3-§. Saralash algoritmlari. Eng oddiy algoritmlar. Past baho

Mazkur mavzuda algoritmlarning yangi sinfi - saralash algoritmlarini o‘rganamiz. Ma’lumotlarni qayta ishlashda ma’lumotning axborot (**info**) maydonini bilish va uni mashinada joylashishini tasavvur qilish juda muhimdir.

Saralashning ichki va tashqi saralash turi mavjud:

1.ichki saralash - bu tezkor xotiradagi ma’lumotlarni saralash;

2. tashqi saralash - tashqi xotira (fayllar)dagi ma’lumotlarni saralash.

Saralash deganda ma’lumotlarni xotirada muayyan tartibda uning kalitlari bo‘yicha joylashtirish, muayyan tartib deganda esa kalit qiymati bo‘yicha o‘sish (yoki kamayish) tartibida ro‘yxatning boshidan oxirigacha joylashtirish tushuniladi.

Saralash algoritmlarining samaradorligini bir necha parametrlari bo‘yicha farqlash mumkin. Bu parametrlarning asosiylari quyidagilar hisoblanadi:

- saralash uchun sarflanadigan vaqt;
- saralash uchun talab qilinadigan tezkor xotira hajmi.

Saralash algoritmlarini baholashda faqat “joyida” saralash usullarini qarab chiqamiz, ya’ni saralash jarayoni uchun qo‘shimcha xotira zahirasi talab qilinmaydi. Saralash uchun sarf qilinadigan vaqtni esa, saralash bajarilishi jarayonida amalga oshiriladigan taqqoslashlar va o‘rin almashtirishlar soni orqali hisoblash mumkin. Ixtiyoriy saralash usulida taqqoslashlar soni $O(n\log_2 n)$ dan $O(n^2)$ gacha bo‘lgan oraliqda yotadi.

Ma’lumotlarni saralashning **qat’iy (to‘g‘ri) va yaxshilangan** usullari mavjud bo‘lib, qat’iy usullariga quyidagilarni misol qilib olish mumkin:

1) to‘g‘ridan-to‘g‘ri qo‘yish orqali saralash usuli;
2) to‘g‘ridan-to‘g‘ri tanlash orqali saralash usuli;
3) to‘g‘ridan-to‘g‘ri almashtirish orqali saralash usuli.

Bu uchala saralash usullarining samaradorligi deyarli bir xil.

Lug‘atlarda "saralash" (sorting) so‘zi "toifalarga ajratish, tartiblash, baholash" deb ta’riflanadi, ammo dasturchilar odatda bu so‘zni tor ma’noda ishlatishadi, ularga ba’zi bir aniq tartibda elementlarni qayta joylashtirishga murojaat qilishadi. Bu jarayonni, ehtimol, saralash deb emas, balki tartiblash (ordering) yoki ketma-ketlik (sequencing) deb atash kerak. Biroq, saralash so‘zi dasturlash jargonida allaqachon mustahkam o‘rnashgan, shu sababli kelajakda "saralash" so‘zini tor ma’noda "tartiblash" dan foydalanamiz. Bu shuni anglatadiki, endi "saralash" ta’rifini shakllantirishimiz mumkin, bu kelgusida ishlatiladi.

Tartiblash – bu berilgan obyektlar to‘plamini muayyan tartibda qayta tartibga solish jarayoni. Saralashning maqsadi elementlarni topishni osonlashtirishdir.

Saralash algoritmi – bu ro‘yxatdagi elementlarni saralash algoritmi. Agar ro‘yxat elementida bir nechta maydon bo‘lsa, saralash amalga oshiriladigan maydon **saralash kaliti** deb ataladi. Amalda raqam ko‘pincha kalit sifatida ishlatiladi va ba’zi ma’lumotlar algoritm ishlashiga hech qanday ta’sir ko‘rsatmaydigan qolgan maydonlarda saqlanadi.

Ehtimol, boshqa hech qanday muammo saralash muammosi kabi juda ko‘p turli xil yechimlarni keltirib chiqarmagan. Butun dunyoda tan olingan eng yaxshi algoritm bormi? Umuman aytganda, yo‘q. Biroq, kirish ma’lumotlarining taxminiy xususiyatlarini hisobga olgan holda, siz eng yaxshi ishlaydigan usulni tanlashingiz mumkin.

Saralash usullari juda ko‘p, ularning har biri o‘zining afzalliklari va kamchiliklariga ega. Tartiblash algoritmlari katta amaliy ahamiyatga ega, ular o‘zlari uchun qiziqarli. Bu juda chuqur o‘rganilgan informatika sohasi axborot qidirish tizimlarida, harbiy sohada va bank sohalarida ko‘proq qo‘llaniladi. Ammo hozirgi kunda axborot oqimini tartiblash masalasi deyarli har bir sohaga kirib bordi.

Algoritmlarni saralashga bo‘lgan umumiy ilmiy qiziqishdan tashqari, har bir algoritmda uning **murakkabligi** deb ataladigan narsani baholash qiziq. Murakkablik algoritmning boshlang‘ich bosqichlarining maksimal soni sifatida tushuniladi. Tartiblash misollari algoritmni murakkablashtirish orqali ko‘rsatilishi mumkin, garchi hozirda aniq

usullar mavjud bo‘lsa-da, siz samaradorlikda sezilarli yutuqqa erishishingiz mumkin.

Massivlarni saralash masalasini yechishda odatda qo‘shimcha xotiradan foydalanishni minimallashtirish talabi qo‘yiladi, bu esa qo‘shimcha massivlardan foydalanishga yo‘l qo‘yilmasligini anglatadi.

Quyidagi ko‘rsatkichlar ham muhimdir:

• **Xotira.** Bir qator algoritmlar ma’lumotlarni vaqtincha saqlash uchun qo‘shimcha xotira ajratishni talab qiladi. Amaldagi xotirani baholashda boshlang‘ich massiv egallagan joy, kirish ketma-ketligidan mustaqil sarflangan xotira, masalan, dastur kodini saqlash uchun ajratilgan joy hisobga olinmaydi.

• **Turg‘unlik**. Doimiy saralash teng elementlarning nisbiy holatini o‘zgartirmaydi. Ushbu xususiyat juda foydali bo‘lishi mumkin, agar ular bir nechta maydonlardan iborat bo‘lsa va saralash ulardan biri tomonidan amalga oshirilsa.

Barcha saralash usullarini ikkita katta guruhga bo‘lish mumkin:

• Saralashning bevosita usullari;

• Takomillashtirilgan saralash usullari;

To‘g‘ridan-to‘g‘ri tartiblash usullari usul asosida yotadigan prinsipga ko‘ra, uchta kichik guruhga bo‘linadi:

• oddiy qo‘shimchalar bo‘yicha saralash (qo‘shish);

• tanlash (saralash) bo‘yicha saralash;

• Almashish bo‘yicha saralash ("Pufakchali" saralash).

Takomillashtirilgan tartiblash usullari to‘g‘ridan-to‘g‘ri prinsiplarga asoslanadi, ammo saralash usulini tezlashtirish uchun ba’zi bir original g‘oyalardan foydalaniladi. To‘g‘ridan-to‘g‘ri saralash usullari amalda kamdan kam qo‘llaniladi, chunki ular nisbatan past ko‘rsatkichlarga ega. Biroq, ular shu usullar asosida kelib asoslangan takomillashtirilgan usullarning mohiyatini yanada yaxshi namoyish etadi. Bundan tashqari, ba’zi hollarda (odatda massivning uzunligi kichik bo‘lsa yoki yoki massiv elementlarining boshlang‘ich joylashuvi bilan) to‘g‘ridan-to‘g‘ri usullar takomillashtirilgan usullardan ham ustun bo‘lishi mumkin.

3.1. Ichki saralash muammosining bayoni va samaradorlikni baholash yondashuvlari

Ko‘pgina hollarda, ma’lumotlarning ba’zi bir mezonlarga muvofiq tartiblanishi ma’lumotlarni qayta ishlashni soddalashtiradi. Masalan, ikkilik qidiruvni ketma-ket qidirishni amalga oshirishda vaqtni sezilarli darajada tejash, ikkilik yoki boshqa turdagi qidiruv algoritmlarini amalga oshirishda katta yutuqlarni ta’minlash uchun ma’lumotlar to‘plamini oldindan saralashga vaqt sarflash uchun yetarli sababdir.

Eng muhimi, in situ (joylashtirish) bo‘yicha tartiblash algoritmlari bo‘lib, ular tartiblangan ketma-ketlikni egallagan xotira maydoni ichidagi elementlarning almashinishini ta’minlaydi. Bunday holda, ma’lumotlar ketma-ketligi odatda tezkor xotirada joylashgan massiv sifatida tushuniladi - bunday massivlarni saralash ichki saralash deb ataladi, aksincha tashqi saralash - ba’zi tashqi manbalardan ma’lumotlarni olish (masalan, diskdagi fayl) sifatida tushuniladi.

Odatda ma’lumotlar ba’zi bir muhim qiymatlarning ko‘tarilish yoki kamayish tartibida saralanadi.

Algoritmni tahlil qilish ushbu algoritm yordamida muammoni hal qilish uchun qancha vaqt sarflanishi haqida tasavvurga ega bo‘lishni o‘z ichiga oladi. Algoritmni baholashda, ma’lum bir algoritm uchun uning ishlashi davomida eng muhim bo‘lgan amallar sonidan kelib chiqadi. Saralash algoritmlari uchun bunday amallar asosiy taqqoslash amallari va tartiblangan elementlarning uzatish amallari hisoblanadi.

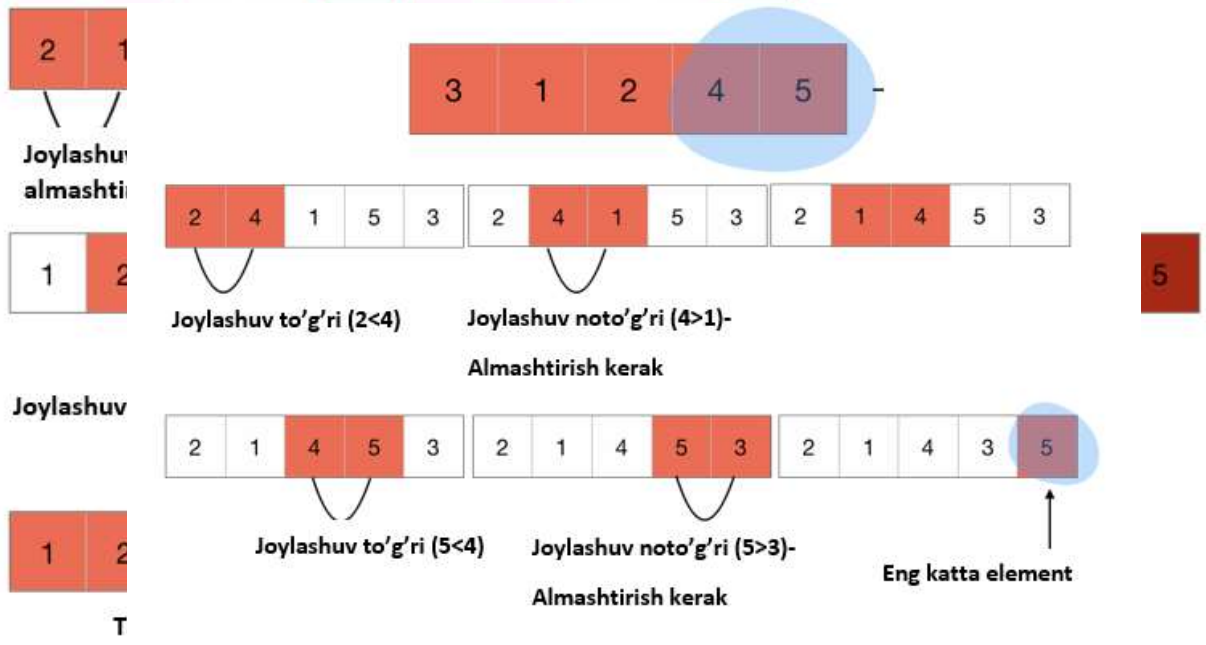
Algoritm samaradorligini baholashda, odatda, uchta variantni baholashga harakat qilinadi: eng yaxshi holat (vazifa eng qisqa vaqt ichida amalga oshirilganda), eng yomon holat (vazifa maksimal vaqt ichida amalga oshirilganda) va o‘rta holat , (buni odatda tahlil qilish eng qiyin). Algoritmlarni saralashning asosiy ko‘rsatkichlari bu algoritm ishlashi davomida amalga oshirilgan taqqoslash va almashtirishlarning o‘rtacha soni.

Shu bilan birga, algoritm tomonidan bajariladigan amallar sonini aniq bilish samaradorlikni tahlil qilish nuqtai nazaridan juda muhim

emas. N - massiv elementlari sonining ko‘payishi bilan amallar sonining o‘sish sur’ati muhimroqdir.

3.2. Oddiy saralash algoritmlari va ularning tahlili

Pufakchali saralash (Bubble sort). Ushbu algoritmda massivni takrorlash bilan boshlaymiz va birinchi elementni ikkinchisiga taqqoslaymiz va agar ular noto‘g‘ri tartibda joylashgan bo‘lsa, ularni



11-rasm. Pufakchali saralash algoritmining ishlash prinsipi

almashtiramiz, keyin ikkinchi va uchinchisini taqqoslaymiz va hokazo. Ushbu takrorlashdan so‘ng eng katta element quyida keltirilgan rasmda ko‘rsatilgandek massivning oxiriga o‘tadi.

Endi eng katta element o‘z joyini egallaydi, shuning uchun biz yana ushbu jarayonni takrorlaymiz va allaqachon o‘z pozitsiyalarida bo‘lgan elementlarni qoldiramiz va bu tartiblangan massivni beradi.

Butun jarayonni quyidagi bosqichlarda yozishimiz mumkin:

1) Massiv bo‘yicha takrorlashni boshlash.

2) Qo‘shni elementlarni solishtirish. Masalan, birinchi va ikkinchi, ikkinchi va uchinchi va boshqalar.

3) Agar ular tartiblangan bo‘lmasa, ularni almashtirish.

4) To‘g‘ri pozitsiyalariga joylashtirilgan elementlardan tashqari, ushbu amallarni takrorlash.

Pufakchali saralash (Bubble sort) algoritmining C++ dastur kodi. Ushbu algoritm ta'limiy hisoblanadi va samaradorligi pastligi sababli amalda deyarli hech qachon qo'llanilmaydi: u kichik elementlar (ular "toshbaqalar" deb nomlanadi) massiv oxirida joylashgan testlarda sekin ishlaydi. Biroq, ko'plab boshqa usullar bunga asoslangan, masalan, Sheyker saralash va taroqsimon saralashlari.

Vaqt bo'yicha murakkabligi:

Eng yomon baho: $O(n^2)$

O'rtacha baho: $O(n^2)$

Eng yaxshi baho: O(n)

```
void bubbleSort(int arr[], int n)
{
  int i, j;
  bool swapped;
  for (i = 0; i < n-1; i++)
  {
    swapped = false;
    for (j = 0; j < n-i-1; j++)
    {
      if (arr[j] > arr[j+1])
      {
        swap(&arr[j], &arr[j+1]);
        swapped = true;
      }
    }

    if (swapped == false)
      break;
  }
}
```

Sheyker saralashi. Sheyker (kokteyl) saralashi - bu Pufakchali (Bubble Sort) algoritmining varianti. Bubble sort algoritmi har doim chapdan elementlarni aylanib o'tadi va birinchi iteratsiyada eng katta elementni to'g'ri holatiga, ikkinchisida esa ikkinchi takrorlashda va

hokazo. Kokteyl saralash berilgan massiv orqali har ikki yo‘nalishda ham muqobil ravishda harakatlanadi.

Algoritmning har bir takrorlanishi 2 bosqichga bo‘linadi:

Birinchi bosqich massivni xuddi Bubble Sort singari chapdan o‘ngga aylantiradi. Sikl davomida qo‘shni elementlar taqqoslanadi va agar chapdagi qiymat o‘ngdagi qiymatdan katta bo‘lsa, qiymatlar almashtiriladi. Birinchi takrorlash oxirida eng katta son massivning oxirida bo‘ladi.

Ikkinchi bosqich massivni qarama-qarshi yo‘nalishda aylantiradi - bu eng so‘nggi saralangan elementdan oldin va massivning boshiga qaytish. Bu yerda qo‘shni elementlar taqqoslanadi va agar kerak bo‘lsa almashtiriladi.

Quyidagi massivni ko‘rib chiqaylik (6 1 4 2 8 0 2).

Birinchi oldinga o‘tish:

(6 1 4 2 8 0 2)? (1 6 4 2 8 0 2), 6> 1 bilan almashtirish
(1 6 4 2 8 0 2)? (1 4 6 2 8 0 2), 6> 4 bilan almashtirish
(1 4 6 2 8 0 2)? (1 4 2 6 8 0 2), 6> 2 bilan almashtirish
(1 4 2 6 8 0 2)? (1 4 2 6 8 0 2)
(1 4 2 6 8 0 2)? (1 4 2 6 0 8 2), 8> 0 bilan almashtirish
(1 4 2 6 0 8 2)? (1 4 2 6 0 2 8), 8> 2 bilan almashtirish

Birinchi oldinga o‘tishdan so‘ng, massivning eng katta elementi massivning oxirgi indeksida bo‘ladi.

Birinchi orqaga o‘tish:

(1 4 2 6 0 2 8)? (1 4 2 6 0 2 8)
(1 4 2 6 0 2 8)? (1 4 2 0 6 2 8), 6> 0 bilan almashtirish
(1 4 2 0 6 2 8)? (1 4 0 2 6 2 8), 2> 0 bilan almashtirish
(1 4 0 2 6 2 8)? (1 0 4 2 6 2 8), 4> 0 bilan almashtirish
(1 0 4 2 6 2 8)? (0 1 4 2 6 2 8), 1> 0 bilan almashtirish

Birinchi orqaga uzatgandan so‘ng, massivning eng kichik elementi massivning birinchi indeksida bo‘ladi.

Ikkinchi oldinga o‘tish:
(0 1 4 2 6 2 8)? (0 1 4 2 6 2 8)
(0 1 4 2 6 2 8)? (0 1 2 4 6 2 8), 4>2 bilan almashtirish
(0 1 2 4 6 2 8)? (0 1 2 4 6 2 8)
(0 1 2 4 6 2 8)? (0 1 2 4 2 6 8), 6>2 bilan almashtirish

Ikkinchi orqaga o‘tish:

(0 1 2 4 2 6 8)? (0 1 2 2 4 6 8), 4>2 bilan almashtirish

Endi, massiv allaqachon saralangan, ammo bizning algoritmimiz tugallanganligini bilmaydi. Algoritm bu saralanganligini bilish uchun barcha o‘tishlarni hech qanday almashtirishsiz bajarishi kerak.

(0 1 2 2 4 6 8) ? (0 1 2 2 4 6 8)
(0 1 2 2 4 6 8) ? (0 1 2 2 4 6 8)

Quyida yuqoridagi algoritmning bajarilishi keltirilgan:

```
void CocktailSort(int a[], int n)
{
   bool swapped = true;
   int start = 0;
   int end = n - 1;

   while (swapped)
   {
      swapped = false;

      for (int i = start; i < end; ++i)
      {
         if (a[i] > a[i + 1]) {
            swap(a[i], a[i + 1]);
            swapped = true;
         }
      }
```

```
        if (!swapped)
            break;

        swapped = false;

        --end;

        for (int i = end - 1; i >= start; --i)
        {
            if (a[i] > a[i + 1]) {
                swap(a[i], a[i + 1]);
                swapped = true;
            }
        }
        ++start;
    } }
```

Vaqt bo‘yicha murakkabligi:
Eng yomon baho: $O(n^2)$
O‘rtacha baho: $O(n^2)$
Eng yaxshi baho: O(n)

3. Taroqsimon saralash. Comb sort. Taroqsimon saralash – “Pufakchali” saralashning yaxshiroq varianti. Uning g‘oyasi algoritmni sekinlashtiradigan qator oxiridagi kichik qiymatlarga ega elementlarni "yo‘q qilish". Agar pufakchali va Sheyker saralashlarida, massiv bo‘ylab takrorlanganda qo‘shni elementlar taqqoslansa, u holda "tarash" paytida avval taqqoslangan qiymatlar orasida yetarlicha katta masofa olinadi, so‘ngra u minimal darajaga tushadi.

Dastlabki bo‘shliq tasodifiy tanlanmasligi kerak, lekin maxsus qiymatni hisobga olgan holda - kamaytirish qiymati, uning optimal qiymati 1,247 ga teng. Dastlab elementlar orasidagi masofa massivning kattaligiga 1,247 ga teng bo‘ladi; har bir keyingi bosqichda masofa yana qisqartirish koeffitsiyentiga bo‘linadi - va shunga o‘xshash jarayon algoritm oxirigacha davom etadi.

Vaqt bo‘yicha murakkabligi:
Eng yomon baho: $O(n^2)$
O‘rtacha baho: $\Omega(n^2/2^p)$, p – inkrementlar miqdori
Eng yaxshi baho: O(n logn)

```
void combSort(int a[], int n)
{
   int k = n;

   bool swapped = true;

   while (k != 1 || swapped == true)
   {
      k = getNextk(gap);

      swapped = false;

      for (int i=0; i<n-k; i++)
      {
         if (a[i] > a[i+k])
         {
            swap(a[i], a[i+k]);
            swapped = true;
         }
      }
   }
}
```

4. Joylashtirish bo‘yicha saralash (Insertion sort). Joylashtirish bo‘yicha saralashda massiv asta-sekin chapdan o‘ngga takrorlanadi. Bunday holda, har bir keyingi element minimal va maksimal qiymatga ega bo‘lgan eng yaqin elementlar orasida bo‘lishi uchun joylashtiriladi.

Vaqt bo‘yicha murakkabligi:
Eng yomon baho: $O(n^2)$
O‘rtacha baho: $O(n^2)$,

Eng yaxshi baho: O(n)

```
void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++)
   {
      key = arr[i];
      j = i - 1;


      while (j >= 0 && arr[j] > key)
      {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}
```

5. Tanlash bo‘yicha saralash (Selection sort). Birinchidan, siz massivning kichik qismini ko‘rib chiqishingiz va undagi maksimal (yoki minimal) miqdorni topishingiz kerak. Keyin tanlangan qiymat birinchi saralanmagan elementning qiymati bilan almashtiriladi. Ushbu qadam massivning saralanmagan ichki qismlari tugamaguncha takrorlanishi kerak.

Vaqt bo‘yicha murakkabligi:
Eng yomon baho: $O(n^2)$
O‘rtacha baho: $O(n^2)$,
Eng yaxshi baho: $O(n^2)$

```
void selectionSort(int arr[], int n)
{
   int i, j, min_idx;

   for (i = 0; i < n-1; i++)
```

```
    {
        min_idx = i;
        for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
            min_idx = j;

        swap(&arr[min_idx], &arr[i]);
    }
}
```

Ushbu algoritmlarning elementlari soni bir xil bo‘lgan holatda qanday vaqt ichida bajarilishi va sarflangan xotira hajmi haqidagi qiyosiy jadval quyida berilgan.

Sinov o‘tkaziladigan kompyuter quyidagi xususiyatlarga ega: AMD A6-3400M 4x1.4 GHz, 8 Gb operativ xotira, Windows 10 x64 build 10586.36.

2-jadval.

Turli saralash algortimlarining to‘liq saralanmagan massiv uchun ishlash vaqti va ishlatilgan xotira sig‘imi

Saralash usuli		10 ta element uchun		50 ta element uchun		200 ta element uchun		1000 ta element uchun	
		Vaqt (ms)	Xotira (K)	Vaqt (ms)	Xotira (K)	Vaqt (ms)	Xotira (K)	Vaqt (ms)	Xotira (K)
Tanlash bo‘yicha saralash	**Selection sort**	13	510K	37	637	118	854	550	936
Pufakchali saralash	**Bubble sort**	11	524	37	629	116	863	564	932
Joylashtirish bo‘yicha saralash	**Insertion sort**	12	512	38	641	116	849	556	928
Taroqsimon saralash	**Comb sort**	12	505	37	632	117	854	560	936

Qisman tartiblangan massiv (elementlarning yarmi saralangan):

3-jadval.

Turli saralash algortimlarining qisman saralangan massiv uchun ishlash vaqti va ishlatilgan xotira sig‘imi

Saralash usuli		10 ta element uchun		50 ta element uchun		200 ta element uchun		1000 ta element uchun	
		Vaqt (ms)	Xotira (K)	Vaqt (ms)	Xotira (K)	Vaqt (ms)	Xotira (K)	Vaqt (ms)	Xotira (K)
Tanlash bo‘yicha saralash	**Selection sort**	10	501	32	612	113	823	498	942
Pufakchali saralash	**Bubble sort**	9	498	32	601	110	812	509	939
Joylashtirish bo‘yicha saralash	**Insertion sort**	9	482	31	597	112	802	502	920
Taroqsimon saralash	**Comb sort**	10	498	33	563	116	601	505	907

Mavzu yuzasidan savollar:

1. Tartiblash va saralash tushunchalariga ta’rif bering
2. Tanlash bo‘yicha saralash algoritmining murakkabligini baholang
3. Taroqsimon saralash va pufakchali saralash o‘rtasidagi o‘xshashliklarni keltiring
4. Saralash algoritmlari qanday yondashuvlar asosida baholanadi?
5. Yuqorida keltirilganlardan tashqari yana qanday saralash algoritmlarini bilasiz?