

**TOSHKENT
UNIVERSITETI**

AMALIY FANLAR

“MA'LUMOTLAR TUZILMASI VA ALGORITMLAR” FANI

“Kompyuter injiniring” kafedrası

Katta o'qituvchi Kendjayeva Dildora Xudayberganovna

1 - Ma'ruza

Kirish. Hisoblash modellari, algoritmlar va ularning murakkabligi.

Algoritm tushunchasini formallashtirish, Hisoblash modellari, Algoritmarning murakkabligi, Algoritmarning yomon, o'rta, yaxshi holatlari tushunchalar



Asosiy adabiyotlar:

- 1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.
- 2. O. R. Yusupov, I. Q. Ximmatov, E. Sh. Eshonqulov. Algoritmlar va berilganlar strukturalari. Oliy o'quv yurtlari uchun o'quv qo'llanma. – Samarqand: SamDU nashri. 2021-yil, 204 bet.
- 3. Xayitmatov O'T., Inogomjonov E.E., Sharipov B.A., Ruzmetova N., Ma'lumotlar tuzilmasi va algoritmlari fanidan o'quv qo'llanma
- 4. Rahimboboeva D. "Ma'lumotlar tuzilmasi va algoritmlari" fanidan o'quv qo'llanma – T.: TDIU, 2011.-135 bet.



MA'RUZA REJASI

Kirish.

Algoritm tushunchasini formallashtirish

Hisoblash modellari

Algoritmarning murakkabligi

Algoritmarning yomon, o'rta, yaxshi holatlari tushunchalar



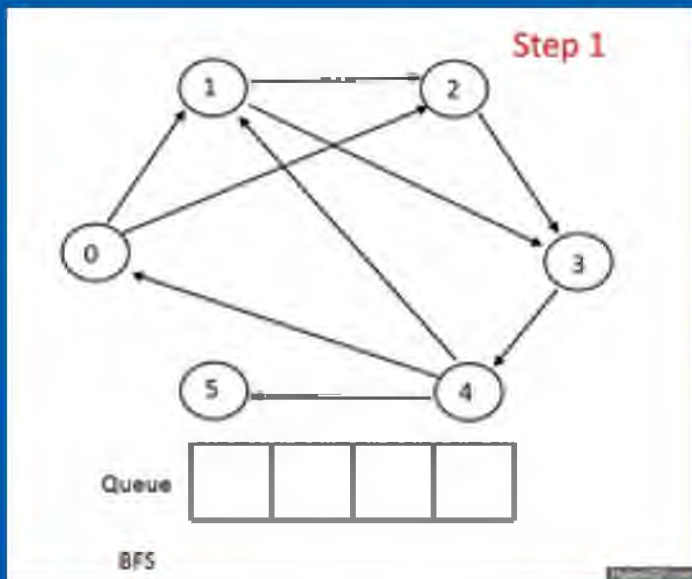
Kirish.

- “Ma'lumotlar tuzilmasi va algoritmlar” fanini o'zlashtirishning **maqsadi** dasturlashda ishlatiladigan ma'lumotlar tuzilmalarini, ularning spetsifikatsiyasi va amalga oshirilishini, ma'lumotlarni qayta ishlash algoritmlarini va ushbu algoritmlarni tahlil qilishni, algoritmlar va ma'lumotlar tuzilmalarining o'zaro bog'liqligini o'rganishdir.
- “Ma'lumotlar tuzilmasi va algoritmlar” fani har qanday dasturiy ta'minot tizimining **asosidir**: taqsimlangan tizimlar, mobil ilovalar, ma'lumotlar bazasi, veb-ilovalar.





Kirish.



- Fanning **vazifalariga** algoritmlar va ma'lumotlar tuzilmalari rivojlanishiga asos bo'lgan asosiy nazariy tushunchalarni shakllantirish, ma'lumotlarning abstrakt turi (MAT) modeli (paradigmasi) yordamida murakkab (dinamik) ma'lumotlar tuzilmalarini qurish va ulardan foydalanish.
- Algoritmlar va dasturlarning murakkabligini tahlil qilish to'g'risida g'oyalar va bilimlarni shakllantirishdan iborat.



Kirish.



- **Algoritm tushunchasi.** Avvalo algoritm tushunchasi IX asrda yashab irod etgan buyuk bobokalonimiz Muhammad al-Xorazmiy nomi bilan uzviy bog'liqligini eslatib o'tish lozim.
- **Algoritm so'zi** al- Xorazmiyning arifmetikaga bag'ishlangan asarining dastlabki betidagi **"Dixit Algoritmi"** ("dediki al-Xorazmiy" ning lotincha ifodasi) degan jumalardan kelib chiqqan. Shundan so'ng al-Xorazmiyning sanoq sistemasini takomillashtirishga qo'shgan hissasi, uning asarlari algoritm tushunchasining kiritilishiga sabab bo'lganligi ta'kidlab o'tiladi.



Kirish.

Algoritmlar nazariyasida hal qilingan maqsad va vazifalar:

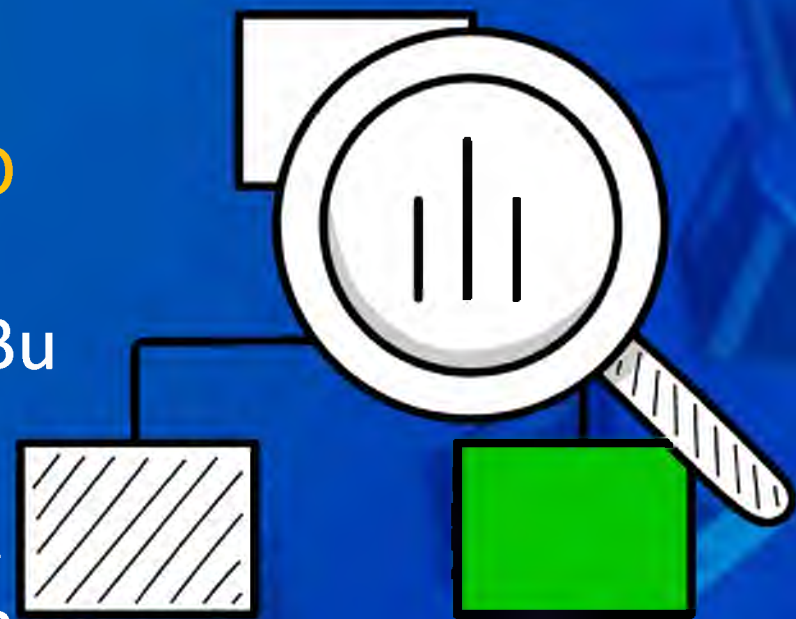
- "algoritm" tushunchasini formallashtirish (rasmiylashtirish) va formal (rasmiy) algoritmik tizimlarni o'rganish;
- muammolarning algoritmik yechimini rasmiy tasdiqlash;
- vazifalarni tasniflash, murakkablik sinflarini aniqlash va tadbiq qilish;
- algoritmarning murakkabligini asimptotik tahlil qilish;
- rekursiv algoritmlarni o'rganish va tahlil qilish;
- algoritmlar sifatini qiyosiy baholash mezonlarini ishlab chiqish.





Algoritm tushunchasini formallashtirish

- **1- ta'rif.** Algoritm - bu ma'lum bir tilda berilgan, mumkin bo'lgan dastlabki ma'lumotlar sinfi uchun masalani hal qilish uchun mumkin bo'lgan elementar amallarning cheklangan ketma-ketligi.
- Masalaning dastlabki ma'lumotlarining to'plami **D** bo'lsin va **R** - mumkin bo'lgan natijalar to'plami, shunda algoritm $D \rightarrow R$ ko'rinishida tasvirlanadi. Bu tasvirlanish to'liq bo'lmasligi mumkin.
- Agar natija faqat **ba'zi** $d \in D$ uchun olingan bo'lsa, algoritm **qismaniy** algoritm va agar **barcha** $d \in D$ uchun to'g'ri natija olsa **to'liq** algoritm deyiladi.





Algoritm tushunchasini formallashtirish

- 2- ta'rif. Algoritm - bu cheklangan vaqt ichida masalani yechish natijasiga erishish uchun ijrochining harakatlari tartibini tavsiflovchi aniq ko'rsatmalar to'plami.

Algoritmning aniq yoki bilvosita turli xil ta'riflari bir qator talablarni keltirib chiqaradi:

- algoritmda cheklangan miqdordagi elementar bajarilishi mumkin bo'lgan ketma-ketlik bo'lishi kerak, ya'ni yozuvning aniqligi talabi bajarilishi kerak;
- algoritm masalani yechishda cheklangan sonli bosqichlarni bajarishi kerak, ya'ni harakatlarning aniqligi talabi bajarilishi kerak;
- barcha qabul qilingan kirish ma'lumotlari uchun algoritm bir xil bo'lishi kerak, ya'ni universallik talabiga javob berish;
- algoritm qo'yilgan vazifaga nisbatan to'g'ri yechimga olib kelishi kerak, ya'ni to'g'rilik talabi bajarilishi kerak.





Algoritm tushunchasini formallashtirish

Algoritmning asosiy xossalari haqida quyidagilarni ta'kidlash mumkin:

Diskretlilik

- ya'ni algoritmni chekli sondagi oddiy ko'rsatmalar ketma-ketligi shaklida ifodalash mumkin.

Tushunarlilik

- ya'ni ijrochiga tavsiya etilayotgan ko'rsatmalar uning uchun tushunarli bo'lishi shart, aks holda ijrochi oddiy amalni ham bajara olmay qolishi mumkin. Har bir ijrochining bajara olishi mumkin bo'lgan ko'rsatmalar tizimi mavjud.

Aniqlik

- ya'ni ijrochiga berilayotgan ko'rsatmalar aniq mazmunda bo'lishi lozim hamda faqat algoritmda ko'rsatilgan tartibda bajarilishi shart.

Ommaviylik

- ya'ni har bir algoritm mazmuniga ko'ra bir turdagi masalalarning barchasi uchun yaroqli bo'lishi lozim. **Masalan**, ikki oddiy kasr umumiy maxrajini topish algoritmi har qanday kasrlar umumiy maxrajini topish uchun ishlatiladi.

Natijaviylik

- ya'ni har bir algoritm chekli sondagi qadamlardan so'ng albatta natija berishi lozim.



Algoritm tushunchasini formallashtirish

- Algoritmning tasvirlash usullari haqida gapirganda algoritmning berilish usullari xilma-xilligi va ular orasida eng ko'p uchraydiganlari quyidagilar ekanligini ko'rsatib o'tish joiz:

Algoritmning so'zlar orqali ifodalanishi

Algoritmning formulalar yordamida berilishi.

Algoritmning jadval ko'rinishida berilishi

- **masalan**, turli matematik jadvallar, lotereya yutuqlari jadvali, funksiyalar qiymatlari jadvallari bunga misol bo'ladi.

Algoritmning dastur shaklida ifodalanishi

- ya'ni algoritm kompyuter ijrochisiga tushunarli bo'lgan dastur shaklida beriladi.

Algoritmning algoritmik tilda tasvirlanishi

- ya'ni algoritm bir xil va aniq ifodalash, bajarish uchun qo'llanadigan belgilash va qoidalar majmui algoritmik til orqali ifodalashdir.


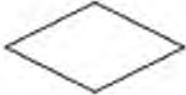



Algoritmlarning grafik shaklda tasvirlanishi.






- **Masalan**, grafiklar, sxemalar ya'ni blok - sxema bunga misol bo'la oladi.



Algoritm tushunchasini formallashtirish

Blok sxemaning asosiy elementlari quyidagilar:

No	SHakllar nomi	SHakllar	Mazmuni
1.	Jarayon		Xisoblash va xisoblash ketma-ketlik jarayoni
2.	Echim		SHartni tekshirish
3.	Modifikatsiya		Sikl boshi
4.	CHeklangan jarayon		Qism dasturini hisoblash
5.	Hujjatlar		CHiqarish, natijani qog'ozga chop etish

6.	Kiritish-chiqarish		Qiymatlarni qayta ishlash uchun kiritish va qayta ishlash natijalarini chiqarish
7.	Boshlanish-tamom		Boshlanish, tamom, qiymatlarni qayta ishlash yoki dasturni bajarilish jarayonini uzilishi
8.	Birlashtirish		SHakllarni birlashtirish ko'rsatkichi
9.	Displey, ekran		Qiymatlarni ekranga chiqarish
10.	Qo'lda kiritish		Qiymatlarni klaviatura orqali kiritish



Hisoblash modellari

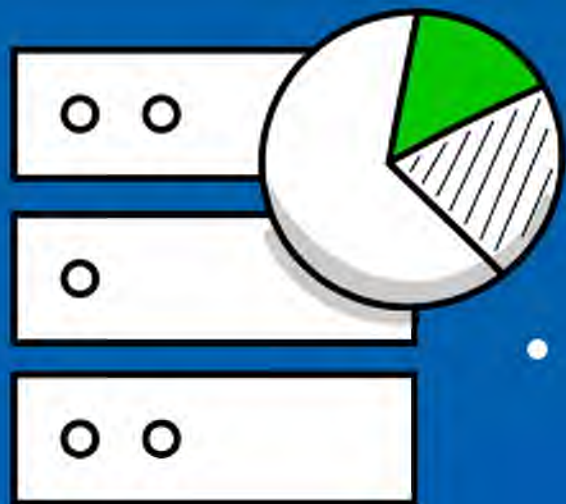
- **Hisoblash nazariyasi** va **hisoblash murakkabligi nazariyasi** hisoblash modelini nafaqat hisoblash uchun foydalaniladigan qabul qilinadigan amallar to'plamining ta'rifi, balki ularni qo'llashning nisbiy xarajatlari sifatida ham ko'rib chiqadi.
- Kerakli **hisoblash manbalarini** - ijro etish vaqtini, xotira hajmini, shuningdek algoritmlarning cheklanishlarini yoki kompyuterni xarakterlash mumkin - faqat ma'lum bir hisoblash modeli tanlangan taqdirda.





Hisoblash modellari

- Modelga asoslangan muhandislikda **hisoblash modeli** va uning tanlovi, agar uning alohida qismlarining xatti-harakatlari ma'lum bo'lsa, umuman tizim qanday ishlaydi degan savolga javob beradi.



Hisoblash murakkabligining **asimptotik bahosida** hisoblash modeli ma'lum narx bilan qabul qilinadigan primitiv amallar orqali aniqlanadi.

- Ma'lum amallar to'plamiga va ularning **hisoblash murakkabligiga** qarab bir qator hisoblash modellari ma'lum. Ular quyidagi keng toifalarga bo'linadi: algoritm hisoblashning murakkabligini yuqori chegarasini olish uchun foydalaniladigan **abstrakt modellar** va algoritmik masalalar uchun hisoblash murakkabligining pastki chegarasini olish uchun **ishlatiladigan qaror modellari**.



Hisoblash modellari

Tyuring tezisi. Chyorch tezisi. Algoritm tushunchasini aniqlashga yondashishlar. Algoritm tushunchasini aniqlash bo'yicha yondashishlarni **uch asosiy yo'nalishga** bo'lish mumkin.

- **Birinchi yo'nalish** effektiv **hisoblanuvchi funksiya** tushunchasini aniqlash bilan bog'liq. Bu yo'nalish bo'yicha A.Chyorch, K.Gyodel, S.Klini tadqiqot ishlarini olib borishgan.





Hisoblash modellari



- **Ikkinchi yoʻnalish** algoritm tushunchasini bevosita aniqlash bilan bogʻliq: 1936-1937-yillarda, A.Tyuring Chyorch ishlaridan bexabar holda yangi funksiyalar sinfini kiritdi. Bu funksiyalarni — **"Tyuring boʻyicha hisoblanuvchi funksiyalar"** deb atadilar.
- **Uchinchi yoʻnalish** - Rossiya matematigi A.Markov tomonidan ishlab chiqilgan **normal algoritmlar** tushunchasi bilan bogʻliq.



Algoritmlarning murakkabligi

- Algoritmlarni baholash uchun ko'pgina **mezonlar** mavjud. Odatda kirituvchi berilganlarni ko'payishida masalani yechish uchun kerak bo'ladigan **vaqt va xotira** hajmlarini o'sish tartibini aniqlash muammosi qo'yiladi. Har bir aniq masala bilan kiritiladigan berilganlarni miqdorini aniqlovchi qandaydir sonni bog'lash zarur. Bunday son **masalaning kattaligi** deb qabul qilinadi.
- Masalan, ikkita matritsani ko'paytirish masalasining o'lchami bo'lib, matritsalar kattaligiga xizmat qilishi mumkin.
- Algoritm sarflanayotgan vaqt masalaning o'lchami funksiyasi sifatida algoritmni **vaqt bo'yicha qiyinligi** deb nomlanadi. Bunday funksiyaga masalaning kattaligi oshganda limit ostidagi o'zgarish **asimptotik qiyinlik** deb aytiladi.



Algoritmlarning murakkabligi

- **Murakkablikni baholash.** Algoritmlarning murakkabligi odatda bajarilish **vaqti** yoki ishlatilgan **xotira** bo'yicha baholanadi. Ikkala holatda ham, murakkablik kiritilgan ma'lumotlarning **hajmiga** bog'liq:
- 100 ta elementdan iborat massivi xuddi shunga o'xshash 1000 ta elementdan iborat massivga qaraganda **tezroq** qayta ishlanadi.
- Shu bilan birga, aniq vaqt bilan bir necha kishi qiziqadi: bu protsessorga bog'liq, ma'lumotlar turi, dasturlash tili va boshqa ko'plab parametrlarga ham bog'liq.
- Faqatgina **asimptotik murakkablik** muhim, ya'ni kirish ma'lumotlarining kattaligi cheksizlikka intilayotgandagi murakkablik.



Algoritmlarning murakkabligi

- **Masalan**, ba'zi bir algoritmgaga kirish ma'lumotlarining n ta elementlarini qayta ishlash uchun $4n^3 + 7n$ ta shartli amallarni bajarish kerak. n ning ortishi bilan ishning umumiy davomiyligi n ning kubi uni 4 ga ko'paytirgandan yoki $7n$ ni qo'shgandan ko'ra ko'proq ta'sir qiladi.
- Ushbu algoritmnining vaqt murakkabligi $O(n^3)$, ya'ni u kubik bilan kiritilgan ma'lumotlarning hajmiga bog'liq bo'ladi.





Algoritmlarning murakkabligi

- Bosh harf **O** dan foydalanish matematikadan kelib chiqadi, bu yerda ushbu belgi funksiyalarning **asimptotik harakatlarini** taqqoslash uchun ishlatiladi.
- Rasmiy ravishda **$O(f(n))$** algoritmning ishlash vaqti (yoki egallagan xotira miqdori), kiritilgan ma'lumotlarning hajmiga qarab, **$f(n)$** ga ko'paytiriladigan ba'zi konstantalardan tezroq emasligini anglatadi.



Algoritmlarning murakkabligi

- **$O(n)$ - chiziqli murakkablik.** Bunday murakkablik, masalan, saralanmagan massivdagi eng katta elementni topish algoritmiga ega. Qaysi biri maksimal ekanligini aniqlash uchun massivning barcha n elementlaridan o'tishimiz kerak bo'ladi.
- **$O(\log n)$ - logaritmik murakkablik.** Eng oddiy misol - ikkilik qidirish. Agar massiv saralangan bo'lsa, uni yarmiga bo'lish orqali ma'lum bir qiymatga ega ekanligini tekshirishimiz mumkin. O'rta elementni tekshirib ko'ramiz, agar u kattaroq bo'lsa, unda massivning ikkinchi yarmini tashlab yuboramiz. Agar kichikroq bo'lsa, unda aksincha - biz dastlabki yarmini tashlaymiz va shu tarzda ikkiga bo'linishni davom ettiramiz, natijada $(\log n)$ elementlarini tekshiramiz.
- **$O(n^2)$ - kvadratik murakkablik.** Bunday murakkablik, masalan, element qo'shilishi natijasida yangi saralash algoritmiga ega. Kanonik dasturda bu ikkita ichki sikldan iborat: biri butun massivni bosib o'tish, ikkinchisi esa allaqachon ajratilgan qismdan keyingi element uchun joy topish. Shunday qilib, amallar soni $n*n$, ya'ni n^2 kabi massiv o'lchamiga bog'liq bo'ladi.
- Murakkablikning boshqa ko'rinishlari ham mavjud, ammo ularning barchasi bir xil prinsipga asoslanadi.



Algoritmlarning murakkabligi

- Algoritmning ishlash **vaqti** umuman kiritilgan ma'lumotlarning **hajmiga** bog'liq emasligi ham sodir bo'ladi. Bu holda murakkablik **$O(1)$** bilan belgilanadi.
- Masalan, massivning uchinchi elementi qiymatini aniqlash uchun elementlarni eslab qolishingiz yoki ular orqali bir necha bor o'tishingiz shart emas. Siz har doim ma'lumotlarni kiritish oqimidagi uchinchi elementni kutishingiz kerak va bu esa siz uchun natija bo'ladi, bu har qanday ma'lumot uchun hisoblash uchun bir xil vaqtni oladi.



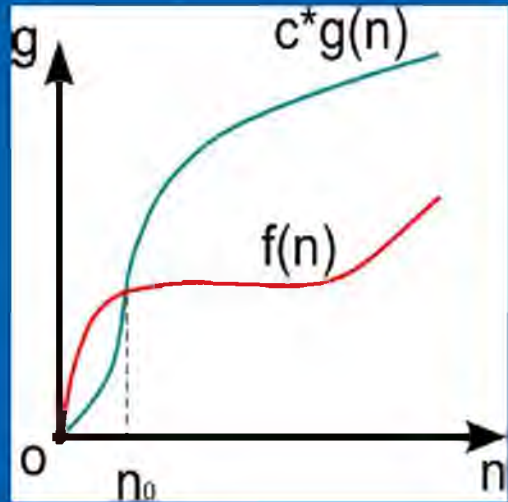


Algoritmlarning murakkabligi

- Murakkablikning o'sish tartibi (yoki **aksiomatik murakkablik**) katta kirish hajmi uchun algoritmning murakkablik funksiyasining taxminiy xatti-harakatini tavsiflaydi.
- Bundan kelib chiqadiki, vaqt murakkabligini baholashda elementar amallarni ko'rib chiqishning hojati yo'q, **algoritm qadamlarini** ko'rib chiqish kifoya.
- **Algoritm qadami** - bu ketma-ket joylashtirilgan elementar amallar to'plami, uning bajarilish vaqti kirish qadamiga bog'liq emas, ya'ni yuqoridan qandaydir doimiy bilan chegaralangan.



Algoritmlarning murakkabligi



- **Asimptotik baholashning ko'rinishlari.** $F(n) > 0$ murakkabligini, bir xil tartibdagi $g(n) > 0$ funksiyasini, kirish $n > 0$ o'lchamini ko'rib chiqaylik.
- Agar $f(n) = O(g(n))$ va $n > n_0$ uchun $c > 0$, $n_0 > 0$ konstantalar mavjud bo'lsa, u holda $0 < f(n) < c * g(n)$.
- Bu holda $g(n)$ funksiyasi $f(n)$ uchun **asimptotik-aniiq baho** hisoblanadi. Agar $f(n)$ algoritmnining murakkablik funksiyasi bo'lsa, unda murakkablik tartibi $f(n)$ uchun - $O(g(n))$ deb belgilanadi. Ushbu ibora doimiy koeffitsiyentgacha $g(n)$ dan **tez o'smaydigan** funksiyalar sinfini belgilaydi.



Algoritmlarning murakkabligi

- Asimptotik funksiyalarga misollar

$f(n)$	$g(n)$
$2n^2 + 7n - 3$	n^2
$98n * \ln(n)$	$n * \ln(n)$
$5n + 2$	n
8	1



Algoritmlarning yomon, oʻrta, yaxshi holatlari tushunchalar

- Algoritm murakkabligining oʻsish tartibi $O(n)$ deb aytganda nimani nazarda tutamiz? Bu oʻrtacha? Yoki eng yomoni? Ehtimol, eng yaxshisi?
- Agar eng yomon holat va oʻrtacha koʻrsatkichlar bir-biridan farq qilmasa, odatda, eng yomon holat nazarda tutiladi.
- Masalan, biz oʻrtacha $O(1)$ oʻsadigan, lekin vaqti-vaqti bilan $O(n)$ ga aylanadigan algoritmlarning misollarini koʻrib chiqamiz (masalan, massivga element qoʻshish). Bunday holda, algoritm oʻrtacha vaqt ichida doimiy ishlashini koʻrsatamiz va **murakkablik oshadigan holatlarni** tushuntiramiz.



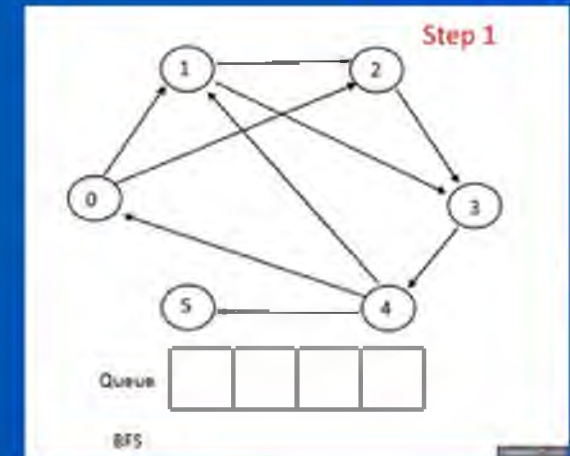
Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- Algoritmlar va ma'lumotlar tuzilmalarining **murakkabligini** o'lchashda odatda ikkita narsa haqida gaplashamiz:
- ishni bajarish uchun zarur bo'lgan **amallar soni** (hisoblash murakkabligi) va algoritm zarur bo'lgan **resurslar**, xususan, xotira (fazoviy murakkablik).
- O'n baravar tezroq ishlaydigan, ammo o'n barobar ko'proq joy ishlatadigan algoritm ko'proq xotirali server mashinasi uchun yaxshi bo'lishi mumkin. Ammo xotira hajmi chekli o'rnatilgan tizimlarda ushbu algoritmdan foydalanib bo'lmaydi.



Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- Odatda, quyidagi amallar hisobga olinadi:
- 1) taqqoslashlar ("katta", "kichik", "teng");
- 2) o'zlashtirish (ta'minlash);
- 3) xotira ajratish.
- Qaysi amalni hisoblash esa, odatda kontekstda aniqlanadi.
- Masalan, ma'lumotlar tarkibidagi **elementni topish** algoritmini tavsiflashda biz deyarli taqqoslash amallarini nazarda tutamiz. **Qidirish**, avvalambor, o'qish jarayonidir, shuning uchun ta'minlash yoki xotira ajratishda hech qanday ma'no yo'q.





Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- **Tartiblash** haqida gapirganda esa, **taqqoslash**, **xotira ajratish va ta'minlash** amallarini hisobga olishimiz mumkin. Bunday hollarda biz qaysi amallarni ko'rib chiqayotganimizni aniq ko'rsatib beramiz.
- Algoritmni har xil hajm va miqdorlarning boshlang'ich qiymatlari bilan qanday ishlashini, qanday manbalarga ehtiyoj borligini va yakuniy natijani chiqarish uchun qancha vaqt ketishini **baholash** ham muhimdir.





Algoritmlarning yomon, oʻrta, yaxshi holatlari tushunchalar

- Algoritm murakkabligining asosiy koʻrsatkichi bu muammoni hal qilish uchun sarflanadigan **vaqt** va kerakli **xotira hajmi**.
- Shuningdek, muammolar sinfi uchun murakkablikni tahlil qilishda maʼlum bir maʼlumot miqdori - **kirish kattaligini** tavsiflovchi maʼlum bir raqam aniqlanadi. Shunday qilib, biz algoritmning murakkabligi kirish oʻlchamining funksiyasi degan xulosaga kelishimiz mumkin.
- **Yomon, oʻrtacha** yoki **eng yaxshi** darajadagi murakkablik tushunchalari mavjud. Odatda, eng yomon holatning murakkabligi baholanadi.





Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- Eng yomon holatda **vaqt murakkabligi** - bu berilgan kattalikdagi masalani yechishda algoritm ishlashi davomida bajariladigan amallarning maksimal soniga teng bo'lgan kirish kattaligining funksiyasidir.
- Eng yomon **sig'imli murakkablik** - bu kirish hajmining ma'lum hajmdagi muammolarni yechishda erishilgan maksimal xotira yacheykalari soniga teng funksiyasi.





Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- **Algoritm murakkabligini baholash kriteriyalari.** Bir xil me'yorda **o'lchash kriteriyasi** algoritmning har bir bosqichi bir vaqt birligida, xotira yacheykasi esa hajmning bir birligida (konstanta bo'yicha aniqlikda) bajarilishini nazarda tutadi.
- **Logarifmik o'lchash kriteriyasi** ma'lum amal bilan qayta ishlanadigan operand o'lchamini va xotira yacheykasida saqlanadigan qiymatni hisobga oladi.

$$l(i) = \begin{cases} \lceil \log_2(|i|) \rceil + 1, & i \neq 0 \\ 1, & i = 0 \end{cases}$$



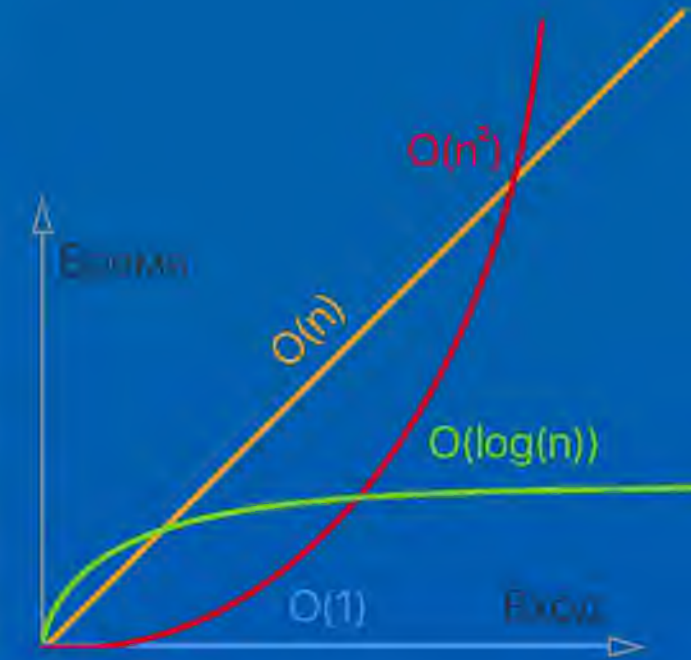
Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

```
#include <iostream>
using namespace std;
int main()
{
    int n = 20;
    long long result=1;
    for (int i=2; i<=n; i++)
        result *=i;
    cout<<result;
    return 0;
}
```

- **1-misol.** Faktorialni hisoblashning murakkabligini baholash.
- Berilgan masalaning kirish kattaligi **n** ekanligini aniqlash oson. Qadamlar soni **(n - 1)**.
- Shunday qilib, **bir xil me'yorda** o'lchash kriteriyasi uchun vaqt murakkabligi $O(n)$ dir.



Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar



- **Logarifmik o'lchash kriteriyasi** bilan vaqt murakkabligi. Shu nuqtada, baholanishi kerak bo'lgan amallarni ajratib ko'rsatish kerak.
- Birinchidan, bu **taqqoslash amallari**.
- Ikkinchidan, o'zgaruvchi qiymatiga **ta'sir qiluvchi amallar** (qo'shish, ko'paytirish, bo'lish, ayirish).
- **O'zlashtirish (ta'minlash)** amali hisobga olinmaydi, chunki ular bir zumda amalga oshiriladi deb taxmin qilinadi.



Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- Shunday qilib, ushbu masala uchta amal ajratilgan:

1) $i \leq n$ i-qadamda $\log(n)$ ni olamiz. Qadamlar soni $n - 1$ ta bo'lgani uchun ushbu amalning murakkabligi $(n - 1) * \log(n)$ ga tengdir.

2) $i++$; i-qadamda $\log(n)$ ni olamiz.

Ushbu holatda, quyidagicha yig'indi hosil bo'ladi:

```
#include <iostream>
using namespace std;
int main()
{
    int n = 20;
    long long result=1;
    for (int i=2; i<=n; i++)
        result *=i;
    cout<<result;
    return 0;
}
```

$$\sum_{i=2}^n \log_2 i = \log_2(i!)$$



Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

```
#include <iostream>
using namespace std;
int main()
{
    int n = 20;
    long long result=1;
    for (int i=2; i<=n; i++)
        result *=i;
    cout<<result;
    return 0;
}
```

- 3) result *=i; i-qadamda $\log((i-1)!)$ ni olamiz.
- Ushbu holatda, quyidagi yig'indi hosil bo'ladi:

$$\sum_{i=2}^n \log_2((i-1)!) = \log_2 \left(\prod_{i=2}^n (i-1)! \right)$$

- Agar biz barcha olingan qiymatlarni yig'sak va n ning o'sishi bilan asta sekin o'sadigan atamalarni bekor qilsak,

$$O\left(\log_2 \left(\prod_{i=2}^n (i-1)! \right)\right)$$

- biz yakuniy ifodasini olamiz.



Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- Bir xil me'yorda o'lchash kriteriyasi bo'yicha sig'imning murakkabligi.
- Bu yerda hamma narsa oddiy. O'zgaruvchilar sonini hisoblashingiz kerak. Agar topshiriq massivlardan foydalansa, massivdagi har bir yacheyka o'zgaruvchi hisoblanadi. O'zgaruvchilar soni kirish kattaligiga bog'liq bo'lmaganligi sababli, murakkablik $O(1)$ bo'ladi.
- Logarifmik o'lchash kriteriyasi ega bo'lgan sig'imning murakkabligi.
- Bunday holda, siz xotira yacheykasida bo'lishi mumkin bo'lgan maksimal qiymatni hisobga olishingiz kerak. Agar qiymat aniqlanmagan bo'lsa (masalan, $i > 10$ operand bo'lganda), u holda V_{\max} chegaraviy qiymati bor deb hisoblanadi.
- Ushbu masalada qiymati $n(i)$ dan oshmaydigan va $n(\text{result})$ qiymatidan oshmaydigan o'zgaruvchi mavjud. Shunday qilib, $O(\log(n!))$ ga teng.



Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- **2-misol.** Massiv elementlari yig'indisi. Bir o'lchovli massivning barcha elementlari qiymatlari yig'indisini hisoblaydigan quyidagi algoritm bor deylik:

- Birinchi yondashuvda biz quyidagi taxminlarni olamiz. Birinchi ifoda bir marta bajariladi va u kiritilgan ma'lumotlarning o'lchamiga bog'liq emas. Ikkinchi operatorning bajarilish soni kiritilgan ma'lumotlarning o'lchamiga bog'liq (xususan, n - massivning uzunligiga). Ushbu holatda bu $n + 1$ (for siklining boshi uning tanasidan bir marta ko'proq bajarilishini unutmang).

Shunga ko'ra uchinchi operator n mavhum vaqt birligida bajariladi. Shunday qilib, bizda:

- Barcha operatorlarning algoritmlar murakkabligi yig'indisini hisoblash natijasida $2n + 2$ murakkablikni olish mumkin.

```
(1)  S=0;  
(2)  for(int i=0; i<n; i++)  
(3)  S=S+A[i];
```

```
S=0;                (1)  
for(int i=0; i<n; i++) (n+1)  
S=S+A[i];           (n)
```



Algoritmlarning yomon, o'rta, yaxshi holatlari tushunchalar

- Algoritmlarni baholashda ko'pincha quyidagi funksiyalar qo'llaniladi: $\log_2 n$, n , $n \cdot \log_2 n$, n^2 , n^3 , $2n$, $10n$, $n!$. $O(\log n)$ ko'rinishida baholangan algoritmlar, har qanday sababga ko'ra, juda **tez algoritmlar** deb nomlanadi. $O(n)$ va $O(n \log n)$ deb baholangan algoritmlar **tezkor algoritmlar** deb ataladi.
- $O(n^2)$, $O(n^3)$ yoki umumiy holatda $O(n^c)$ bo'lgan algoritmlar **polynomial algoritmlar** deyiladi, $O(2n)$, $O(10n)$, $O(n!)$ baholangan algoritmlar esa **polynomial bo'lmagan algoritmlar** deyiladi.



**Do you have
any questions?**

