I BOB Ma’lumotlarning abstrak turlari va chiziqli ma’lumotlar tuzilmalari

1. Ma’lumotlar tuzilmalari va algoritmlar: asosiy tushinchalar va ta’riflar.

Ma’lumotlar tuzilishi va algoritmlar (MTvA)” ikkita alohida, ammo o‘zaro bog’liq mavzular kombinatsiyasi sifatida qaraladi. MTvA har bir dasturchi bilish kerak bo‘lgan eng muhim ko‘nikmalardan biridir. Ko‘pincha ushbu texnologiyalarni yaxshi biladiganlar boshqalarga qaraganda yaxshiroq dasturchilar hisoblanadi va shuning uchun deyarli har biri gigant texnologiya kompaniya intervyularidan yaxshi natijaga erishadilar.

1.1 Ma’lumotlarning tuzilishi

Ma’lumotlar tuzilishi ma’lumotlarni samarali ishlatish uchun qurilmalarimizda saqlash va tartibga solishning maxsus usuli sifatida qaraladi. Ma’lumotlar tuzilmalaridan foydalanishning asosiy g’oyasi vaqt va xotira murakkabliklarni minimallashtirishdir. Samarali ma’lumotlar tuzilishi xotirada minimal joy egallaydi va ma’lumotlarni ustida amallar bajarish uchun minimal vaqt talab etiladi.

Ma’lumotlar tuzilishi va algoritmlar o‘rganishni qanday boshlash kerak?

Birinchi va asosiy narsa-bu umumiy protsedurani ketma-ket bajarish kerak bo‘lgan kichik qismlarga bo‘lish. Ma’lumotlar tuzilishi va algoritmlar o‘rganishning to‘liq jarayoni 4 qismga bo‘linishi mumkin:

- Vaqt va xotira murakkabliklari haqida bilib olish
- Ma’lumotlar tuzilmalari asoslarini bilib olish
- Algoritmlar asoslarini bilib olish
- MTvA bo‘yicha amaliy masalalarni yechish.

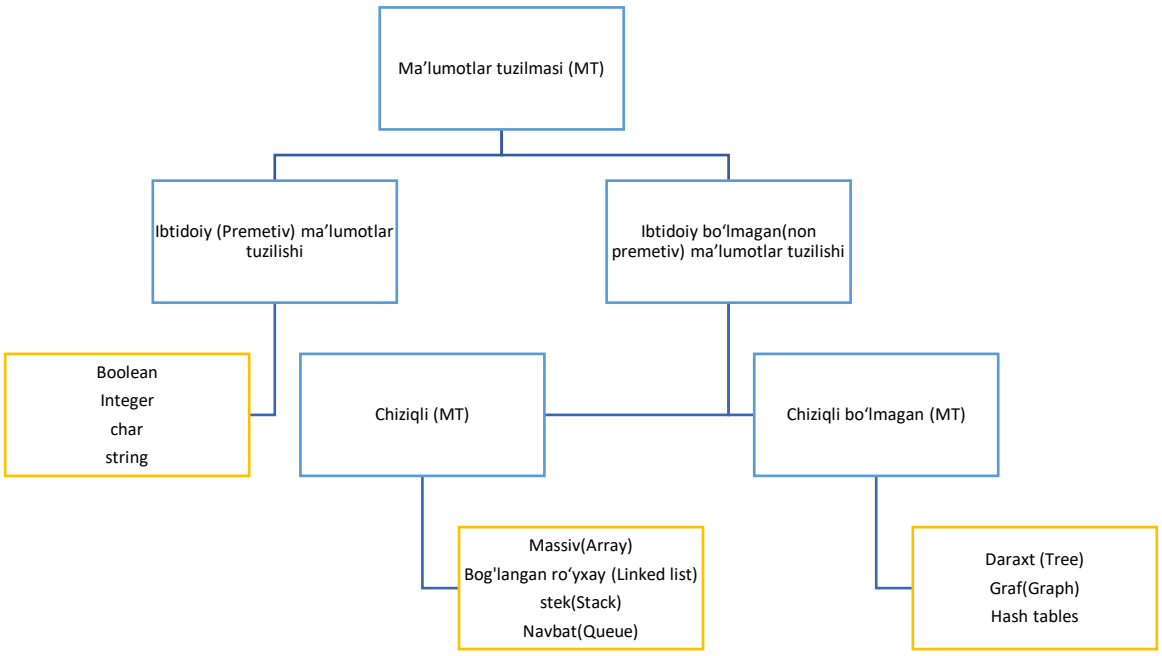
Ma’lumotlar tuzilishi- vazifasi bu kompyuterda kiritilgan ma’lumotlarni samarali foydalanish uchun ularni tartibga solish va saqlash hisoblanadi. Bu kompyuter

dasturida ma’lumotlarning mantiqiy yoki matematik ko‘rinishini amalga oshirilishini anglatadi.

Ma’lumotlar tuzilmalarini ikkita keng toifaga bo‘lish mumkin:

Ma’lumotlarning chiziqli tuzilishi: ma’lumotlar elementlari ketma-ket yoki chiziqli ravishda joylashtirilgan, bu erda har bir element avvalgi va keyingi qo‘shni elementlarga biriktirilgan ma’lumotlar tuzilishi chiziqli ma’lumotlar tuzilishi deb ataladi. Masalan: Massiv, bog’langan ro‘yxat, Stek va navbat.

Chiziqli bo‘lmagan ma’lumotlar tuzilishi: ma’lumotlar elementlari ketma-ket yoki chiziqli joylashtirilmagan ma’lumotlar tuzilmalari deyiladi. Chiziqli bo‘lmagan ma’lumotlar tuzilmalariga graflar va daraxtlarni misol keltirsak bo‘ladi.



Rasm- 1Ma’lumotlar tuzilmasining toifalari

Ma’lumotlar tuzilmalari keng ko‘lamli kompyuter dasturlari va ilovalarida qo‘llaniladi, jumladan:

- *Ma’lumotlar bazalari:* ma’lumotlar tuzilmalari ma’lumotlar bazasida ma’lumotlarni tartibga solish va saqlash uchun ishlatiladi, bu esa samarali qidirish va manipulyatsiya qilishga imkon beradi.
- *Operatsion tizimlar:* ma’lumotlar tuzilmalari xotira va fayllar kabi tizim resurslarini boshqarish uchun operatsion tizimlarni loyihalash va amalga oshirishda qo‘llaniladi.
- *Kompyuter Grafikasi:* ma’lumotlar tuzilmalari kompyuter Grafsi dasturlarida geometrik shakllar va boshqa Graf elementlarni ifodalash uchun ishlatiladi.
- *Sun’iy intellekt:* ma’lumotlar tuzilmalari sun’iy intellekt tizimlarida bilim va ma’lumotlarni ifodalash uchun ishlatiladi.

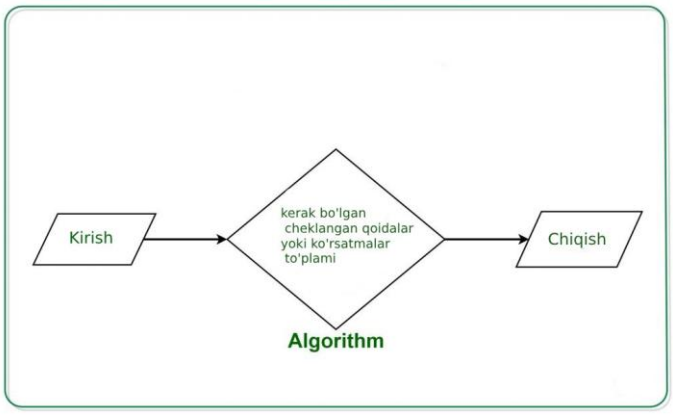
Ma’lumotlar tuzilmalaridan foydalanishning afzalliklari:

- *Samaradorlik:* ma’lumotlar tuzilmalari ma’lumotlarni samarali saqlash va olish imkonini beradi, bu ishlash juda muhim bo‘lgan ilovalarda muhim ahamiyatga ega.
- *Moslashuvchanlik:* ma’lumotlar tuzilmalari ma’lumotlarni tartibga solish va saqlashning moslashuvchan usulini ta’minlaydi, bu esa oson o‘zgartirish va manipulyatsiya qilish imkonini beradi.
- *Qayta foydalanish imkoniyati:* ma’lumotlar tuzilmalari bir nechta dastur va dasturlarda ishlatilishi mumkin, bu ortiqcha kodga bo‘lgan ehtiyojni kamaytiradi.
- *Ta’minot:* yaxshi ishlab chiqilgan ma’lumotlar tuzilmalari dasturlarni vaqt o‘tishi bilan tushunish, o‘zgartirish va saqlashni osonlashtirishi mumkin.

1.2 Algoritm tushunchalar

Algoritm bu odatda ma’lum bir muammolar guruhini yechish yoki ma’lum bir hisoblash turini bajarish uchun ishlatiladigan jarayon yoki aniq belgilangan ko‘rsatmalar to‘plami sifatida aniqlanadi. Algoritm so‘zi hisob-kitoblarda yoki boshqa muammolarni hal qilish operatsiyalarida bajarilishi kerak bo‘lgan

cheklangan qoidalar yoki ko‘rsatmalar to‘plami degan ma’noni anglatadi.Algoritmlar turli sohalarda hal qiluvchi rol o‘ynaydi va ko‘plab ilovalarga ega. Algoritmlardan foydalaniladigan ba’zi asosiy sohalarga quyidagilar kiradi:



Rasm- 2 Algoritm tushinchasi

- Kompyuter fanlari: algoritmlar kompyuter dasturlashning asosini tashkil etadi va oddiy saralash va qidirishdan tortib sun’iy intellekt va mashinani o‘rganish kabi murakkab vazifalargacha bo‘lgan muammolarni hal qilish uchun ishlatiladi.
- Matematika: algoritmlar matematik masalalarni echish uchun ishlatiladi, masalan, chiziqli tenglamalar tizimining optimal echimini topish yoki grafda eng qisqa yo‘lni topish.
- Operatsiyalarni o‘rganish: algoritmlar transport, logistika va resurslarni taqsimlash kabi sohalarda optimallashtirish va qaror qabul qilish uchun ishlatiladi.
- Sun’iy intellekt: algoritmlar sun’iy intellekt va mashinani o‘rganishning asosi bo‘lib, tasvirni aniqlash, tabiiy tilni qayta ishlash va qaror qabul qilish kabi vazifalarni bajara oladigan aqlli tizimlarni ishlab chiqish uchun ishlatiladi.
- Ma’lumotlar tuzilmalari: algoritmlar marketing, moliya va sog’liqni saqlash kabi sohalarda katta hajmdagi ma’lumotlarni tahlil qilish, qayta ishlash va ulardan tushunchalarni olish uchun ishlatiladi.

Algoritmlar nimaga kerak?

- Algoritmlar murakkab muammolarni samarali hal qilish uchun zarurdir.
- Jarayonlarni avtomatlashtirishga yordam beradi va ularni yanada ishonchli, tezroq va bajarishni osonlashtiradi.
- Algoritmlar, shuningdek, kompyuterlarga odamlar qo‘lda bajarishi qiyin yoki imkonsiz bo‘lgan vazifalarni bajarishga imkon beradi.
- Matematika, informatika, muhandislik, moliya va boshqa ko‘plab sohalarda jarayonlarni optimallashtirish, ma’lumotlarni tahlil qilish, bashorat qilish va muammolarga echim topish uchun ishlatiladi.

Algoritmning xususiyatlari

Algoritmning asosiy xossalari haqida quyidagilarni ta’kidlash mumkin:

1- xossa. Diskretlilik, ya’ni algoritmni chekli sondagi oddiy ko‘rsatmalar ketma-ketligi shaklida ifodalash mumkin.

2- xossa. 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.

3- xossa. Aniqlik, ya’ni ijrochiga berilayotgan ko‘rsatmalar aniq mazmunda bo‘lishi lozim hamda faqat algoritmda ko‘rsatilgan tartibda bajarilishi shart.

4- xossa. 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.

5- xossa. Natijaviylik, ya’ni har bir algoritm chekli sondagi qadamlardan so‘ng albatta natija berishi lozim.

Bu xossalar mohiyatini o‘rganish va konkret algoritmlar uchun qarab chiqish talabalarning xossalar mazmunini bilib olishlariga yordam beradi.

Algoritmlarning afzalliklari:

- Buni tushunish oson.
- Berilgan muammoning echimini bosqichma-bosqich namoyish etish.
- Algoritmda muammo kichikroq qismlarga yoki bosqichlarga bo‘linadi, shuning uchun dasturchi uni haqiqiy dasturga aylantirishi osonroq bo‘ladi.

Algoritmlarning kamchiliklari:

- Algoritmni yozish uzoq vaqt talab etadi, shuning uchun ko‘p vaqt sarf qilinadi.
- Algoritmlar orqali murakkab mantiqni tushunish juda qiyin bo‘lishi mumkin.

1.3 Algoritmning murakkabligi.

Algoritm sarflanadigan xotira va vaqt miqdoriga qarab murakkabligi topiladi. Demak, algoritmning murakkabligi uni bajarish va kutilgan natijani olish uchun zarur bo‘lgan vaqt o‘lchovini va barcha ma’lumotlarni (kirish va chiqish) saqlash uchun zarur bo‘lgan xotirani anglatadi. Shuning uchun bu ikki omil algoritm samaradorligini belgilaydi.

***Vaqt omili**:* vaqt saralash algoritmidagi taqqoslash kabi asosiy operatsiyalar sonini hisoblash bilan o‘lchanadi.

Xotira omili: bo‘sh xotira algoritm tomonidan ishlash/bajarish uchun zarur bo‘lgan maksimal xotira maydonini hisoblash orqali o‘lchanadi.

Shuning uchun algoritmning murakkabligini ikki turga bo‘lish mumkin:

1. Xotira murakkabligi: algoritmning xotira murakkabligi o‘zgaruvchilarni saqlash va natijani olish uchun algoritm tomonidan talab qilinadigan xotira hajmini anglatadi. Bu kirishlar, vaqtinchalik operatsiyalar yoki chiqishlar uchun bo‘lishi mumkin. Algoritmning xotira murakkabligi quyidagi komponentnlarni aniqlash orqali hisoblanadi:

- Ruxsat etilgan qism: bu algoritm talab qiladigan bo‘sh joyni anglatadi. Masalan, kirish o‘zgaruvchilari, chiqish o‘zgaruvchilari, dastur hajmi va boshqalar.
- O‘zgaruvchan qism: bu algoritmni amalga oshirish asosida har xil bo‘lishi mumkin bo‘lgan bo‘shliqni anglatadi. Masalan, vaqtinchalik o‘zgaruvchilar, dinamik xotira ajratish, rekursiya stek maydoni va boshqalar.

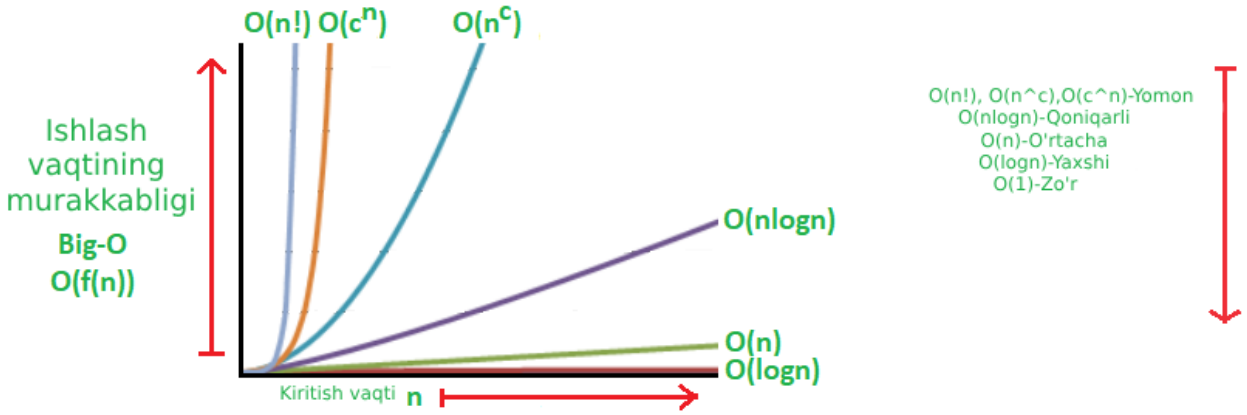
2. Vaqt murakkabligi: algoritmning vaqt murakkabligi algoritmni bajarish va natijani olish uchun zarur bo‘lgan vaqtni anglatadi. Bu oddiy operatsiyalar, shartli if-else, takrorlanish operatorlari va boshqalar uchun bo‘lishi mumkin. Algoritmning vaqt murakkabligi quyidagi komponentlarni aniqlash orqali ham hisoblanadi:

- Doimiy vaqt qismi: faqat bir marta bajarilgan har qanday ko‘rsatma ushbu qismga kiradi. Masalan, kirish, chiqish, if-else, almashtirish, arifmetik operatsiyalar va boshqalar.
- O‘zgaruvchan vaqt qismi: bir necha marta bajariladigan har qanday ko‘rsatma, aytaylik n marta, ushbu qismda keladi. Misol uchun, sikllar, rekursiya va hokazo.

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, vaqt murakkabligi: bu protsessorga, 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.

Asimptotik murakkablik bu kirish ma’lumotlarining kattaligi cheksizlikka intilayotgandagi murakkablik hisoblanadi. Katta **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.



Rasm- 3 Algoritmning ishlash vaqtining murakkabligi

Ushbu turdagi algoritmlarning ba’zi misollari (eng yomon stsenariylarda) quyida keltirilgan:

- O(logn) – Binar qidiruv (Binary search).
- O(n) – Chiziqli qidiruv (Linear search).
- O(nlogn) – birlashtirib saralash (merge sort).
- $O(n^2)$ –Pufakchali saralash (Bubble sort).

1.4 Katta O murakkabligini aniqlash uchun amaliy misollari

Algoritmlarning ishlashini yoki algoritmik murakkablikni o‘rganish algoritmlarni tahlil qilish sohasiga tegishli. Ushbu usul muammoni hal qilish uchun zarur bo‘lgan resurslarni (masalan, disk xotirasi yoki vaqt) hisoblab chiqadi. Bu erda biz birinchi navbatda vaqtga e’tibor qaratamiz, chunki algoritm vazifani qanchalik tez bajara olsa, u shunchalik samarali bo‘ladi.

Katta O belgisi bizga kirish hajmi algoritmning ishlash vaqtiga qanday ta’sir qilishini tahlil qilishga yordam beradi. Big O ning ma’nosini tushunish uchun o‘sish tezligini bilish muhimdir. Bu har bir kirish hajmi uchun zarur bo‘lgan vaqtni anglatadi. Ba’zi algoritmlarni o‘rganamiz va ularning vaqt murakkabligini baholaymiz.

1. O‘zgarmas vaqt algoritmlari (Constant Time Algorithms) – O(1)

Avval n ta o‘zgaruvchisini 10000 qiymati bilan ishga tushiradigan va keyin uni chiqaradigan oddiy algoritmni ko‘rib chiqamiz:

Algoritm 1: o‘zgaruvchini ishga tushirish va chop etish

```
Data: Initial state
//Ma’lumotlar: dastlabki holat
Result: Initialize a variable with a big number and print it.
//Natija: katta raqam bilan o‘zgaruvchini ishga tushiring va uni chop eting.
n <-10000;
print n;
//chop etish n;
```

Ushbu kod n qiymatidan qat’iy nazar, belgilangan vaqt ichida bajariladi va algoritmning vaqt murakkabligi O(1) ga teng. Shu bilan bir qatorda, for sikl yordamida n o‘zgaruvchini uch marta chop etishimiz mumkin:

Algoritm 2: O‘zgaruvchini 3 marta ishga tushirish va chop etish

```
Data: Initial state
```

```
// Ma’lumotlar: dastlabki holat
Result: Initialize a variable with a big number and print it thrice.
// Natija: biz o‘zgaruvchini katta raqam bilan boshlaymiz va uni uch marta chop etamiz.
n < - 10000;
for i in range(1, 4) do // i (1,4) oralgi’i uchun
print n; //chiqarish n
end  //tamom
```

Yuqoridagi misol ham o‘zgarmas vaqtga (konstantta) ega. Bajarish uchun uch baravar ko‘p vaqt kerak bo‘lsa ham, bu n kirish hajmiga bog’liq emas . biz doimiy vaqt algoritmlarini O(1) deb belgilaymiz. Kirish hajmidan qat'iy nazar, bu odatdagidan uch baravar ko‘p vaqt talab etadi. Shuning uchun O(2), O(3) yoki hatto O(1000) ham O(1) bilan bir xil.Ishga tushirish uchun qancha vaqt ketishi bizni qiziqtirmaydi, faqat muhim narsa shundaki, bu o‘zgarmas (konstanta) vaqtni oladi.

2. Logaritmik vaqt algoritmlari (Logarithmic Time Algorithms)-O(log (n))

Asimptotik ravishda o‘zgarmas vaqt algoritmlari eng tezkor hisoblanadi. Keyinchalik logaritmik vaqt murakkabligiga ega algoritmlar keladi. Biroq, ularni tasavvur qilish qiyinroq. Logaritmik vaqt algoritmining odatiy misollaridan biri bu ikkilik qidirish algoritmi:

```
Algoritm 3: Ikkilik Qidiruv
Data: Sorted array A, and target value x
// Ma’lumotlar: saralangan massiv a va maqsad qiymati x
Result: Index of x in A, or -1 if not found
// Natija: indeks A ichida x, yoki -1 agar topilmasa
```

```
low <- 0;
high <- len(A) - 1;
while low _ high do
     mid <- (low + high) /2;
     if A[mid] < x then
        |low <- mid + 1;
     end
     else if A[mid] > x then
|       high <- mid - 1;
     end
     else
   return mid;
   end
end
return-1;
```

Ikkilik qidiruv algoritmida har bir iteratsiyada maqsadli qiymatni topmaguncha yoki -1 ni qaytarmagunisha massivni ikkiga bo‘ladi. Shunday qilib algoritmning ish vaqti $\log_2(n)$ funktsiyasiga mutanosib, bu erda n massivdagi elementlar soni. Masalan, n=8 ga teng bo‘lsa, $\log_2(8) = 3$ marta takrorlanadi.

3. Chiziqli algoritmlar(Linear Time Algorithms)-O (n)

Vaqt murakkabligi ularning kirish hajmiga mutanosib bo‘lgan chiziqli vaqt algoritmlarini ko‘rib chiqamiz.

```
Algoritm 4: Raqam Hisoblagichi
Data: Input value n
//Ma’lumotlar: n qiymatini kiritish
Result: Print numbers from 1 to n
```

```
// 1 dan n gacha raqamlarni chop etish
for i in range(1, n + 1) do //(1,n+1)oralig'ida
     print i;
end
```

Ushbu misolda n takrorlash soni kirish hajmiga to‘g’ridan-to‘g’ri proportsionaldir, ns. sifatida n ortadi, algoritmni bajarish uchun vaqt chiziqli ravishda ortadi. Shuning uchun algoritmning vaqt murakkabligi O(n). Vaqt murakkabligini belgilashda biz 0,1 n yoki (1000n+1000) o‘rtasida farq qilmaymiz, chunki ikkalasi ham O(n) vaqt murakkabligiga ega va kirish hajmi bilan bevosita bog’liq o‘sadi.

4.nlogn vaqt murakkabliqi algoritmlar-O(nlogn)

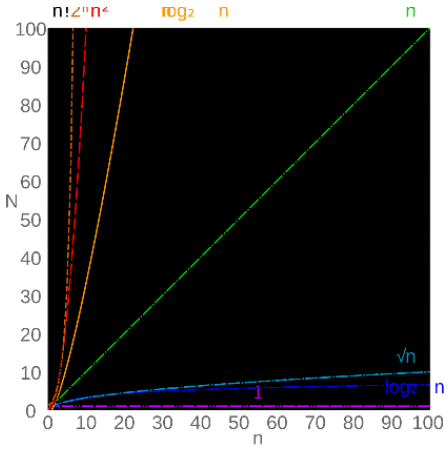
Nlogn algoritmlari chiziqli vaqt murakkabligiga ega algoritmlarga qaraganda yomonroq ishlaydi. Buning sababi shundaki, ularning ishlash vaqti kirish hajmi bilan chiziqli va logaritmik ravishda oshadi. Masalan, for tskil operatori bilan quyidagi algoritmni ko‘rib chiqamiz:

Algoritm 5: ikkita raqamning barcha kombiatsiyalari

```
Data: Input value n
//Ma'lumotlar: n kirish qiymati n
Result: Print all pairs of numbers from 1 to
//1 dan n gacha bo‘lgan barcha juft raqamlarni chop
eting
       for i in range(1, n + 1) do
           for jin range(1, log(n) + 1) do
           print (i, j); //chop etish (i, j);
       end
```

```
end
```

Ushbu misolda tashqi sikl n marta, ichki sikl esa log (n) marta ishlaydi. Sikllar joylashtirilganligi sababli, umumiy son n * log(n) va biz algoritmning vaqt murakkabligini o(n*log(n)) deb belgilaymiz. Nlogn vaqt algoritmining yana bir misoli Quicksort algoritmi.



Rasm- 4Taqdim etilgan ayrim vaqt murakkabliklarini kirish hajmi va vaqt murakkabligi bilan grafikda ko‘rinishi

Algoritmlarning turli xil chiqishlari (ish vaqti) n (kirish hajmi) kattalashganda tez ajralib chiqadi. Matematik misolni ko‘rib chiqamiz:

Agar n = 10	Agar n=20,
log(10) = 1;	log(20) = 2.996;
10 = 10;	20 = 20;
10log(10)=10;	20log(20)=59.9;
10^2=100;	20^2=400;
2^{10}=1024;	220=1048576;
10!=3628800;	20!=2.432902e+1818;

Jadval- 1 Ish vaqtini tahlil qilish

Murakkablik	**n**	***Operatsialar soni (10)***	**Bajarish vaqti/µsec**
konstantalik	*O(1)*	*1*	*1 µsec*
logarifimlik	*O(logn)*	*3.32*	*3 µsec*
chiziqli	*O(n)*	*10*	*10 µsec*
O(nlogn)	*O(nlogn)*	*33.2*	*33 µsec*
kvadrat	$O(n^2)$	10^2	*100 µsec*
kub	$O(n^3)$	10^3	*1msec*
exponensial	$O(2^n)$	*1024*	*10 msec*
faktorial	*O(n!)*	*10!*	*3.6288 sec*

Jadval- 2 Algoritmlar sinflari va ularni sekundiga 1 million operatsiyani bajaradigan kompyuterda bajarish vaqtlari (1 sek = 106 µsec):

Mavzu yuzasidan savollar:

1 Ma’lumotlar tuzilmasiga ta’rif bering.

2 Ma’lumotlar tuzilmalarining qo‘llanish joylari.

3 Ma’lumotlar tuzilmalarining toifalarni tushintirib bering.

4 Algoritm tushunchasiga ta’rif bering.

5 Algoritm xossalari va uni tasvirlash usullariga to‘xtalib o‘ting

6 Algoritm yaxshi, o‘rtacha, yomon bahosiga misollar keltiring

7 Bosh harf O dan foydalanish.