqidagi masalaga o'xshash eng sodda masalalar sinfini ko'ramiz. Ularni tekshirishda maqsad funksiyasi $f(x)$ $[a,b]$ kesmada differensiallanuvchi va uning hosilasi $f'(x)$ uchun oshkor ifoda topish imkoniyati bor deb faraz qilamiz.

Hosila 0 ga aylanadigan nuqtalar $f(x)$ funktsiyaning kritik nuqtalari deyiladi. Agar hosilani funktsiyaning o'zgarish tezligi deb qarasa, u holda kritik nuqtalarda bu tezlik 0 ga teng.

$f(x)$ funktsiya o'zining eng kichik (eng katta) qiymatiga yo $[a,b]$ kesma chegaraviy nuqtalarida yoki uning biror ichki nuqtasida erishadi.

Qaralayotgan funktsiyalar sinfi uchun optimallashtirish masalasini echishning quyidagi qoidasini ta'riflaymiz:

$[a,b]$ kesmada differensiallanuvchi $f(x)$ funktsiyaning eng kichik va eng katta qiymatlarini topish uchun shu kesmada uning barcha kritik nuqtalarini topish, ularga chegaraviy a va b nuqtalarni qo'yish va barcha shu nuqtalarda funktsiya qiymatlarini taqqoslash kerak. Ulardan eng kichik va eng kattasi $f(x)$ funktsiyaning butun kesma uchun eng kichik va eng katta qiymatlarini beradi. Chegaraviy nuqtalarni topish kerak emas, shuning uchun hamma ish

$$f'(x) = 0 \quad (11)$$

tenglamaning ildizlaridan iborat bo'lgan kritik nuqtalarni topishga keltiriladi.

Optimallashtirish masalasini echishning yuqorida bayon etilgan qoidasini namoyish qilish uchun $[-2,3]$ kesmada

$$f(x) = 3x^4 - 4x^3 - 12x^2 + 2 \quad (12)$$

funktsiyani ko'ramiz. Uning hosilasini topamiz:

$$f'(x) = 12x^3 - 12x^2 - 24x$$

Shunday qilib, (11) tenglama kritik nuqtalarni topish uchun berilgan holda

$$x^3 - x^2 - 2x = 0 \quad (13)$$

ko'rinishga ega bo'ladi. Shu tenglamaning hamma $x_1 = -1$, $x_2 = 0$, $x_3 = 2$ ildizlari berilgan kesmaga tegishli. Ularga chegaraviy $a = -2$, $b = 3$ nuqtalarni qo'shib, (12) funktsiyaning mos qiymatlarini hisoblaymiz:

$$f(-2) = 34, f(-1) = -3, f(0) = 2, f(2) = -30, f(3) = 29$$

Bu sonlarni taqqoslashdan $f(x)$ funksiyaning eng kichik qiymati kritik nuqtalardan biri x_{q2} da, eng katta qiymati x_{q-2} nuqtada erishishi kelib chiqadi, bunda

$$F_{\min} = f(2) = -30, \quad f_{\max} = f(-2) = 34$$

Eng sodda, konserva bankasi haqidagi masalada yoki ko'rilgan (12) misoldagi kabi hollarda hosilaning nollarini analitik ravishda topish mumkin. Ammo bu usulda barcha kritik nuqtalarni topish muhim, aks holda xatolikka yo'l quyish mumkin.

Bir o'lchovli optimallashtirish masalalarini sonli echish.

Misol ko'raylik. Kimyoviy zavod biror moddani ishlab chiqaradi. Bizni qiziqtiradigan mahsulotning miqdori harorat bilan aniqlanadi: $y=f(T)$. Haroratni ma'lum chegaralarda o'zgartirish mumkin: $T_1 \leq T \leq T_2$ funksiyaning ko'rinishi avvaldan ma'lum emas, u foydalaniladigan xom ashyoga bog'liq. Navbatdagi bir partiya xom ashyo olingandan keyin ishlab chiqarishni eng qulay tashkil etish, ya'ni $f(T)$ funksiya o'zining eng katta qiymatiga erishishi uchun zarur bulgan T harorat topilsin.

Berilgan holda $f(T)$ maqsad funksiyasi uchun hech qanday formula yo'q. Biror T temperaturada uning qiymatini hisoblash uchun yo laboratoriyada yoki ishlab chiqarish sharoitlarida tajriba o'tkazit kerak. Ravshanki, o'lchashlar chekli sonda bo'lishi kerak, shu sababli $f(T)$ funksiyaning qiymatlari chekli sondagi nuqtalarda ma'lum bo'ladi. Uning hosilasi qiymatlarini biz mutlaqo bila olmaymiz.

Shunday optimallashtirish masalalari ham bo'lishi mumkinki, ular da $y=f(x)$ maqsad funksiyasi biror matematik masalani sonli echish natijasida topiladi. Bunda biz maqsad funksiyasining oshkor formulasiga ega bo'lmaymiz, ammo uning qiymatini istalgan $x \in [a,b]$ argument uchun topa olamiz.

Shunday qilib, bir o'lchovli optimallashtirish masalasining quyidagicha qo'yilishi bilan bog'liq bo'lgan matematik masalalarni muvaffaqiyatli qilamiz: uzluksiz $f(x)$ funksiyaning $[a,b]$ kesmaning biror chekli sondagi nuqtalaridagi qiymatlarini aniqlab, uning shu kesmadagi eng kichik (eng katta) qiymatini taqribiy toping.

Bu masalani turli yo'llar bilan echish mumkin:

1. Nuqtalarni kesma bo'yicha tekis taqsimlash metodi.

Biror n soni olamiz, $h=(b-a)/n$ qadamni hisoblaymiz va $f(x)$ funksiyaning qiymatlarini $x_k = a+kh$ ($k=0,1,...,n$) nuqtalarda hisoblaymiz:

$y_k=f(x_k)$, so'ngra topilgan sonlar orasidan eng kichigini topamiz:

$$m_n = \min(u_0, u_1, \dots, u_n) > m_n > m = \min_{x \in [a,b]} f(x)$$

m_n soni $f(x)$ funksiyaning $[a,b]$ kesmadagi eng kichik taqribiy qiymati deb qabul qilish mumkin. $f(x)$ funksiyaning uzluksizligiga asosan

$$\lim_{n \rightarrow \infty} m_n = m = \min_{x \in [a,b]} f(x)$$

tenglikka egamiz, ya'ni nuqtalar soni n ortib borishi bilan m_n m ni qabul qilish natijasida yo'l qo'yiladigan xato 0 ga intiladi.

Funksiyaning eng kichik qiymatini aniqlashdagi xatolik $\delta_n = m_n - m$ berilgan ε aniqlikdan ortiq bo'lmasligi uchun, $\delta_n \leq \varepsilon$ bo'lishi uchun qanday n ni olish kerak?

Agar bizga $f(x)$ funksiyaning $[a,b]$ kesmada uzluksizligigina ma'lum bo'lsa, qo'yilgan savolga javob berish mumkin emas. Bu qiyinchilik x_k nuqtalarni tavsiya etilgan tanlash usuliga bo'liq emas. Qanday n olmaylik va $[a,b]$ kesmada n ta nuqtani qanday tanlamaylik, doim shunday uzluksiz funksiyaning ko'rsatish mumkinki, u uchun m_n son m dan ε dan ko'proq va farq qiladi.

Masalani $\varepsilon (\delta_n \leq s)$ aniqlik bilan echish uchun zarur bo'lgan nuqtalar soni n ni qat'iy baholash faqat qaralayotgan funksiyalar sinfini toraytirish hisobiga berilishi mumkin. Nuqtalar soni va aniqlik haqidagi masalani echishda maqsad funksiyasining xossalari, uning masalaning xarakteri va xususiyatlaridan kelib chiqadigan silliqlik darajasi haqidagi barcha qo'shimcha informasiyadan foydalanish muhim.

2. Nuqtalarni kesma bo'yicha maqsad funksiyasini hisoblash natijalarini e'tiborga oluvchi taqsimlash metodi.

YUqorida bayon etilgan metod uchun $[a,b]$ kesma bo'yicha x_k "sinash" nuqtalarini tekis taqsimlash xarakterli. Ularning joylashishi avvaldan qat'iy belgilangan va $y_k=f(x_k)$ sonlarni hisoblash natijasida $f(x)$ funksiya haqida olinadigan informasiyaga bog'liq emas. Bu usul butun kesmani tekshirib chiqish imkonini beradi. x_k nuqtalarni tekis taqsimlaganda kesmaning barcha qismlariga: maqsad funksiyasi katta bo'lganlariga ham, u kamayadiganlariga ham bir xil ahamiyat beramiz. Bu hisobni cho'zadi va tekshirishni qiyinlashtiradi.

Shu sxema bo'yicha hisoblashni tashkil qilishni o'rmonda tajribasiz zamburug' teruvchining o'zini tutishi bilan taqqoslash mumkin. Zamburug'ni izlab u zamburug'li yoki zamburug'siz joylarning farqiga bormasdan butun o'rmon bo'ylab yuradi. Natijada zamburuqsiz joylarni qarab chiqish uning ancha kuchi va vaqti bekorga ketadi. Tajribali zamburug'hi o'zini butunlay boshqacha tutadi. U zamburug'li joylarda ancha turadi va ularni ayniqsa e'tibor bilan qarab chiqadi, zamburug' siz joylardan ortiqcha vaqt sarf qilmasdan tez o'tib ketadi.

Funksiyaning eng kichik qiymatini izlashni "tajribali zamburug'chi" metodi bilan tashkil qilish uchun x_k nuqtalarni avvaldan mo'ljallab tanlash usulidan voz kechish, navbatdagi nuqtani $f(x)$ funksiya haqida uning qiymatini avvalgi nuqtalarda hisoblash natijasida olingan informasiya asosida tanlash lozim. Bunda $[a,b]$ kesmaning hisobashlar funksiyaga kichik qiymatlar beradigan qismlariga alohida e'tibor berish kerak.

$f(x)$ funksiyaning qiymatlari ikki chegaraviy $x_0=a$ va $x_1=b$ nuqtalarda hisoblaymiz: $y_0=f(x_0)$, $y_1=f(x_1)$. Shundan keyin navbatdagi x_2 nuqtani kesmaning funksiya kamroq qiymat qabul qiladigan chegarasiga yaqinroqda tanlaymiz. Uning holatini u_0 va y_1 sonlar orasidagi munosabatga qarab aniqlaymiz: ular orasidagi farq qancha katta bo'lsa, x_2 nuqtaning tegishli tomonga siljishi shuncha kup bo'ladi. x_3 nuqtani x_0 , x_k x_2 nuqtalarning o'zaro joylashishiga va u_0 , y_1 , u_2 sonlar orasidagi munosabatlarga qarab tanlaymiz va h.k.z.

3. Maxsus metodlar.

Optimallashtirish masalasini echish haqida yangi masalalar quyish uchun zamburug'larni terish haqidagi misoldan yana bir marta foydalanamiz. Zamburug'chi o'rmonda unda zamburug' borligidan boshqa hech narsa bilmagan holda birinchi marta kirishi mumkin. Boshqa hol bo'lishi ham mumkin. Odam o'zi bilgan o'rmonga keladi. Birinchi va ikkinchi holda uning o'zini tutishi turlicha bo'ladi. Bunda agar u o'rmonning unga ma'lum xususiyatlaridan foydalana bilsa, savatni ancha tez to'ldiradi.

Hozircha optimallashtirish masalalarini muvaffaqiyatli qilar ekanmiz, biz ularni echishning universal metodlari haqida gapirdik. Ammo ko'pgina hollarda masalaning xarakteridan maqsad funksiyasining xossalari haqida qandaydir qo'shimcha informatsiya kelib chiqadi. Bundan maxsus aloritmlarni ishlab chiqishda foydalanish mumkin. Bunday usul hisoblashlar hajmini kamaytirishga va javobni eng samarador usul bilan topishga imkon beradi. Misol sifatida maqsad funksiyasi $y=f(x)$ $[a,b]$ kesmada faqat bitta minimumga ega ekani bizga avvaldan ma'lum bo'lgan holni ko'ramiz. Bu holda masalani echish uchun quyidagi metoddan foydalanish mumkin. Birinchi qadam olamiz va $f(x)$ funksiyaning x_0 ga, $x_0=a+h$, $x_0=a+2h, \dots$ nuqtalardagi qiymatlarini birin-ketin hisoblaymiz va topilgan u_0, y_1, u_2, \dots sonlarni o'zaro taqqoslaymiz. Avval ular kamayadi: $u_0 > y_1 > u_2 > \dots$, ammo keyin shunday x_k ga x_k+h nuqta topiladiki, undagi funksiya qiymati $y_k=f(x_k)$ uchun $y_{k-1} > u_k$, $u_{k+1} \geq u_k$ tengsizliklar o'rinli bo'ladi. Bundan funksiyaning eng kichik qiymati $[x_{k-1}, x_{k+1}]$ kesmada erishilishi ko'rinadi va uni taqriban $y_k=f(x_k)$ deb olish mumkin bo'ladi. Agar masalani echish aniqligi ta'minlanmagan bo'lsa, u holda h qadamni kamaytirib, bayon etilgan prosedurani $[x_{k-1}, x_{k+1}]$ kesma uchun qaytarish kerak.

Kimyoviy jarayon uchun optimal temperatura haqidagi masala shunga o'xshash masalalarga kiradi. Ko'pgina kimyoviy reaksiyalar uchun temperatura T ning o'sishi bilan funksiya avval o'sadi, keyin esa maksimumdan o'tib, kamaya boshlaydi. Biz shu maksimumni topishimiz lozim bo'ladi. Buning uchun yuqorida bayon etilgan metoddan foylanasa bo'ladi. U $f(T)$ funksiyaning uncha ko'p o'lchashlarini talab etmaydi. Biz minimumni emas, maksimumni izlayotganimiz metod nuqtai nazaridan ahamiyatga ega emas, faqat barcha tengsizliklar o'z belgilarini qarama-qarshisiga o'zgartiradi.

Ko'p o'lchovli optimallashtirish masalalari.

Biz hozirgacha maqsad funksiyasi bitta argumentga bog'liq bo'lgan bir o'lchovli optimallashtirish masallarini muvaffaqiyatli qildik. Ammo optimallashtirish amaliy ahamiyatga ega bo'lgan ko'pchilik masalalari ko'p o'lchovlidir: ularda maqsad funksiyasi bir necha argumentga bog'liq bo'ladi, ba'zida argumentlar soni anchagina katta bo'lishi mumkin.

Masalan, kimyoviy ishlab chiqarish haqidagi masalani eslaylik. Biz qayd qildikki, unda maqsad funksiyasi temperaturaga bog'liq va uni ma'lum yo'l bilan tanlansa, unumdorlik maksimal bo'ladi. Ammo unumdorlik temperatura bilan birga bosimga, ishlatiladigan xom ashyolar, katalizatorlarning to'yinganliklari orasidagi munosabatga va qator boshqa faktorlarga bog'liq. Shunday qilib kimyoviy ishlab chiqarishning eng yaxshi shartlarini tanlash masalasi - bu optimallashtirish tipik ko'p o'lchovli masalasidir.

Bunday masalalarning matematik qo'yilishi ularning bir o'lchovli holda qo'yilishiga o'xshash: argumentning mumkin bo'lgan qiymatlari bo'yicha biror berilgan E to'plamda maqsad funksiyasining eng kichik (eng katta) qiymati topilsin. Maqsad funksiyasi uzluksiz, E tuplam yopiq chegaralangan soha bo'lganda Veyershtass teoremasi o'rinli. Bu bilan echimning mavjudligi avvaldan ma'lum bo'lgan optimallashtirish masalalari sinfi ajratiladi. Bundan keyin barcha qaraladigan masalalar shu sinfga mansub deb faraz qilamiz.

Bir o'lchovli holdagi kabi masalaning xakteri va mos ravishda mumkin bo'lgan echish metodlari maqsad funksiyasi haqida u ni tekshirish jarayonida bizga ma'lum bo'lgan informatsiyaga bog'liq bo'ladi. Bir xil hollarda maqsad funksiyasi analitik formula bilan beriladi va differensiallanuvchi buladi. Bunda uning xususiy hosilalarini hisoblash, har bir nuqtada funksiyaning o'sish va kamayish yo'nalishlarini aniqlaydigan gradient uchun oshkor ifoda topish va bu informatsiyadan masalani echishda foydalanish mumkin. Boshqa xollarda maqsad funksiyasi uchun hech qanday formula yo'q, faqat uning qiymatini

qaralayotgan sohaning istalgan nuqtasida aniqlash (hisoblar yordamida, tajriba natijasida va h.k.z.) mumkin. Bunday masalalarni echish jarayonida maqsad funksiyasining chekli nuqtalardagi qiymatlari topiladi va shu informasiya bo'yicha butun soha uchun uning eng kichik qiymatini taqribiy topish talab etiladi.

Ko'p o'lchovli masalalar, murakkabroq va sermashaqqatdir. Funksiyaning eng kichik qiymatini izlashning bir o'lchovli masalalar uchun yuqorida muhokama qilingan \oyasi bo'yicha eng soda taqribiy metodini olamiz. +aralayotgan sohani h qadamli tur bilan qoplaymiz va uning tugunlarida funksiya qiymatlarini aniqlaymiz. Topilgan sonlarni o'zaro taqqoslab, ular ichida eng kichigini olamiz va uni butun sohada funksiyaning taqribiy eng kichik qiymati deb qabul qilamiz.

Bu metoddan ikki o'lchovli, uch o'lchovli masalalarni echishda ham foydalaniladi. Ammo u katta o'lchovli masalalarni echishda hisoblashlarga juda ko'p vaqt sarflanganligi sababli amalda yaramaydi.

Haqiqatan maqsad funksiyasi 5 ta o'zgaruvchiga bog'liq, uning aniqlanish sohasi besh o'lchovli kubdan iborat bo'lsin. Turni qurishda shu kubning har bir tomonini 40 ta bo'lakka bo'lamiz. Unda tur tugunlarining umumiy soni 41-10 ga teng bo'ladi. Bitta nuqtada funksiya qiymatini hisoblash uchun 1000 ta arifmetik amal bajarilishi talab etilsin. Bunda amallarning umumiy soni 10^{11} ni tashkil etadi. Bu esa sekundiga 1 mln. arifmetik amal bajaradigan EHM uchun 1 sutkadan ko'proq uzluksiz ishlash demakdir.

Olib borilgan baholash ko'rsatadiki, optimallashtirishning katta masalalari uchun uzluksiz saralash metodi yaramaydi.

Nazorat savollari:

- 1. Optimallashtirish masalasi deganda nimani tushunamiz?**
- 2. Bir o'lsnovli optimallashtirish masalasi deganda nimani tushunamiz?**
- 3. Ko'p o'lsnovli optimallashtirish masalasi deganda nimani tushunamiz?**
- 4. Optimallashtirish masalalarini esnuvsni qanday algoritmlarni bilaiz?**
- 5. Nuqtalarni kesma bo'yisna tekis taqsimlash metodining mohiyati nimada?**
- 6. Nuqtalarni maqsad funksiyasini hisoblash natijalariga qarab taqsimlash metodining ahamiyati nimada?**

Foydalanilgan adabiyotlar:

**Amaliy matematikadan kirish lelsiyalari. А.НТихонов,
Д.П.Костомаров. Toshkent. O'qituvchi, 1987. 137-168 b.**

12-Mavzu. Ichki Saralash algoritmlari (2 soat)

Reja:

1. "Pufakchali" saralash algoritmi.
2. "Piramidali" saralash algoritmi.
3. "Tez" saralash algoritmi.

Bizga X fayl berilgan bulsin. Fayl $\lambda(1), \lambda(2), \dots, \lambda(n)$ (1)
yozuvlardan tashkil topgan. Har bitta $\lambda(i)$ yozuvga qandaydir xossa, boshqacha aytganda Kod $\lambda(i)$ ($i=1, n$) kalit berkitilgan deb hisoblaymiz. Odatda kalit-bu qandaydir alohida yozuv sohasi yoki yozuv sohalari kombinasiyasidir. Ushbu kalitlar to'plami elementlari kamaymaslik (o'smaslik) tartibida joylashtirilishi mumkin, deb hisoblansin.

Faylni saralash masalasining qo'yilishi: (1) yozuvlarning shunday ketma-ketlik kombinasiyasi topilsinki, ularning kalitlari kamaymaslik tartibida joylashsin:

$$\mu(1), \mu(2), \dots, \mu(n)$$
$$K_{\mu(1)} \leq K_{\mu(2)} \leq \dots \leq K_{\mu(n)} \quad (2)$$

Ilmiy-texnik masalalarni echishda yozuv ko'pincha kalit sohasidan iborat bo'ladi. (1) fayldan (2) faylni hosil qilish uchun EHM xotirasida fayllarning fizik jihatdan o'rin almashinuvi talab etiladi. Ko'p hollarda (2) o'rin almashinuvni real holda olish talab etilmaydi. Bunda (2) ni u yoki bu usul bilan shunday tavsiflash kerakki, (1) yozuvlarga bevosita murojaat ularning kalitlari ketma-ketligi tartibida amalga oshirilsin. Buni masalan, fayldagi (2) elementlar adreslari ro'yxatini tuzish yo'li bilan amalga oshirish mumkin. (1) uchun bunday ro'yxatni tuzish adresli saralash deb ataladi.

1-Misol. Quyidagi jadvalda 7 ta yozuvdan iborat fayl keltirilgan. Ularning har biri simvolli massivning bitta elementidan iborat bo'ladi. Yozuv alohida 5 ta sohadan iborat: nomer, familiya-ism, kurs, fan, olgan bahosi. Ushbu faylni alfavit tartibida adresli kodlash quyidagi ro'yxatni tuzish bilan amalga oshiriladi:

$$3, 5, 6, 7, 2, 4, 1. \quad (3)$$

No	F.I.SH	Kurs	Fan	Olgan bahosi
1	Qayumova K.	2 Am	Inf. va AT	4
2	Isaev T.	3 Mat	Inf. va AT	4
3	Alieva L.	1 Fiz	Inf. va AT	3
4	Saidov B.	3 PXM	Inf. va AT	5
5	Bozorova N.	4 Mat	Inf. va AT	4
6	Boboev J.	2 Am	Inf. va AT	5
7	Zokirov S.	3 Am	Inf. va AT	5

Bu erda 3 3-yozuvni (Alieva L.),..., 1 1-yozuvni bildiradi.

Ba'zan konkret faylni bir nechta kalit bo'yicha saralashga to'g'ri keladi.

Saralash 2 turga: ichki va tashqi saralashga bulinadi. Ichki saralashda operativ xotiradagi axborotlar qayta ishlanadi, tashqi saralashda tashqi xotiradagi axborotlar qayta ishlanadi. Optimallashtirish muammosi bu ikkala holda bir-biridan farq qiladi. Ichki saralashda kalitlarni taqqoslashlar va fayl yozuvlarining joyini o'zgartirishlar sonini kamaytirishga xarakat qilinadi. Tashqi saralashda mos algoritmlar effektivligining asosiy faktori disk qurilmalariga murojaatlar sonidir.

Bundan keyin faqat ichki saralash haqida gap borib, bir o'ldhovli simvolli yoki sonli massivlardan iborat fayllar bilan ish ko'ramiz. Bunday fayllarning yozuvlari va kalitlari sifatida massivlarning mos elementlari qiymatlarini ko'rib o'tamiz.

2-Misol. Berilgan sonli massivni saralash kerak bo'lsin:

$$7.2, 3, 8, 4, 8, 5.14, 9, 1 \quad (4)$$

Oddiy saralash: 1, 3, 4, 5.14, 7.2, 8, 8, 9.

Adresli saralash: 7.2, 3, 8, 4, 8, 5.14, 9, 1
5 2 6 3 7 4 8 1

Pufakchali saralash. Saralashlarning turli algoritmlari mavjud bo'lib, ulardan eng sodda va ko'p qullaniluvchisi-bu "pufakchali" saralashdir.

$S(a_1, a_2, \dots, a_n)$ massiv berilgan bo'lsin. S massivning a_i va a_j elementlari inversiyani tashkil etiladi deyiladi, qachonki:

$$i > j \text{ uchun } a(i) > a(j) \text{ bo'lsa.}$$

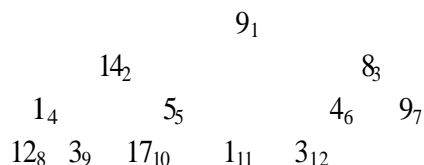
"Pufakchali" saralashning mazmuni shundan iboratki, Bunda S massivning oxiridan boshiga qarab elementlar ketma-ket tekshiriladi va inversiyali qo'shni elementlarning o'rnini almashtirib boriladi. Ko'rilayotgan algoritmlar murakkablik darajasi $O(n^2)$, ya'ni ixtiyoriy S va n uchun algoritmlar S n^2 ta taqqoslash va o'rin almashtirish operatsiyalarini bajaradi. Bu erda S n ga bog'liq bo'lmagan o'zgarmas son. Quyida "Pufakchali" saralashning Beysik algoritmik tilidagi dasturini keltiramiz:

10 REM PUFKSHALI SARALASH

```
20 SCREEN 0: COLOR 15,4: KEY OFF
30 INPUT "ELEMENTLAR SONI" ; N: DIM A (N)
40 FOR I=1 TO N: A(I)=INT(RND(-TIME)*100):NEXT I
50 REM O'ZAK
60 FOR I=2 TO N: FOR J=N TO I STEP -1
70 IF A(J-1)>A(J) THEN SWAP A(J-1), A(J)
80 NEXT J,I
100 FOR I=1 TO N: PRINT A(I);:NEXT I
110 END
```

Ushbu dastur kiritilgan p soni bo'yicha tasodifiy butun sonli (0 dan 99 gacha) massiv yaratadi va uni saralaydi. Algoritm o'zagini 50-90 satrlar tashkil etadi. Bu saralash usuli qo'shimcha xotira talab etmaydi, ammo ko'p vaqt talab etadi.

Daraxt usulida saralash. Saralashning barcha usullari S massiv elementlarini ko'rib chiqish va ular ustida qandaydir amallar bajarishdan iboratdir. Bunday algoritmlardan biri saralanayotgan S massivni binar D daraxt ko'rinishida ifodalashdir. Quyida uning sxematik tasvirini keltiramiz:



Bunda S massiv: 9 14 8 1 5 4 9 12 3 17 1 3 elementlaridir;

Bu erda $8 \leq n \leq 16$; 1 dan boshlangan natural sonlar bilan yuqoridan pastga va chapdan ungga qarab D daraxtning barcha uchlari nomerlab chiqilgan. Ushbu nomerlar adreslar rolini bajaradi. Binar D daraxtda bitta ildiz tugun bo'lib, uning ajdodi bo'lmaydi. Tugunning adresi -1; ixtiyoriy boshqa tugunlar bitta ajdodga va bitta yoki ikkita avlodga ega bo'ladi. Ba'zi hollarda daraxtlar ikki o'lchovli massivlar ko'rinishida har bir tugunning ajdod va avlodlari uchun adreslari oshkor tarzda ifodalanadi $k(k=1, n)$. Ammo real holatda bu adreslarni saqlash emas, k nomer bo'yicha hisoblash osonroqdir:

- a) ajdodlar uchun: $x(k) = k/2, k=1, 2, \dots, p;$
 $2^*k, k=1, 2, \dots, n/2$
- b) chapdagi avlod uchun $u(k) =$
 $0, k > n/2$
- v) o'ngdagi avlod uchun $z(k) =$
 $2^*k+1, k=1, 2, \dots, (n-1)/2$
 $0, k > (n-1)/2$

Daraxtdagi ixtiyoriy tugun boshqa daraxt uchun ildiz vazifasini bajarishi mumkin.

Piramidali saralash. Piramidali saralash Dj.Uilyame tomonidan taklif etilgan va R.Floyt tomonidan rivojlantirilgan. Bunda S massiv D binar daraxt ko'rinishida ifodalanadi va qo'shimcha xotira talab etmaydi. Algoritmning bajarilish murakkabligi $O(n \log_2 n)$ ga teng.

$$S(1), S(2), \dots, S(n) \quad (5)$$

massiv berilgan bo'lsin. (5) elementlarning

$$S(p), S(p+1), \dots, S(q) \quad (1 < p < q < n) \quad (6)$$

Ko'rinishdagi ketma-ketlik piramida deb ataladi, qachonki quyidagi shartlardan biri bajarilsa:

- 1) $2p > q$
- 2) $1p = q, S(p) > S(q)$
- 3) $2p < q, S(j) > S(2j) \quad (p < j < q/2)$

Ta'rifdan quyidagilar kelib chiqadi:

- 1) Ixtiyoriy (5) ketma-ketlik uchun $S(n/2+1), S(n/2+2), \dots, S(n)$ ketma-ketlik piramida bo'lib hisoblanadi;
- 2) Agar (5) ketma-ketlik piramida bo'lsa, u holda $S(1) > \max S(j)$ (7)

- 3) Agar (5) ketma-ketlik piramida bulib, binar D daraxt kurinishida berilgan bo'lsa, D dagi ixtiyoriy tugunning qiymati uning chap va o'ng avlodlari qiymatidan kichik bo'lmaydi.

2-Misol. 90, 70, 11, 8, 3, 9, 7, 5, 6, 1, 2 ketma-ketlik berilgan va u piramidadir:

```

          90
        70      11
      8      3      9      7
    5      6      12
  
```

Piramidali saralash ikki etapdan iborat buladi:

1-etap. Piramidani qurish. (5) ketma-ketlikda

$$S(n/2+1), S(n/2+2), \dots, S(n) \quad (8)$$

Piramidadir. (8) ketma-ketlikka (5) dan qolgan elementlarni qo'shamiz.

$S(j+1), S(j+2), \dots, S(n)$ piramida bo'lsin. Chapdan $S(j)$ elementni qo'shib,

$$S(a), S(j+1), S(j+2), \dots, S(n) \quad (9)$$

(9) ni yana piramidaga aylantiraymiz, ya'ni $S(j)$ va uning ikkita avlodi $S(2j)$ va $S(2j+1)$ lar tekshiriladi. Bunda agar $S(j)$ avlodlaridan kichik bo'lmasa hisoblashlar to'xtatiladi, chunki (9) piramida bo'lib hisoblanadi. Aks holda $S(j)$ va $\max(S(2j), S(2j+1))$ qiymatlarni almashtiramiz va h.k.z. Oxirida (5) piramidga aylanadi va (7) bajariladi.

Olingan S piramidani joriy deb e'lon qilamiz va 2-etapga o'tamiz.

2-etap. Joriy S piramidada 1-element qolganlaridan kichik emas. S ning chekka elementlari qiymatlarini o'zaro almashtirib, S ni ung tomondan bittaga qisqartiramiz. Hosil bo'lgan ketma-ketlik piramida bo'lmasligi ham mumkin. $S(1)$ element uchun 1-etapdagi jarayonni qo'llab, o'zgartirilgan S ketma-ketlik yana piramidaga aylantiriladi. 2-etapni $p-1$ marata bajarib, S ni o'smaslik tartibida saralab olamiz.

Ushbu saralash usulini konkret misolda ko'rib o'tamiz.

4-misol.

23, 77, 12, 7, 44, 82, 16, 53 ketma-ketlik uchun piramidali saralash o'tkazamiz. Bunda algoritm bajarilish jarayonidagi S ketma-ketlikning joriy elementlari yozib olinsin.

Quyida S ketma-ketlikning algoritm bajarilishining hap bir 1 va 2 etap realizasiyasidagi qiymatlari ko'rsatilgan.

Piramidani qurish

23	77	12	7	44	82	16	53
23	77	12	53	44	82	16	7
23	77	82	53	44	12	16	7
23	77	82	53	44	12	16	7
82	77	23	53	44	12	16	7

Piramidani saralash

7	77	23	53	44	12	17	82
16	53	23	7	44	12	77	82
12	44	23	7	16	53	77	82
12	16	23	7	44	53	77	82
7	16	12	23	44	53	77	82
12	7	16	23	44	53	77	82

7	12	16	23	44	53	77	82
---	----	----	----	----	----	----	----

Piramidali saralash usulining analizi shuni ko'rsatadiki, uning bajarilishi uchun $3n \log_2 n$ tadan ko'p bo'lmagan elementar operatsiya bajarilishi talab etiladi.

Quyida bir o'lchovli massivni kamaymaslik tartibida piramidali saralshaning Beysik algoritmik tilidagi dasturi keltirilgan:

```

10 REM PIRAMIDALI SARALASH
20 PRINT SARALASH VA=TI T:
30 PRINT N=100, T=19 SEK
40 PRINT N=500, T=2 MIN 8 SEC
50 PRINT N=1000, T=4 MIN 47.7 SES
60 PRINT N=2000, T=10 MIN 37.1 SES
70 PRINT KIRITISH
80 SCREEN 0: COLOR 15,4: KEY OFF
90 PRINT "PIRAMIDALI SARALASH"
100 PRINT
110 INPUT "ELEMENTLAR SONI" ; N: DIM A (N)
120 SLS
130 LOSATE 8,9: PRINT "XISOBLASHLAR"
140 GOSUB 330
150 TIME=0:K=N: PRINT "O'ZAK"
160 FOR J=N/2 TO 1 STEP-1:GOSUB 210: NEXT
170 FOR K=N-1 TO 1 STEP -1
180 SWAP S(1),S(K+1):X=1: GOSUB 210
190 NEXT: GOTO 260
200 PRINT
210 Y=X+X: ON SGN(Y-K)+2 GOTO 220,230,250
220 IF S(Y) <S(Y+1) THEN Y=Y+1
230 IF S(X)>=S(Y) THEN 250
240 SWAP S(X),S(Y):X=Y:GOTO 210
250 RETURN: PRINT "SHI=ARISH"
260 SLS
270 PRINT "SARALASH VA=TI -"; TIME/50; 'SEK"
280 PRINT "ELEMENTLAR SONI-"; N: PRINT
290 PRINT "MASSIV";
300 FOR J=1 TO N: PRINT S(J);:NEXT J
310 END: PRINT "PIRAMIDA DASTURI"
320 PRINT
330 FOR J=1 TO N : S(J)=INT(RND)(1)*100):NEXT J
340 RETURN

```

Dasturni o'smaslik bo'yicha saralashga aylantirish uchun 220-230-satrlardagi "<" va ">q" amallarini mos ravishda ">" va "<q" amallari bilan almashtirish kifoya bo'ladi. Dastur o'zagini 150-250-satrlar tashkil qiladi. 160-satr piramidani quradi, 170-190- satrlar esa u ni

saralaydi. Hisoblashlarning har ikkala etapida ham 210-250-satrlardagi "Elash" qism dasturidan foydalaniladi.

Tez saralash

K.Xoorning Tez saralash algoritmi bo'lishli saralash deb ataldi. Ushbu algoritmi boshqa saralash usullariga nisbatan vaqt bo'yicha yaxshi natijalar ko'rsatadi. Tez saralash usulining mohiyati quyidagidan iborat:

$S(k)$ ($k=1,2,\dots,n$)- bir o'lchovli massiv berilgan bo'lsin. $x \in S(k)$ dan olingan qandaydir element bo'lsin. Bunda S shunday ikkita S_1 va S_2 ($S_1 \cup S_2 = S$) kesishmaydigan bo'sh emas qismlarga bo'linadiki, S_1 dagi elementlar x dan katta bo'lmasin, S_2 dagi elementlar esa x dan kichik bo'lmasin:

6,23,17,8, 14,25,6,3,30,7

$x=14$; S ni qandaydir $a > x$ element uchraguncha tekshiramiz: aq23; Keyin S ni qandaydir $b < x$ element topilguncha undan chapga tekshiramiz: bq7; a va b larning o'rinlarini almashtiramiz. Jarayonni davom ettirib, quyidagi ketma-ketlikka ega bo'lamiz:

6, 7, 3, 8, 6 | 14, 25, 17, 30, 23.

Shunday qilib, S_1 va S_2 bo'laklar ham xuddi yuqoridagi kabi saralanadi. Jarayon har bir bo'lakda bittadan element qolguniga qadar davom ettiriladi. Natijada saralangan massivga ega bo'lamiz. quyida biz Xoor algoritmining Beysik algoritmik tilidagi dasturini keltiramiz:

10 REM TEZ SARALASH

20 PRINT SARALASH VA=TI T:

30 PRINT N=100, T=16 SEK

40 PRINT N=500, T=1 MIN 31 SEK

50 PRINT N=1000, T=3 MIN 14.2 SEK

60 PRINT N=2000, T=6 MIN 18.7 SEK

70 PRINT N=3000, T=10 MIN 18.7 SEK

90 PRINT

100 SCREEN 0: SOLOR 15,4: KEY OFF

110 DEFINT I,J,A,B,K,T,N

120 PRINT "TEZ SARSLASH"

130 PRINT

140 INPUT "ELEMENTLAR SONI" ; N: SLS

150 LOSATE 8,9: PRINT "XISOBLASHLAR"

160 DIM S(N), T1(13),T2(13): GOSUB 380

170 TIME=0: PRINT "STZAK"

180 K=1:T1(1)=1:T2(1)=N:PRINT"STEKNOMI"

190 A=T1(K): B=T2(K): K=K+1: PRINT "STEKDAN O'='ISH"

200 I=A:J=B:X=(A+B)/2: PRINT "AJPATISH"

210 IF S(I)<X THEN I=I+1: GOTO 210

220 IF X<S(J) THEN J=J-1: GOTO 220

```

230 IF K=J THEN SWAP S(I),S(J): 1=1+1 :J=J+1
240 IF K=J THEN 210
250 IF J-A>=B-I THEN 280: "STEKKA YOZISH"
260 IF KB THEN K=K+1:T1(K)=I: T2(K)=B
270 B=J: IF A<B THEN 200 ELSE 300
280 IF A<J THEN K=K+1: T1(K)=A: T2(K)=J
290 A=I:IF A<B THEN 200
300 IF K>0 THEN 190: PRNT "CHIQARISH"
310 SLS: PRINT "SARALASH VAQTI -"; TIME/50; 'SEK"
320 PRINT "ELEMENTLAR SONI-"; N: PRINT
330 PRINT ""
340 PRINT "MASSIV";
350 FOR U=1 TO N: PRINT S(U);:NEXT U
310 END: PRINT "TEZ SARALASH"
320 PRINT
330 FOR J=1 TO N : S(J)=INT(RND)(1)* 1000):NEXT J
340 RETURN

```

Nazorat uchun savollar:

1. Ichki saralash deganda nimani tushunamiz?
2. Pufakchali saralash usuli va uning mohiyati nimada?
3. Piramidali saralash usuli va uning mohiyati nimada?
4. Tez saralash usuli va uning mohiyatim nimada?

Foydalanilgan adabiyotlar:

1. А.Р. Есауан и др. Информатика. М.:Просвещение,1991.201-212 с.
2. <http://structur.h1.ru/hash.htm>

13 -Mavzu . Tashqi saralash algoritmlari (2 soat)

Reja:

- 1.Diskli xotira qurilmasining tuzilishi
- 2.Bouz-Nelson algoritmi
3. Ketma-ket qo'shib olish usulida saralash
- 4.Takrorlanuvchi balansli saralash

Kalit so'zlar: Fayl, Silindrlar, Treklar, Adreslash, Birlashuv

Tashqi saralash jarayoni tashqi xotirada saqlanuvchi axborotlarni saralash vazifasini bajaradi. Tashqi saralash jarayoni ichki saralashdan katta farq qiladi. Chunki tashqi fayllarga murojaat to'g'ridan –to'g'ri emas, ketma-ket(bolab) usulda amalga oshiriladi. Bunda axborotlarni faqat bloklab o'qish mumkin. Tashqi saralash jarayonini tushunish uchun diskarning fizik tuzilishi bilan umumiy tanishib chiqish kerak bo'ladi. Disklarning tashqi sirti magnitli qoplamga ega bo'lib, ular doimiy katta tezlik bilan o'z o'qi atrofida aylanadi. Disklarning har bir ish sohasiga bitta o'qish-yozish qurilmasi o'rnatilgan. Axborotlarga murojaat vaqtida o'qish-yozish qurilmasi tomonidan treklar deb ataluvchi diskdagi ma'lumotlar yozilgan

yo'lakchalardan berilganlar o'qiladi. Bu treklar yig'indisi joriy silindr deb ataladi. qish-yozish qurilmalari maxsus shtangaga o'rnatilgan bo'lib, bu shtanga burilganda o'qishqyozish qurilmasi boshqa silindrga o'tqaziladi. Silindrlar shtanganing bir tomonga xarakati vaqtida o'qish-yozish qurilmalari blokingularga murojaat qilish tartibida nomerlanadi. Berilganlar elementlari orasidagi masofa ular joylashgan silindrlar nomerlari farqidan iborat bo'ladi. Xotira elementlarini adreslash ular joylashgan silindr doirasida amalga oshiriladi. Fayl adreslar tartibi bo'yicha yoziladi, ammo bo'sh joy bo'lmaganda, boshqa silindrga ham yozilishi mumkin. Diskdagi axborotlarga murojaat asosiy xotiradagi axborotlarga murojaatdan anchagina sekin amalga oshiriladi. Chunki bunday murojaat vaqti bu jarayonda bir necha bajariladigan amallarga ketadigan vaqtdan kelib chiqadi:

a) Silindr kerakli elementining o'qishqyozish qurilmasi tagidan o'tishini kutish vaqti;

b) o'qish-yozish qurilmasining boshqa silindrga o'tqazilishini kutish vaqti;

v) Tashqi saralash vaqti;

Tashqi saralash vaqti ham o'z navbatida bir nechta amallar bajarilishiga ketadigan vaqtdan hosil bo'ladi:

a) fayl qismlarining ichki saralanishi;

b) berilganlarning ko'r marta diskka yozilishi va o'qilishi;

v) o'qish-yozish aktlari orasidagi golovka yurishlari;

g) tartiblangan qismlarning birlashuvi jarayonidagi xotiradagi xarakatlar;

Tashqi xotiradagi fayllarni saralash muhim amaliy ahamiyatga egadir. Bunday saralash jarayoni natijasida tashqi xotiradagi axborotlarga murojaat vaqti sezilarli kamaytiriladi va xotiraga axborotlar o'qish-yozish jarayoni ancha tezlashadi.

Tashqi xotiradagi fayllarni saralash fayl bloklari utida bajariladi. Bunday saralash algoritmlaridan biri **Birlashuv yo'li bilan saralash algoritmidir**. Birlashuv tushunchasi ikki yoki undan ortiq tartiblangan ketma-ketliklarning bitta tartiblangan ketma-ketlikka ayni paytda joriy elementlarni siklik tanlash yordamida almashtirish(keltirish) ni bildiradi. Birlashuv jarayoni saralash jarayonlari ichidaeng sodda jarayon hisoblanadi. Birlashuv jarayonini realizasiya qiluvchi bir nechta algoritm mavjud. Bulardan biri **Bouze-Nelson algoritmidir**.

To'g'ridan to'g'ri birlashuv. Algoritm quyidagi qadamlardan iborat:

1. A ketma-ketlik V va S ketma-ketliklarga ajratiladi;
2. V va S ketma-ketliklar alohida elementlarining tartibli juftlashtirilishi yo'li bilan birlashtiriladi;
3. Olingan ketma-ketlik A deb belgilanib, 1-2-qadamlar takrorlanadi. Bunda tartiblangan juftliklar tartiblangan to'rtliklarga birlashadi.
4. Oldingi qadamlar takrorlanib, to'rtliklar sakkizliklarga birlashadi va jarayot butun ketma-ketlik tartiblangunga qadar davom etadi.

Masalan, Aq4455124294180667 ketma-ketlik berilgan bo'lsin. Algoritmik qadamlar ketma-ketlikni quyidagicha saralaydi:

1. V=44551242

S=94180667

A=4494 1855 0612 4267

2. V=44941855

S=06124267

A=06124494 18425567

3. V=06124494

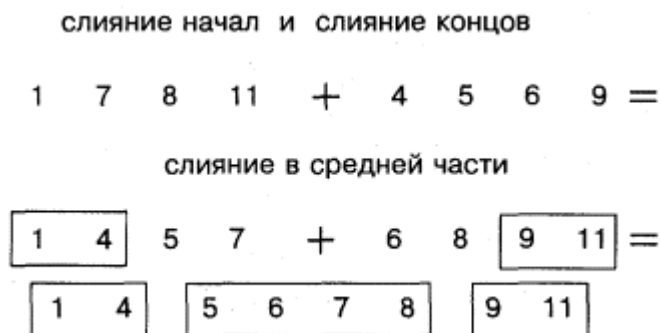
S=18425567

A=0612184244556794

Berilganlarning barcha to'plamlarini qayta ishlovchi jarayon faza deb ataladi. Takrorlanib, saralash jarayonini tashkil qiluvchi qism jarayon etap deb ataladi. Bizning misolimizda saralash

3 etapdan iborat. har bir etap bo'linish va birlashish fazalaridan iborat. Ushbu Birlashuv algoritmining eng katta kamchiligi saralanuvchi berilganlar tomonidan egallangan xotira xajmi saralash jarayonida ikki martaga oshishi hisoblanadi. quyida qo'shimcha xotira talab etmaydigan saralash algoritmini ko'rib chiqamiz.

Rekursiv birlashuv algoritmi. Ushbu algoritmnining mohiyati quyidagidan iborat: ikkita teng tartiblangan qismlarni birlashtirish ularning birinchi yarimqismlarini va ikkinchi yarim qismlarini mos ravishda birlashtirish hamda birinchi natijaning ikkinchi yarmi bilan ikkinchi natijaning birinchi yarmini birlashtirish orqali amalga oshiriladi. Masalan:



Agar qismlar teng bo'lmasa, «yarimlarni» birlashtirish jarayoni «to'rtliklar», «sakkizliklarni» va h.k.z. larni birlashtirishga keltirilishi mumkin. Bunda quyidagi rekursiv jarayon amal qiladi:

Const n=200;

Type

tipkl=word;

tip = Record

kl: tipkl;

z:Array[1..50] of real

End;

Var

A: Array[1..n] of tip;

j:word;

Procedure Bose (Var AA; voz:Boolean);

Var

m,j:word; x:tip; { tip - tip сортируемых записей }

A: Array [1..65520 div Sizeof(tip)] of tip Absolute AA;

Procedure Sli(j,r,m: word); { r – birlashuvchi qismlar orasidagi masofa, m – ularning xajmi }

j – yozuvning eng kichik nomeri }

Begin

if j+r<=n Then

If m=1 Then

Begin

If voz X or (A[j].kl < A[j+r].kl) Then

Begin

x:=A[j];

A[j]:= A[j+r];

A[j+r]:=x

End;

End;

Else Begin m:=m div 2;Sli(j,r,m); { birinch yarim qismlarning birlashuvi }

If j+r+m<=n Then

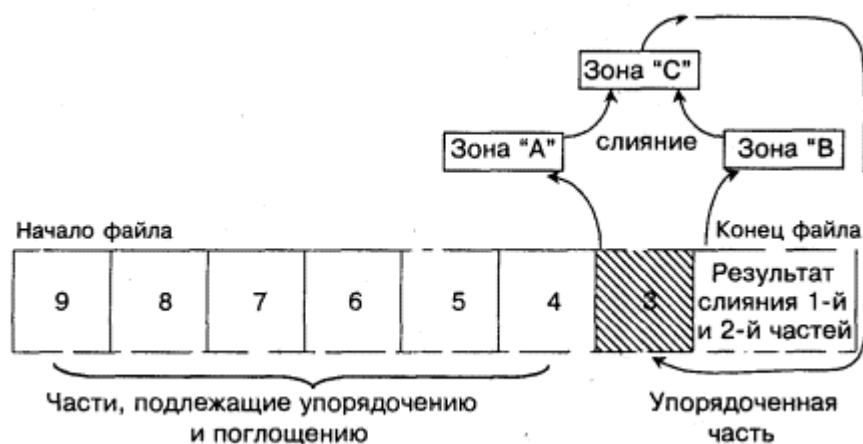
Sli(j+m,r,m); { ikkinchi yarim qismlarning birlashuvi }

```

Sli(j+m,r-m,m) End; { markaziy qismlarning birlashuvi }
End; { Sli blokining oxiri }
Begin m:=1;
Repeat
j:=1; { teng xajmli ro'yxatlar bitlashuvi sikli: }
While j+m<=n do
Begin Sli(j,m,m); j:=j+m+m
End;
m:=m+m { yangi etapdan oldin ro'yxat xajmining ikkilanishi }
Until m >= n { Barcha birlashuvlar daraxtini realizasuyalivchi sikl oxiri }
End{ Bose bloki oxiri };
BEGIN
Randomize;
For j:=1 to n do
begin
A[j].kl:= Random(65535);
Write(A[j].kl:8);
end;
Readln;
Bose(A,true);
For j:=1 to n do
Write(A[j].kl:8);
Readln
END.

```

Ketma-ket qo'shib olish usulida saralash. Algoritm bir nechta fayl qismlariga ega bo'lgan holda, shulardan ikkitasini birlashtirishdan boshlanadi. So'ngra qolgan qismlar ham ketma-ket tartiblangan qismga qo'shib olinadi. Ushbu jarayon quyidagi etaplardan iborat:

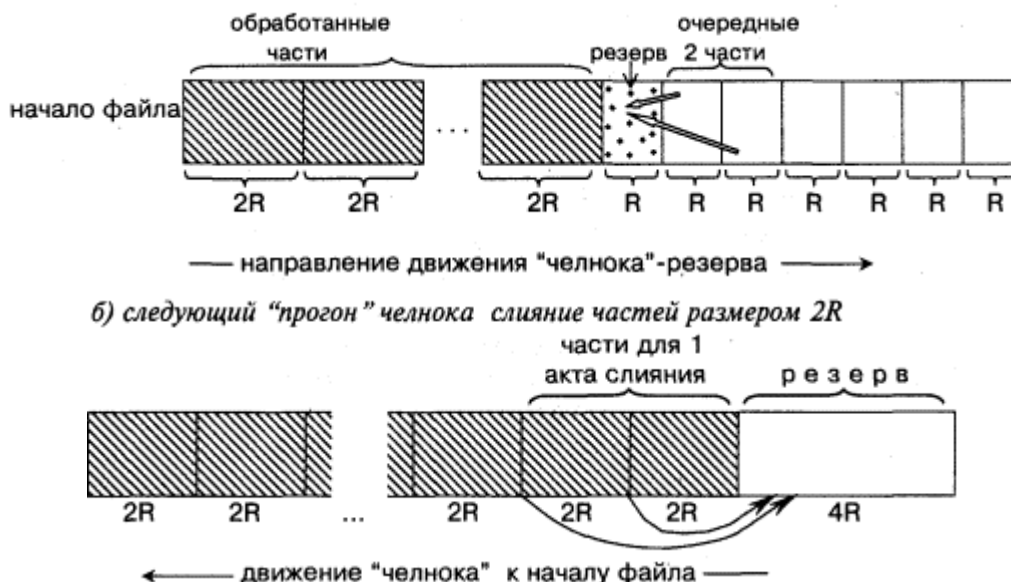


qo'shib olinishdan oldin navbatdagi fayl qismi xotiraning maxsus «A» sohasiga chaqiriladiva shu erda saralanib, qoldiriladi. Faylning oldin tartiblangan qismining boshi «V» soha ga chaqiriladi va birlashtirish jarayoni bajariladi. Bunda «V» sohadagi axborotlar davriy ravishda to'ldirib turiladi. bunda birlashuv natijalari «S» soha ga ketma-ket uzatiladi. «S» sohadan tartiblangan fayl qismi faylning ayni paytda qo'shib olinuvchi qismi joyiga joylashtiriladi. Ushbu jarayonda etaplar soni kam bo'lib, faylning katta bo'lmagan xajmlarida tez bajariladi.

Takrorlanuvchi balansli birlashuv usuli. Bu saralash usulining birinchi etapida ichki saralash usullaridan foydalanib, faylning M ta tartiblangan katta R xajmli qimslari yaratiladi. Ularga nisbatan To'g'ri birlashuv algoritmi qo'llaniladi. Bunda qo'shimcha disk sohasi ajratilib, bu soha bevosita birlashuvchi qfayl qismlaridan oldin yoki keyin joylashtiriladi. Birlashuv jarayoni

tugallangach, bu soha navbatdagi fayl qismlariga o'tqaziladi. Bu soha xajmi fayl qismlari xajmidan kam bo'lmaydi. Bunda ikki fayl qismining birlashuvi natijasi birinchi fayl qismi bilan rezerv xotira qismiga joylashtiriladi. Ikkinchi fayl qismining joyi esa bo'shaydi. Ushbu bo'shagan joy keyingi birlashuvchi fayl qismlari uchun rezerv vazifasini bajaradi. Birlashuv jarayoni natijasida rezerv xotira qismi fayl boshidan fayl oxiriga siljib boradi va aksincha. Saralash jarayoni davomida rezerv xotira qismi kattalashib boradi, chunki birlashuvchi qismlar ning xajmi ortib boradi.

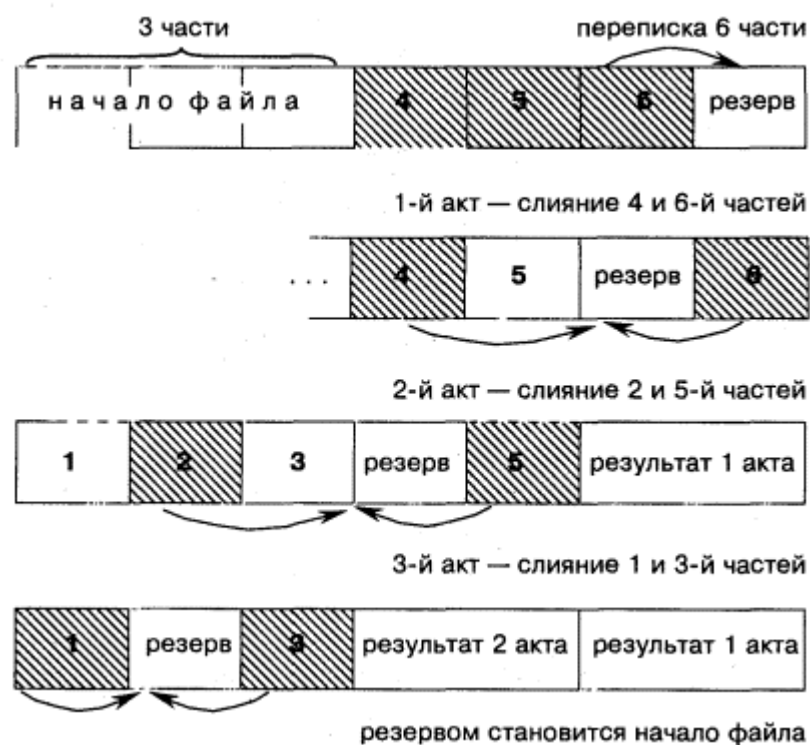
a) R xajmli fayl qismlarining birlashuvi (oraliq holat)



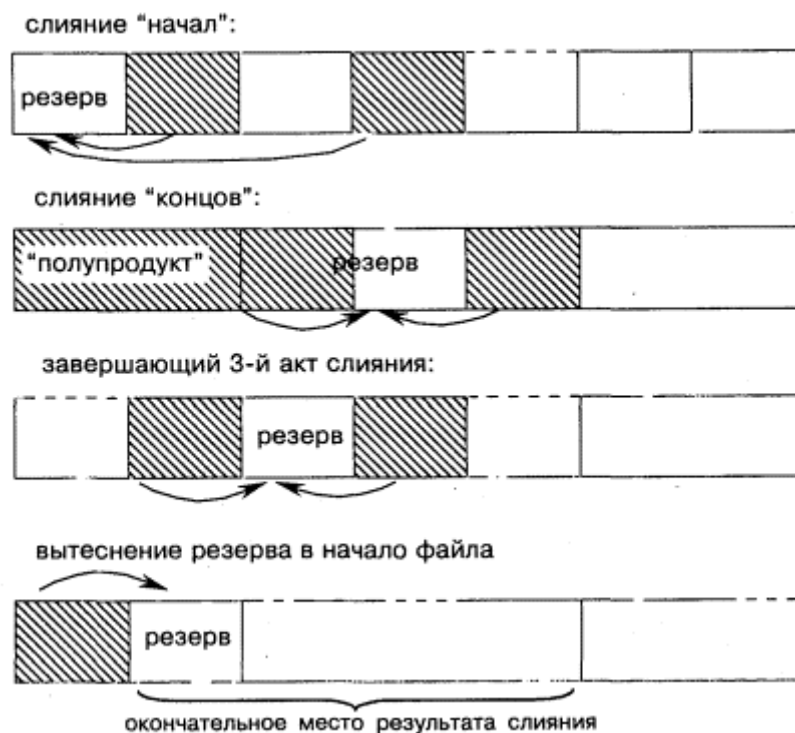
Strelkalar bilan berilganlarning birlashuv paytidagi siljishi ko'rsatilgan.

Rezerv xotira qismi fayl oxiriga etganda kattalashadi. Bu jarayon har ikki etapda yuz beradi. Rezerv xotiraning o'sib borishini chegaralash uchun birlashuvning tugallovchi etaplari modifikatsiyalanadi va rezerv xotira xajmining maksimal qiymati $D \cdot q \cdot 1/6$ fayl xajmiga teng bo'lishiga erishiladi. Buning uchun esa dastur rezerv xotira xajmi va uning pozitsiyasini (fayl boshida yoki oxirida) shunday belgilashi kerakki, saralash jarayonining uchta katta fayl qismi qolga vaqtida bu rezerv xotira qismi fayl boshida tursin.

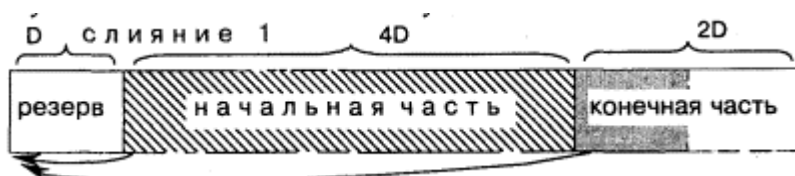
v) 6 ta fayl qismi qolgandagi birlashuv etapi:



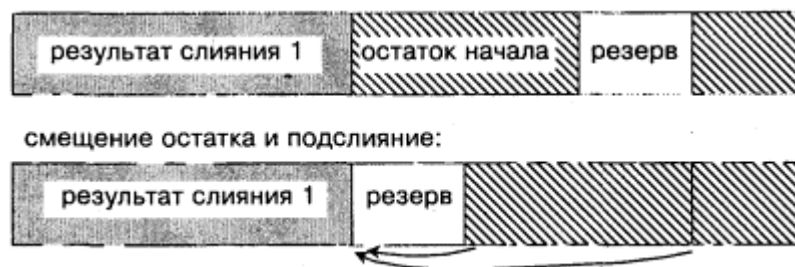
g) «yarimlatib» birlashtirishyoki 3 ta fayl qismi qolgandagi birlashuv etapi. Bunda faylning oxirgi qismi birlashuvd qatnashmaydi.



d) Oxirgi etap; oldin fayl oxirgi qismining birinchi yarmi va butun boshlang'ich qism birlashuv uchun olinadi:



1-birlashuvdan keyin natijaning oxiri uchun joy bo'shatish va birlashuvni davom ettirish maqsadida boshlang'ich qismning qolganini surish kerakmi yoki yo'qligi aniqlanadi. Agar birlashuv jarayonida boshlang'ich qism to'la birlashgan bo'lsa, birlashuv to'xtalib, rezerv xotira qismi fayl oxiriga suriladi, ya'ni fayl oxiri rezerv xotiraning joyiga ko'chirib ;tqziladi.



Nazorat savollari:

1. Tashqi saralashning mohiyati nimada?
2. Qanday tashqi saralash algoritmlari bor?
3. Tashqi xotiradagi axborot o'qish-yozish tezligi qanday
4. Parametrlarga bog'liq bo'ladi?
5. Tashqi saralashdan maqsad nima?
6. Bouz-Nelson algoritmining mohiyati nimada?
7. Ketma-ket qo'shib olish usulida saralashning mohiyati nimada?
8. Takrorlanuvchi balansli birlashuv usulida saralashning mohiyati nimada?

Foydalanilgan adabiyotlar:

<http://structur.h1.ru/hash.htm>

14-Mavzu: Izlash algoritmlari

Reja:

1. Oddiy ko'rib chiqish va binar izlash algoritmlari
2. Vinar daraxtda izlash algoritmlari
3. Raqamli izlash daraxtlari

Kalit so'zlar: Binar izlash, Raqamli izlash daraxti, Massiv

Juda ko'p amaliy masalalar izlash algoritmlariga keltiriladi. Izlash – bu oldindan yig'ilgan katta hajmdagi axborotlar majmuasi ichidan konkret ma'lumotni qidiruv jarayonidir. Berilganlar yozuvlardan iborat bo'lib, har bir yozuv kalitni o'z ichida saqlaydi. Bu kalitlar yozuvlarni bir-biridan farqlash uchun ishlatiladi. Izlash maqsadi berilgan kalitga to'g'ri keluvchi barcha yozuvlarni topishdan iborat. Oldin foydalanuvchi nuqtai nazaridagi izlashni ko'rib o'tamiz. Izlash jarayonlarini quyidagicha klassifikatsiyalash mumkin:



Izlash jarayonlarining ushbu klassifikatsiyasini izlash vositalarini klassifikatsiyasidan farqlay bilish kerak. Ixtiyoriy izlash usulini turli algoritmlar yordamida amalga oshirish mumkin.

Yozuvlarni oddiy ko'rib chiqish usuli. Bu usulni quyidagi algoritm yordamida realizatsiya qilish mumkin:

```

function search(x: integer): integer;
var
i: integer;
begin
for i:=1 to n do
begin
if x = a[i] then
begin
search := i;
exit;
end;
end;
search:=0;
end;
  
```

Бинар излаш(дихотомия) usuli. Ushbu algoritmning mohiyati quyidagidan iborat: Saralangan massivda massiv o'rtasi qidiriladi. Agar izlangan element massiv o'rtasidagi elementdan kichik bo'lsa, chap tomonda izlaymiz, katta bo'lganda esa o'ng tomonda izlanadi. Topilgan intervalda yana o'rtacha element izlanadi va taqqoslash bajariladi va h.k.z.

```

Type fun= function (j:word):Boolean;
Function fa(j:word):Boolean; far; Begin fa:= A[j] < x1 End;
Function fb(j:word):Boolean; far; Begin fb:= A[j] > x2 End;
Procedure Search(f1,f2:fun; L:integer; var m,k,usp:word);
Var i,j,kk:word;
Begin usp:=1; i:=m+1; kk:=k;
If L<>1 Then
Begin {f2 funksiyasi bo'yicha binar izlash sikli: }
Repeat j:=(i+kk) div 2; If f2(j) Then kk:=j Else i:=j+1
Until i=kk; If i=m+1 Then usp:=0
Else If (L > 1) And f1(i-1) Then usp:=0
End; {i – argumentdga eng yaqin katta kalitli topilgan yozuv nomeri} If usp = 0 Then Exit;
If (L > 2) Or (L = 1) Then
  
```

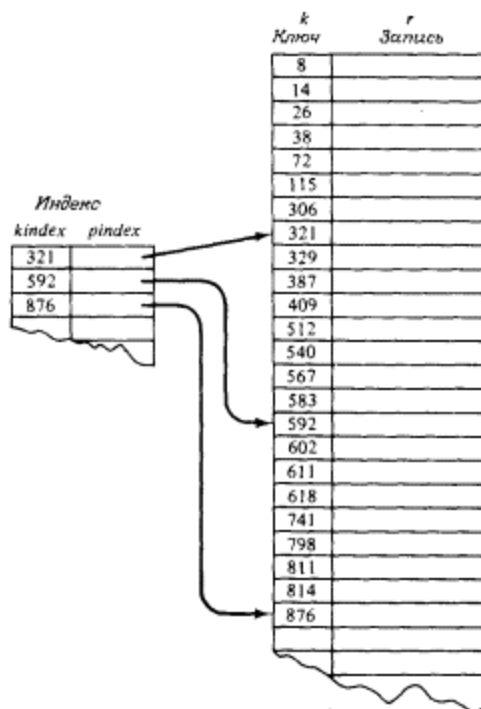
Begin $i := k - 1$; {izlanayotgan yozuvlarni pastdan chegaralaydigan yozuv topiladi; f1 funksiyasi bo'yicha binar izlash sikli}
 Repeat $j := (m + i + 1) \div 2$; If $f1(j)$ Then $m := j$ Else $i := j - 1$
 Until $m = i$;
 If $L = 1$ Then Begin If $m = k - 1$ Then $usp := 0$; $i := m + 1$ End
 End
 Else If $(L = -1)$ Or Not $f1(i - 1)$ Then $i := i - 1$;
 $m := i$; $k := k + 1$
 End;

$L = -1$ ($L = 1$) bo'ladi, pastdan(yuqoridan) yaqinlashish orqali izlash holida; $L = 2$ bo', mos tushish bo'yicha izlash holida(bitta yozuv); $L = 3$ bo'ladi, barcha yozuvlar mos tushishi bo'yicha izlash holida.

$L = 3$ va $usp = 1$ (izlash jarayoni muvaffaqiyatli) bo'lganda topilgan m (Eng kichik), k (eng katta) lar izlanayotgan yozuvlar guruxlariga qo'shni pozitsiyalarni belgilaydi, qolgan hollarda, ya'ni $usp = 1$ ("muvaffaqiyat") bo'lganda, topilgan yozuv nomeri m dan iborat bo'ladi.

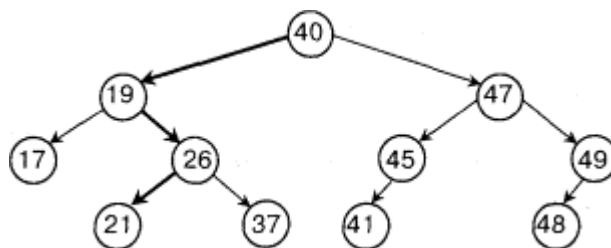
Izlash argumenti oshkor ko'rsatilmagan. Bu argument foydalanuvchi tomonidan yoziladigan mantiqiy $f1$, $f2$ bitta j yozuv nomeridan iborat bo'lgan parametrlilik funksiyalarda berkitilgan. $f1$ ($f2$) da yozuv kaliti argumentdan kichik(katta) bo'lganda $f1$ ($f2$) rost deb yoziladi. m , k – yozuvlarning natijaviy nomerlarigina bo'lmasdan, izlash natijaviy sohasining tashqi chegaralari hamdir.

Indeksli ketma-ket izlash metodi. Ushbu usul saralangan faylda izlash jarayoni effektivligini oshiradi, ammo u qo'shimcha xotira sohasini talab etadi. Bunda saralangan faylga qo'shimcha sifatida indeks deb ataluvchi yordamchi jadval kiritiladi. Indeksning har bir elementi kindex kalitidan va ushbu kalitga mos keluvchi fayldagi yozuv ko'rsatkichidan iborat bo'ladi. Indeksdagi elementlar fayldagi elementlar kabi ushbu kalit bo'yicha saralanishi kerak. Agar indeks faylning 1/8 qismiga teng xajmga ega bo'lsa, fayldagi har bir 8-yozuv indeksda ifodalanadi. Bu quidagi tasvirda ko'rsatilgan:



Binar Daraxtda izlash. Ushbu izlash algoritmi amalda keng qo'llanilib, anchagina sodda va effektiv izlan usuli bo'lib hisoblanadi.

quyidagi daraxtni ko'rib o'tamiz:



Mos tushishlar bo'yicha izlash juda oson usul hisoblanib, uning mohiyati quyidagicha: agar izlanayotgan yozuv kalitdan kichik bo'lsa, chapga yuramiz va o'ngga yuramiz, aksincha bo'lganda.

Yaqinlik bo'yicha izlash. Bunda daraxtni ko'rib chiqishda izlash yo'lidagi tugunlarga ko'rsatkichlarni stekka kiritib boramiz. 20 ga teng izlash argumentida 21 da izlashni to'xtatamiz va stekning boshidan 20 ga yaqin sonni aniqlaymiz.

Interval bo'yicha izlash. Bunda chap yoki unga eng maksimal yaqinlashgan chegara topiladi. So'ngra stekni teskari tartibda ko'rib chiqish yo'li bilan o'ng chegarani qidiramiz, yaqni chap chegaradan katta yoki teng va o'ng chegaradan kichik tugunlarni qidiramizamiz.

Procedure Obrab(ss:pt);

Begin Write(ss^.kl:5) End; { imitiruyumaya obrabotku uzlov }

Procedure Poisk(cor:pt; ar1, ar2:tipkl; l:integer; var pp:pt);

Label 1,2; { sl - BDU tuguniga ko'rsatkich sohasi, ssl - stekdagi bog'lanish }

Type pnt=^z;

z=Record

sl:pt; ssl:pnt End;

Var rr,ss,tt: pt; q,t: pnt;

Procedure Stekin; { BDU tuguniga t ko'rsatkichni stekka qo'shish }

Begin New(q); q^.sl:= ss; q^.ssl:=t; t:=q End;

Begin pp:= Nil; rr:= Nil; t:= Nil; tt:= cor;

If l = 3 Then ar2:=ar1;

While tt <> Nil do { Izlash jarayoni sikli }

Begin ss:= tt;

If ss^.kl = ar1 Then { kalitning mos tushish holati }

Begin pp:= ss; Stekin; Goto 1 End;

If ss^.kl > ar1 Then Begin tt:= tt^.lson; Stekin End

Else Begin tt:= tt^.rson; rr:= ss End;

End;

If (l <> 2) And (t <> Nil) Then pp:= t^.sl;

If l=-1 Then pp:= rr;

1: If pp <> Nil Then If l < 3 Then Obrab(pp)

Else If t^.sl^.kl > ar2 Then pp:= Nil

Else { l=3,4: uchun ishning 2-ETAPI }

While t <> Nil do { BDU ni chapdan qisman ko'rib chiqish sikli }

Begin q:= t; ss:= q^.sl; t:= t^.ssl; Dispose(q);

If ss^.kl > ar2 Then Goto 2; { Ko'rib chiqishni tugallash ehtimoli }

Obrab(ss); ss:= ss^.rson;

While ss <> Nil do

Begin While ss^.lson <> Nil do { "O'yinchi" gacha chapga o'tish }

Begin Stekin; ss:= ss^.lson End;

If ss^.kl > ar2 Then Goto 2; { Ko'rib chiqishni tugallash ehtimoli }

Obrab(ss); ss:= ss^.rson

End

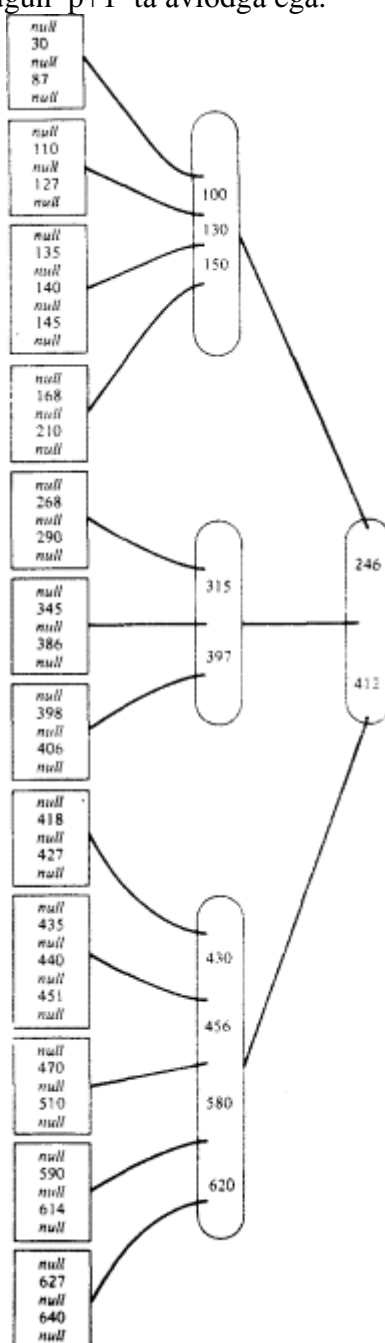
End; {BDU ni chapdan qisman ko'rib chiqish blokining oxiri}

2:While t <> Nil do Begin q:=t; t:=t^.ssl; Dispose(q) End;

End{Binar daraxtda izlash bloki oxiri(BDU)};

Binar daraxt B-daraxtning xususiy holi bo'lib hisoblanadi.m-darajali V- daraxt quyidagi shartlarni qanoatlantiruvchi umumiy daraxt sifatida aniqlanadi:

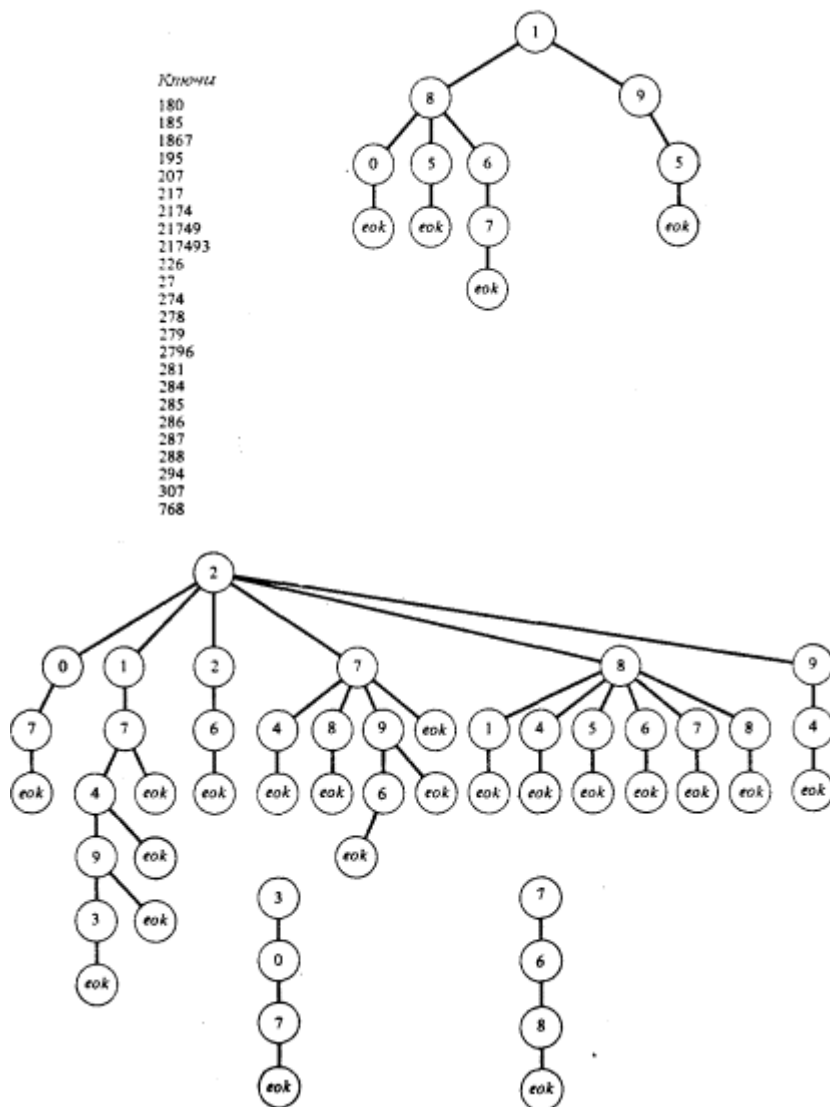
- 1.Har bir tugun maksimal m-1 ta kalit saqlaydi;
- 2.Bosh(ildiz) tugundan boshqa har bir tugun eng kamida $\text{int}((m-1)/2)$ ta kalit saqlaydi;
- 3.Ildiz tugun avlod bo'lmasa , eng kamida 2 ta avlod tugunga ega;
- 4.Barcha avlodlar bitta darajada joylashgan.
- 5.Avlod bo'lmagan n kalitli tugun p+1 ta avlodga ega.



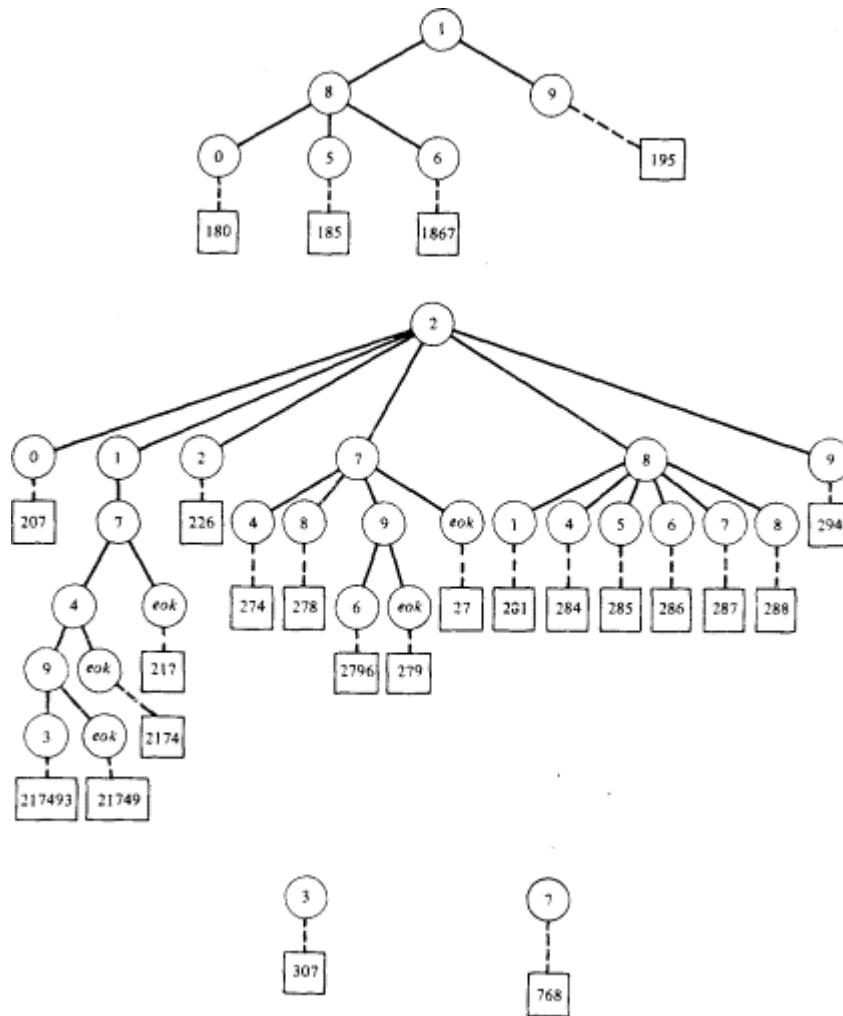
Ushbu rasmda 5-darajali V-daraxt ifodalangan.Bu erda har bir tugun qandaydir tartiblangan($p_1, k_1, p_2, k_2, \dots, k_{n-1}, r_n$) gurux orqali ifodalanishi mumkin.Bu erda p_i ko'rsatkich(bo'sh ko'rsatkich. Gar berilgan tugun avlod bo'lsa) ki esa qandaydir kalit. P_i ko'rsatayotgan tugundagi

kalitlar k_{i-1} va k_i orasida joylashadi. har bir tugun ichida esa $k_1 < k_2 < \dots < k_{n-1}$ tengsizlik bajariladi.

Raqamli izlash daraxtlari. Izlash jarayonini tezlashtirish uchun daraxtlardan foydalanishning boshqa usuli kalitlar tarkib torgan simvollariga asoslanadigan qandaydir umumiy daraxt shakllantirishdan iborat. Masalan, agar kalitlar sonli bo'lsa, har bir raqam pozitsiyasi berilgan tugunning 10 ta mumkin bo'lgan avlodlaridan birini aniqlaydi.



Daraxtning har bir tuguni maxsus eok simvoliga ega. Bu simvol qaysidir kalit oxirini bildiradi. Bunday tugun saqlab qolinuvchi yozuvni ko'rsatuvchi ko'rsatkichni ham o'zida saqlaydi.



Shtrixlangan chiziq daraxt tugunidan kalitga ko'rsatkichni ifodalaydi.

Nazorat savollari:

1. 1.Izlash algoritmlarining mohiyati nimada?
2. 2.Qanday izlash algoritmlari bor?
4. Massivda izlashning mohiyati nimada?
3. Daraxt usulbda izlashning mohiyati nimada?
4. Daraxt uslubida saralashning qanday turlari bor?

Foydalanilgan adabiyotlar:

1. <http://structur.h1.ru/hash.htm>
2. А.Р.Есауян и др. Информатика. М.:Просвещение,1991.212-224 с.

15-Mavzu . Arxivlash algoritmlari (2 soat)

Reja:

3. Seriyalarni ketma-ket kodlash algoritmi
4. Xaffman algoritmi
5. Lempel-Ziv algoritmi

Kalit so'zlar: Arxivlash,Kod, Ikkilik son, Seriya, Alfavit, Simvol

Axborotlarni siqish muammosi hozirgi kunda juda dolzarb bo'lib hisoblanadi. Buning boisi shundaki. Axborotlar oqimining tezlik bilan ortib borishi , ularni qayta ishlash, saqlash va uzatish tezligini ni oshirish muammosini keltirib chiqaradi. Bundan tashqari axborot texnologiyalarining turli hayotiy jabhalarda foydalanilish visitalarining rivojlanib borishi ushbu vositalar ishini ta'minlaydigan dasturiy mahsulotlar xajmining ortib borish tendensiyasini belgilaydi. Bunday sharoitda axborotlarni saqlash, xotira xajmini tejash muammolarining eng optimal echimi axborotlarni siqish usullari va ularga mos dasturiy vositalardan foydalanishdir.

E/M xotirasidagi axborotlarni siqilgan holda saqlash yuqoridagi muammolarni hal etishning birusulidir. Axborotlarni siqish(arxivlash) maxsus dasturlar yordamida amalga oshiriladi.. Ushbu dasturlar esa konkret siqish algoritmlari bo'yicha ishlaydi. Bugungi mavzu ana shunday bir necha siqish algoritmlarining mohiyati bilan tanishishga qaratilgan.

Siqish yoki arxivlvsh jarayonining maqsadi – boshlashg'ich(kiruvchi) axborot oqimini ma'lum usullar bilan kompakt chiquvchi(natijaviy) axborotlar bilan almashtirishdir.

Siqish jarayonlari quyidagi asosiy texnik xarakteristikalarga ega:

- siqilish darajasi(compress rating) yoki boshlang'ich va natijaviy axborot oqimlarining nisbati;
- siqilish tezligi;
- siqilish sifati;

Barcha siqilish usullari ikki katta kategoriyaga bo'linadi: tiklanuvchi va tiklanmas siqilish. Tiklanmas siqilishda berilganlarning boshlang'ich oqimi qayta ishlanishi natijasida tashqi xarakteristikalar bo'yicha boshlang'ich oqimga judu o'xshash, ammo informasion strukturasi bo'yicha yo'qotishlarga ega bo'lgan natijavi yoqim chiqariladi. Axborotlarni arxivlvshning bunday usuli rastrlı grafik fayllar, video va foto axborotlar, nutq va boshqa analogli signallrni raqamli ko'rinishda ifodalashda qo'llaniladi.

Tiklanuvchi siqilish deganda axborotlar xajmini informasion strukturani yo'qotishlarsiz qisqartirish tushuniladi. Ushbu siqilgan axborotlarni faqat dekompressiya(tiklash) qilingandan keyingina qayta ishlash mumkin. Dekompressiya natijasida axborotlar oldingi xajmlarni egallaydi.

Endi tiklanuvchi algoritmlarning asosiy nazariy prinsiplariga to'xtalib o'tamiz. Eng sodda va mashxur arxivlash usuli – **seriyalarni ketma-ket kodlash (RLE) algoritmidir**. Algoritm mohiyati takrorlanuvchi baytlarketma-ketliklari yoki zanjirlarini bitta kodlovchi bayt va ularning takrorlashlar soni hisobchisiga almashtirishdan iborat. Masalan,

44 44 44 11 11 11 11 11 01 33 AA 22 22 berilgan ketma-ketlik bo'lsin. Uni RLE algoritmi yordamida siqish natijasida quyidagi ketmaqketlikka ega bo'lamiz:

03 44 05 11 00 03 01 33 AA 02 22

Birinchi bayt takrorlanuvchi baytlar sonini, ikkinchi bayt takrorlanuvchi baytning o'zini bildiradi. 00 – baytidan keyin takrorlanmaydigan baytlar soni va takrorlanmaydigan baytlarning o'zi keladi.

Ushbu metod juda ko'p takrorlanuvchi baytlarga ega rastarli grafik tasvir fayllari (bmp, pcx, tif, gif) uchun juda effektiv bo'lib hisoblanadi. RLE usulining kamchiligi qisish darajasining pastligidadir.

Xaffman algoritmi – yana bir arxivlash usuli bo'li hisoblanadi. Axborotlarni Xaffman bo'yicha siqishda fayl butunligicha o'qiladi va undagi uchraydigan har bitta simvol uchun umumiy yig'indi miqdorlar hisoblanadi. Bunda 256ta simvolning hammasi hisobga olingan bo'lib, algoritm uchun bajariluvchi fayl bilan matnli fayl orasida hech qanday farq bo'lmaydi. hisoblash jarayoni natijalaridan foydalanib, binar daraxt tuziladi. Bu jarayonni konkret misol vositasida ko'rib chiqamiz:

Bizga 100 bayt xajmli o'zida 6 xil simvol saqlovchi fayl berilgan bo'lsin:

Символ	A	B	C	D	E	F
Число вхождений	10	20	30	5	25	10

Bu sonlarni simvollarning faylda qatnashish chastotalari deb ataymiz:

Символ	C	E	B	F	A	D
Число вхождений	30	25	20	10	10	5

Eng kichik ikkita tugundan yangi tugun hosil qilamiz:

Символ	C	A	D	F	B	E
Число вхождений	30	10	5	10	20	25

15

+ 5 + 10

Ushbu yangi tugunning faylda qatnashish chastotasi 15 ga teng. So'ngra yan ikkita eng kichik chastotali tugunlardan yangi tugun hosil qilamiz:

Символ	C	A	D	F	B	E
Число вхождений	30	10	5	10	20	25

15

25

Shu tarzda bu jarayonni bitta umumlashiruvchi tugun ga kelgunimizcha davom ettiriladi:

Символ	C	A	D	F	B	E
Число вхождений	30	10	5	10	20	25

15

25

55

45

Root (100)

Darxt shakllantirilgach, faylni kodlash mumkin bo'ladi. Kodlash jarayoni eng pastki tugundan boshlanadi. Daraxt bo'ylab pastdan .qoriga barcha burilishlarni hisobga olgan holda bajariladi. CHap tomonga burilish 0 bit, o'ng tomonga burilish 1 bit bilan kodlanadi. Demak S tugun uchun chapga 55(0bit), keyin yana chapga S simvoning o'zigacha(0bit). S simvol uchun Xaffman kodi – 00; A simvol uchun chapga, o'ngga, chapga, chapga. Natijada A ning kodiga

ega bo'lamiz: 0100; D uchun chapga, o'ngga,o'ngga, chapga, o'ngga – 0101; F uchun chapga, o'ngga, o'ngga – 011; V uchun o'ngga, chapga – 10,E uchun o'ngga, o'ngga – 11; Demak,
S=00(2 bit);
A=0100(4 bit);
F=011(3 bit);
V=10(2 bit);
E=11(2 bit).

S – 60 bit, A – 40 bit, F – 30 bit, V – 40 bit, E – 50 bit hammasi 220 bitni tashkil etadi. Bundan 100 baytli axborot 220 bitli kodga almashtiriladi. Kodlash jarayonida simvollar olingan ketma-ketliklarga almashtiriladi. Siqilish jarayonining ishlashi bitta simvol 8 ta emas, 2,3 va 4 bit joni egallashi evaziga bajariladi.

Lempel-Ziv algoritmi. Ushbu algoritmi ilk bor Abraxam Lempel va YAKob Ziv ishlarida tasvirlab berilgan. Bugungi kunda bu algoritmi LZ - algoritmi deb yuritiladi. Ushbu algoritmi modifikatsiyalari boshqa algoritmlarga nisbatan ancha keng tarqalgan. LZ algoritmi asosida fayllarda ko'p uchraydigan ketma-ketliklarni maxsus yaratiluvchi lug'atda saqlanuvchi namunalarga murojaatlar bilan almashtirish yotadi. Masalan, agar arxivlash dasturi lug'atda "ABV" ketma-ketlikka ega bo'lsa va "ABVA" ketma-ketlikka duch kelsa, chiqish fayliga "ABV" uchun lug'atdan kod yoziladi, keyin A uchun kod yoziladi, so'ngra "ABVA" ketma-ketlik lug'atga kiritiladi. Agar keyinroq "ABVAB" ketma-ketlik uchrasa, uning o'rniga "ABVA" uchun kod yoziladi, keyin "B" simvol uchun kod yoziladi hamda "ABVAB" yana lug'atga kiritiladi. Dastur lug'atda mavjud ketma-ketliklarni uchratsa, bu ketma-ketlik uchun kodni beradi va lug'atga bir bayt uzunroq yangi yozuvni kiritadi. Ushbu lug'atning xajmi turli dasturlarda turlicha bo'lishi mumkin. Masalan, Lharc dasturi 4 Kbaytli buferdan, LHA va PKZIP 8 Kilobaytli, ARJ dasturi esa 16 Kbaytli buferdan foydalanadi.

Berilganlarni lug'atdagi qism-satrlar bilan almashtirish jarayoni quyidagicha amalga oshiriladi:

Berilgan qism-satr bilan mos tushuvchi lug'atdagi qism-satrlardan eng uzuni topiladi va chiquvchi oqimga 2 ta satr uzatadi (lenght, distanse): lenght – lug'atda topilgan qism-satr uzunligi, distanse- lug'atdagi qism-satrdan kirish qism-satrigacha bo'lgan masofa. Agar bunday qism satr topilmasa. CHiqish oqimiga kirish oqimining navbatdagi simvoli qo'shiladi.

SHunday qilib, Lempel-Ziv algoritmi boshlang'ich berilganlarning simvollar ketma-ketligini 2 ta parallel uzunliklar va masofalar jadvaliga aylantirib beradi. Ushbu jadvalga yuqorida to'xtalib o'tilgan RLE yoki Xaffman algoritmlaridan birini qo'llash mumkin bo'ladi. SHu tarzda 2 bosqichli kodlash amalga oshiriladi. Ushbu metodni realizatsiyasida ikkala oqimning bitta faylga chiqarilishiga erishish kerak. Bu muammo ikkala oqim simvollarini oralatib yozish yo'li bilan hal etiladi.

Birinchi bosqich: absabsabsabsabs – 1a 1b 1s 33 63 93 123

Ikkinchi bosqich (takroralnuvchi ketma-ketliklarni qisqartirish): 1a 1b 1s 123. Bu ketma-ketlika esa Xaffman algoritmi qo'llaniladi.

Amalda axborotlarni siqish maxsus arxivlvsh dasturlari yordamida bajariladi. Bulardan ba'zilarini eltiramiz:

- PKPAK 3.61 RLE va LZW (Lempel-Ziv-Velch algoritmi) hamda Sqaushed-Xaffman algoritmlariga asoslangan holda ishlaydi;
- PKZIP 1.10 Shrunked metodi, modifikatsiyalangan LZW algoritmi, Imploded usuli, modifikatsiyalangan LZ algorit, Xaffman algoritmlariga asoslanib ishlaydi;
- Lharc LZ va Xaffman algoritmlariga asoslanib ishlaydi;
- Lha LZ va Xaffman algoritmlariga asoslanib ishlaydi;
- ARJ LZ algoritmi va original kodlash usulidan foydalanib ishlaydi;

Nazorat savollari:

1. Ma'lumotlarni arxivlashdan maqsad nima?
2. Arxivlash algoritmlarining mohiyati nimada?
3. Seriyalarni kodlash usulining mohiyati nimada?
4. Xaffman algoritmining mohiyati nimada?
5. Lempel-Ziv-Velsn algoritmining mohiyati nimada?

Foydalanilgan adabiyotlar:

1. Ватолин Д.,Ратушняк А.Методы сжатия данных.Устройство архиваторов,сжатие изображений и видео.Диалог-Мифи-2003,384с.
2. <http://structur.h1.ru/hash.htm>
3. <http://sorting2.shtml>
4. <http://www.izsiti.com/>