SARALASH ALGORITMLARI MOHIYATI VA ULARNING SAMARADORLIGINI BAHOLASH

Raxmonova M.R.
(Muhammad al-Xorazmiy nomidagi TATU assistenti),
Baxtiyorova M.
(Muhammad al-Xorazmiy nomidagi TATU talabasi)

Saralash deb, berilgan obyektlar ketma-ketligini ma`lum mantiqiy tartibda qayta joylashtirish jarayoniga aytiladi. Saralash bir necha ko`rsatkichlarga bog`liq bo`lishi mumkin. Misol uchun maktabda jismoniy tarbiya dars boshida bolalar bo`ylariga qarab safda turishadi. Me`yor topshirish jarayonida esa sinf jurnalidagi familiyalar ketma-ketligiga qarab topshirishadi. Shu yerning o`zida 2 ta saralashdan foydalaniladi. Birinchisi bo`y uzunligi bo`yicha, ikkinchisi sinf jurnalida alvabit tartibida saralanadi.

Saralash jarayoni qanday kechadi? Saralash jarayoni taqqoslashga asoslangan jarayon hisoblanadi. Biror bir ma’lumotni saralash yoki qandaydir qolipga solish juda ham muhim. Sababi, tartibsiz ma’lumotlar bilan ishlash doimo noqulayliklar keltirib chiqaradi va bunday tizim sekin va xatoliklarga moyil bo’ladi.

Tartiblangan ma'lumotlar hammaga yoqadi. Saralash ma'lumotlarni kerakli ketma-ketlikda tartibga solish imkonini beradi, masalan, o'sish yoki kamayish tartibida. Tasavvur qiling-a, siz yirik kompaniyada ishlaysiz va siz xodimlarning ismlarini maoshiga qarab tartiblashingiz kerak. Buning uchun saralash algoritmlari qo'llaniladi.

Quyida saralash algoritmlarining asosiy turlarini ko'rib chiqiladi. Avvalambor saralash algoritmi nima ekanligini aniqlab olaylik.

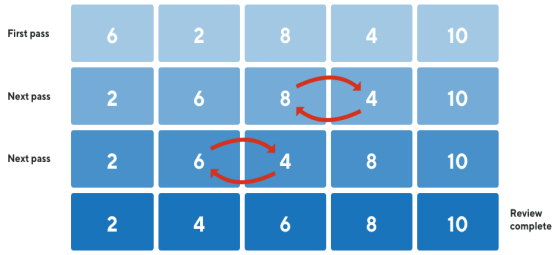
Saralash algoritmlari taqqoslash operatorlari yordamida ro'yxatlar va massivlarda berilgan ma'lumotlarni tartiblab beradi. Bu operatorlar massiv elementlariga qo’llaniladi va ularning ma’lumotlar strukturasidagi tartibini belgilaydi. Misol uchun, quyidagi belgilar (1-rasm) ASCII kodi bo'yicha o'sish tartibida saralangan. Saralash jarayonida elementlar bir-biri bilan taqqoslanadi. ASCII jadvalidagi belgining qiymati qanchalik katta bo'lsa, u ro'yxat boshidan shunchalik uzoqroqda joylashgan bo'ladi. [1]

Dasturlashda turli xil saralash algoritmlar mavjud. Masalani turi mazmuniga qarab turib saralash algoritmlarning biri qo’llaniladi. Keling eng ko’p qo’llaniladigan saralash algoritmlarini ko'rib chiqamiz.

1-rasm. ASCII bo’yicha saralash.

Dasturlashda turli xil saralash algoritmlar mavjud. Masalani turi mazmuniga qarab turib saralash algoritmlarning biri qo’llaniladi. Keling eng ko’p qo’llaniladigan saralash algoritmlarini ko'rib chiqamiz.

Eng ko’p qo’llaniladigan saralash algoritmlaridan biri bu Pufakchali (bubble sort) saralash algoritmidir. Pufakchali saralash algoritmi har bir o'tishda qo'shni elementlarni qayta-qayta taqqoslaydi va almashtiradi (agar kerak bo'lsa) (2-rasm). Bubble Sortning *i*-o'tish joyida (o'sish tartibida) oxirgi *(i-1)* ta elementlar allaqachon tartiblangan bo’ladi va *i*-eng katta element *(N-i)*-pozitsiyaga joylashadi.[2]

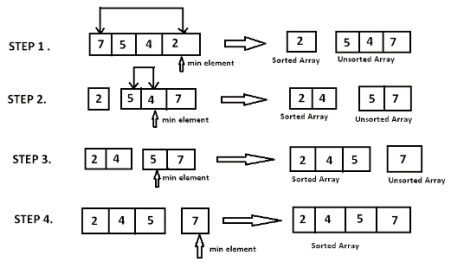


2-rasm. Pufakchali saralash algoritmi.

Pufakchali saralash algoritmi dastur listingi:

```
BubbleSort (Arr, N) // Arr n o’lchamli massiv
{For ( I:= 1 to (N-1) )
{For ( J:= 1 to  (N-I) )
{If ( Arr [J] > Arr[J+1] ) Swap( Arr[j], Arr[J+1] );}}}
```

Keyingi saralash algoritmi bu Tanlash usuli (selection sort). Tanlash usuli - eng kichik elementni tanlaydi va *i*-pog'onaga joylashtiradi (3-rasm). Ushbu algoritm massivni ikki qismga ajratadi: tartiblangan (chap) va tartiblanmagan (o'ng) pastki qator. U tartiblanmagan pastki qatordan eng kichik elementni tanlaydi va shu pastki qatorning birinchi holatiga (o‘sish tartibida) joylashtiradi. Keyingi eng kichik elementni qayta-qayta tanlaydi.[2]



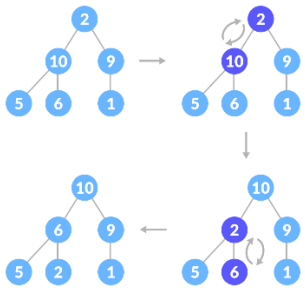
3-rasm. Tanlash usuli (selection sort)

Tanlash usuli dasturlash tilida quyidagicha yozish mumkin:

```
SelectionSort (Arr, N)
{For ( I:= 1 to (N-1) )    {
min_index = I;
For ( J:= I+1 to N )
{If ( Arr [J] < Arr[min_index] )
min_index = J;}
If (min_Index != I)
```

Swap (Arr[I], Arr[min_index]); }}

Uchinchi algoritm bu Yig’ish usuli (heap sort). Algoritm ma'lumotlarni ikkilik daraxt (ikkilik to'p) shaklida tuzadi. Elementlarni joylashtirishning ikkita varianti mavjud - max-heap (ota-onaning qiymati bolalarning qiymatlaridan katta) va min-heap (ota-onaning qiymati bolalarning qiymatlaridan kamroq). Eng katta yoki eng kichik element (turiga qarab) daraxtning ildiziga joylashtiriladi (4-rasm). U to'plamning oxirgi elementi bilan almashtiriladi va massivning oxiriga joylashtiriladi. Uyum hajmi 1 ga kamayadi, shundan so'ng u qayta tiklanadi. Uyum hajmi 1 dan katta bo'lsa, tsikl takrorlanadi.[3]



4-rasm. Yig’ish usuli (heap sort)

Yig’ish usuli dasturlash tilida quyidagicha yozish mumkin:

heapify(array)

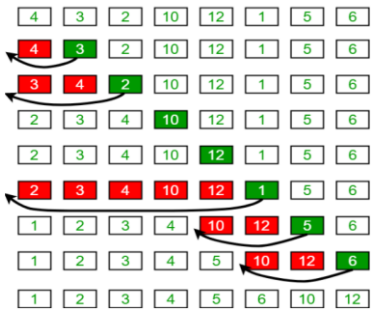
Root = array[0]

Largest = largest(array[0] , array [2 * 0 + 1]. array[2 * 0 + 2])

if(Root != Largest)

Swap(Root, Largest)

Keyingi saralash algoritmi bu Kiritish usulidir (5-rasm insertion sort). Kiritish algoritmi - bu oddiy tartiblash algoritmi bo'lib, u karta o'yinidagi kartalarini saralash usuliga o'xshab ishlaydi. Massiv deyarli tartiblangan va tartiblanmagan qismlarga bo'lingan. Saralanmagan qismdan qiymatlar tanlanadi va tartiblangan qismning to'g'ri joyiga joylashtiriladi.[2]



5-rasm. Kiritish usuli (insertion sort)

Kiritish usulining algoritmi quyidagicha:

InsertionSort (Arr, N) // Arr is an array of size N.

{For (I:= 2 to N) // N elements => (N-1) pass

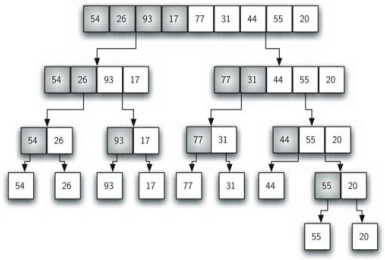
{insert_at = I; // Find suitable position insert_at, for Arr[I]

item = Arr[I]; J=I-1;

While (J ? 1 && item < Arr[J])

```
{Arr[J+1] = Arr[J];
// insert_at = J;
J--;}
insert_at = J+1;
Arr[insert_at] = item; // Arr[J+1] = item;}}}
```

Yana bir saralash usullaridan biri bu - Birlashtirish usuli (6-rasm Merge sort)dir. Bu usul “Bo‘l va zabt et” tamoyiliga amal qiladi, unga ko‘ra ma’lumotlar massivi teng qismlarga bo‘linadi, ular alohida tartiblanadi. Ular birlashgandan so'ng, natijada tartiblangan massiv hosil bo'ladi.[5]



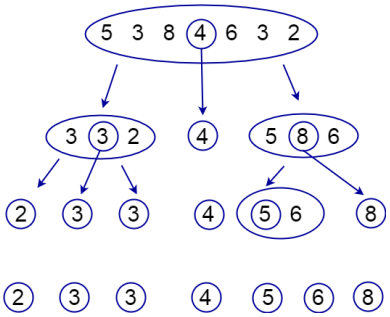
6-rasm. Birlashtirish usuli (Merge sort).

Birlashtirish usulining dastur listingi quyidagi ko’rinishda yozish mumkin:

```
void Merge(int *a, int low, int high, int mid)
{    int i, j, k, temp[high-low+1];
i = low;
k = 0;
j = mid + 1;
while (i <= mid && j <= high)
{if (a[i] < a[j]){temp[k] = a[i];
k++;
  i++;}
    else
    {temp[k] = a[j];
    k++;
    j++;
    }}
  while (i <= mid)
  {temp[k] = a[i];
    k++;
    i++;        }
  while (j <= high)
  {temp[k] = a[j];
k++;
j++;
}
```

```
for (i = low; i <= high; i++)
{a[i] = temp[i-low];}
}
void MergeSort(int *a, int low, int high)
{int mid;
if (low < high)
{mid=(low+high)/2;
MergeSort(a, low, mid);
MergeSort(a, mid+1, high);
 Merge(a, low, high, mid);} }
```

Tanishadigan yana bir saralash algoritmi bu Tez saralash usuli (7-rasm quick sort)dir. Eng tezkor tartiblash algoritmlaridan biri hisoblanadi. Birlashtirish usuli kabi, u bo'lish va egallash asosida ishlaydi.[6]



7-rasm. Tez saralash usuli.

Bu algoritmini ham dastur listingi beriladi va u quyidagi ko’rinishda bo’ladi.

```
quickSort(arr[], low, high)
{if (low < high)
{pi = partition(arr, low, high);
quickSort(arr, low, pi - 1);  // pi dan oldin
quickSort(arr, pi + 1, high); // pi dan keyin
}}
```

Mana 6 turdagi saralash algoritmlari bilan tanishib chiqdik.

Saralash algoritmlarining samaradorligini baholashda ikki mezon bo’yicha fazo va vaqt murakkabligi hisoblanadi.[1]

Fazoviy murakkablik – bu algoritmni bajarish uchun foydalaniladigan xotira hajmi bilan ifodalanadi. Fazoviy murakkablik yordamchi xotira va kirish xotirasini o'z ichiga oladi. Yordamchi xotira - kirish ma'lumotlariga qo'shimcha ravishda algoritm egallagan qo'shimcha joy. Algoritmlarning fazoviy murakkabligini hisoblashda hisobga olinadi.[1]

Vaqtning murakkabligi – bu kirish ma'lumotlarini hisobga olgan holda algoritm vazifani bajarishga sarflagan vaqtni bildiradi. Uni quyidagi belgilar yordamida ifodalash mumkin:

Omega belgisi (Ω)

Katta "O" belgisi (O)

Teta belgisi (Θ)

Quyidagi 1-Jadvalda yuqorida keltirilgan algoritmlarning murakkabliklari taxminiy ko'rsatilgan.[1]

1-Jadval

Saralash algoritmi	Eng yomon holatda ishlash vaqti	O'rtacha ish vaqti	Eng yaxshi holatda ish vaqti	Fazoviy murakkablik
Pufakchali tartiblash	n^2	n^2	n	bitta
Tanlash usuli	n^2	n^2	n^2	bitta
Tez tartiblash	n^2	$n\log n$	$n\log n$	$n\log n$
Yig’ish usuli	$n\log n$	$n\log n$	$n\log n$	bitta
Kiritish usuli	n^2	n^2	n	bitta
Birlashtirish usuli	$n\log n$	$n\log n$	$n\log n$	n

Xulosa qilib aytish mumkinki, har bir saralash algoritmi o'ziga xos vaqt va makon murakkabligiga ega. Vazifalarga qarab, taqdim etilgan algoritmlarning biridan foydalanish mumkin. Lekin mening sub'ektiv fikrimcha, tez tartiblash eng yaxshi algoritmdir. U asosiy tayanch elementni tanlash imkonini beradi va massivni 3 qismga ajratadi: kichik, teng va tayanchdan katta.

FOYDALANILGAN ADABIYOTLAR RO’YHATI:

1. https://proglib.io/p/sravnenie-6-algoritmov-sortirovki-puzyrkom-vyborom-kuchey-vstavkami-sliyaniem-i-bystraya-2022-02-08
2. https://www.geeksforgeeks.org/comparison-among-bubble-sort-selection-sort-and-insertion-sort/
3. https://www.geeksforgeeks.org/heap-sort/#:~:text=Heap%20sort%20is%20a%20comparison,process%20for%20the%20remaining%20elements
4. https://www.geeksforgeeks.org/insertion-sort/
5. https://www.geeksforgeeks.org/merge-sort/
6. https://www.geeksforgeeks.org/quick-sort/