4-§ Birlashtirib saralash algoritmlari

Merge sort algoritmi. Birlashtirib saralash (Merge sort) – tartiblashning tezkor bajariladigan algoritmlaridan biri. Ushbu tartiblash “bo‘lib tashla va hukmronlik qil” prinsipining yaxshi namunasidir. Birinchidan, vazifa bir nechta kichik topshiriqlarga bo‘linadi. Keyin ushbu vazifalar rekursiv chaqiruv yordamida yoki to‘g‘ridan-to‘g‘ri ularning hajmi yetarlicha kichik bo‘lsa hal qilinadi. Nihoyat, ularning yechimlari birlashtirilib, asl muammoning echimi olinadi.

Algoritmning bajarilishi. Saralash muammosini hal qilish uchun uch bosqich quyidagicha bo‘ladi:

1. Saralanadigan massiv taxminan bir xil o‘lchamdagi ikki qismga bo‘linadi;
2. Olingan qismlarning har biri alohida saralanadi (masalan, xuddi shu algoritm bo‘yicha saralanadi);
3. Yarim kattalikdagi ikkita saralangan massivlar birlashtiriladi.

Bu eng mashhur saralash algoritmlaridan biri bo‘lib, rekursiv algoritmlarni yaratishda ishonchni rivojlantirishning ajoyib usuli hisoblanadi.

“Bo‘lib tashla va hukmronlik qil” strategiyasi. “Bo‘lib tashla va hukmronlik qil” strategiyasi yordamida muammoni qismiy jarayonlarga ajratamiz. Har bir kichik topshiriq uchun yechimga ega bo‘lsak, pastki vazifalarni yechish uchun pastki vazifalardan olingan natijalarni "birlashtiramiz".

Aytaylik, biz A massivni saralashni xohladik. Kichik vazifa bu p indeksidan boshlanib, r indeksida tugagan, A [p..r] bilan belgilangan kichik qismini ajratishdir.

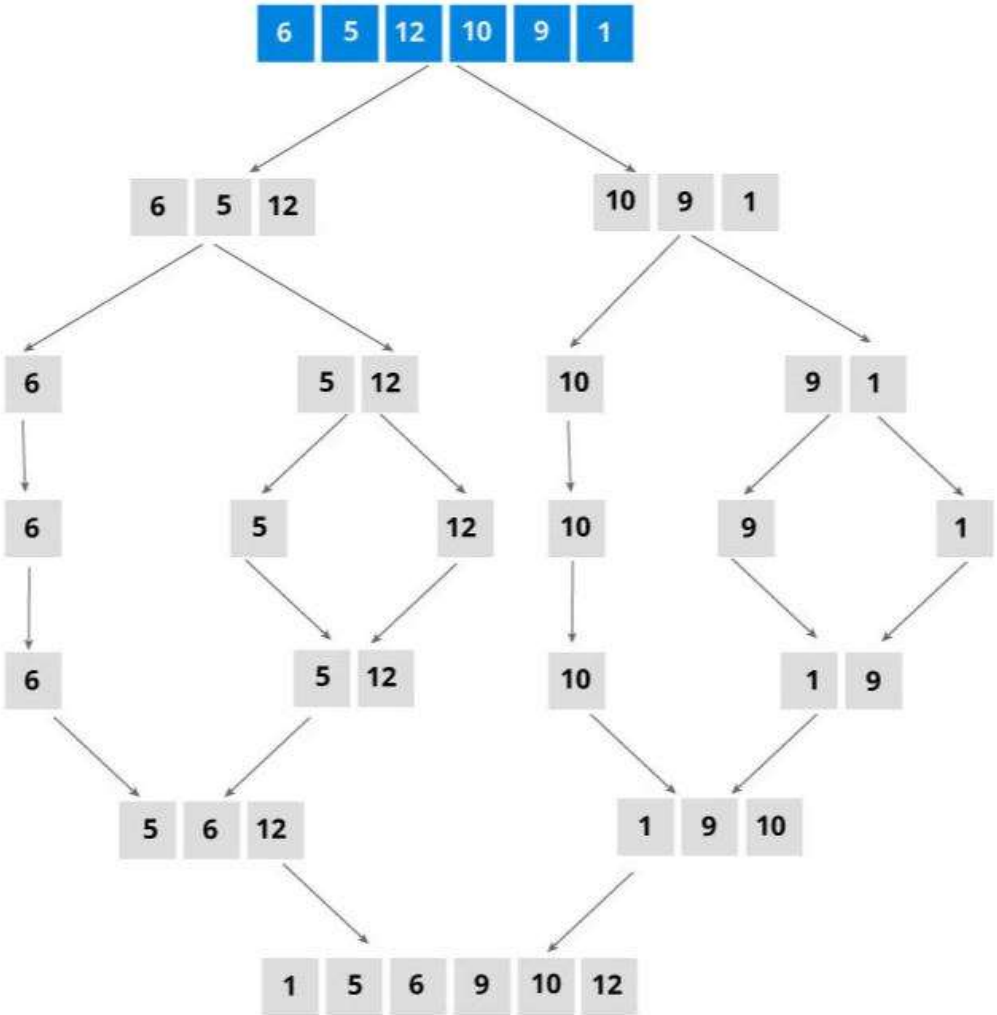
“Bo‘lib tashlash”. Agar q qiymati p va r orasida bo‘lsa, biz A [p..r] massivni ikkita A [p..q] va A [q + 1, r] kichik massivlarga bo‘lishimiz mumkin.

“Hukmronlik qilish”. “Hukmronlik qilish” bosqichida biz ikkala A [p..q] va A [q + 1, r] kichik massivlarni saralashga harakat qilamiz. Agar hali ham boshlang‘ich darajaga yetib bormagan bo‘lsak, yana ikkala quyi qismni ajratib, ularni saralashga harakat qilamiz.

Birlashtirish bosqichi. Birlashtirish bosqichi asosiy pog‘onaga yetib borganida va biz A [p..r] massivi uchun ikkita tartiblangan A [p..q] va A [q + 1, r] kichik massivlarni olsak, natijalarni A [p..r] massiviga birlashtiramiz. Bu ikkita tartiblangan A [p..q] va A [q + 1, r] massivlarning birlashmasidir (12-rasm).

Birlashtirib saralash algoritmi. MergeSort funksiyasi massivni ketma-ket ikki qismga ajratadi, biz 1-darajali ichki massivda MergeSort-ga o‘tishga harakat qiladigan bosqichga yetguncha ya’ni p == r bo‘lguncha.

Shundan so‘ng, birlashtirish funksiyasi ishga tushadi, bu tartiblangan massivlarni butun massiv birlashguncha katta massivlarga birlashtiradi.

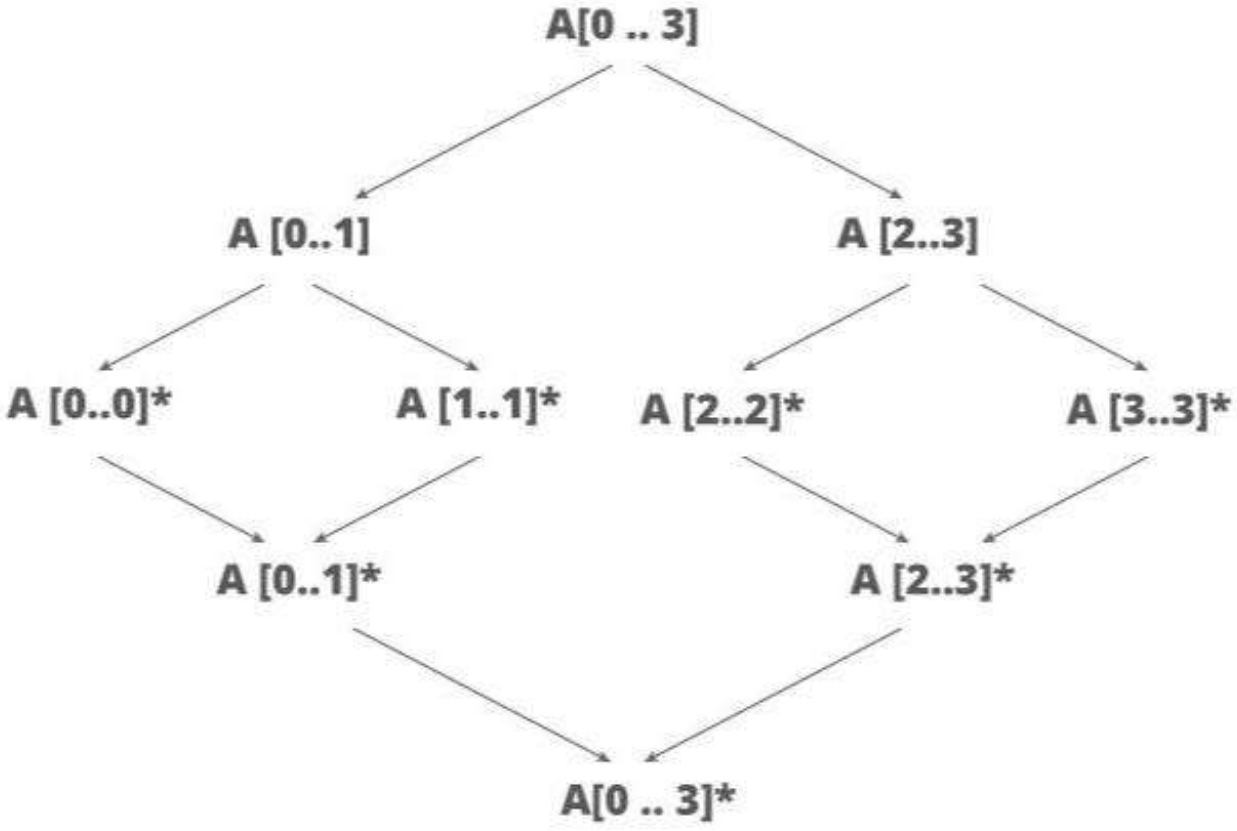


12-rasm. Merge Sort algoritmining ishlash prinsipi

```
1. MergeSort(A, p, r)
2.     If p > r
3.         return;
4.     q = (p+r)/2;
5.     mergeSort(A, p, q)
6.     mergeSort(A, q+1, r)
7.     merge(A, p, q, r)
```

Butun massivni saralash uchun MergeSort (A, 0, length (A) -1) ga murojaat qilishimiz kerak.

13-rasmda ko‘rsatilgandek, birlashtirib saralash algoritmi 1 elementli massivning asosiy holatiga kelgunimizcha massivni rekursiv ravishda ikkiga bo‘ladi. So‘ngra birlashtirish funksiyasi saralangan ichki massivlarni tanlaydi va butun qatorni asta-sekin saralash uchun ularni birlashtiradi.



13-rasm. Merge Sort algoritmida massivni qismlarga bo‘lish jarayoni

Algoritmning eng muhim qismi bu "birlashtirish" bosqichidir. Birlashish bosqichi - ikkita katta ro‘yxat (massiv) yaratish uchun ikkita tartiblangan ro‘yxatni (massivlarni) birlashtirish bo‘yicha oddiy muammoning yechimi.

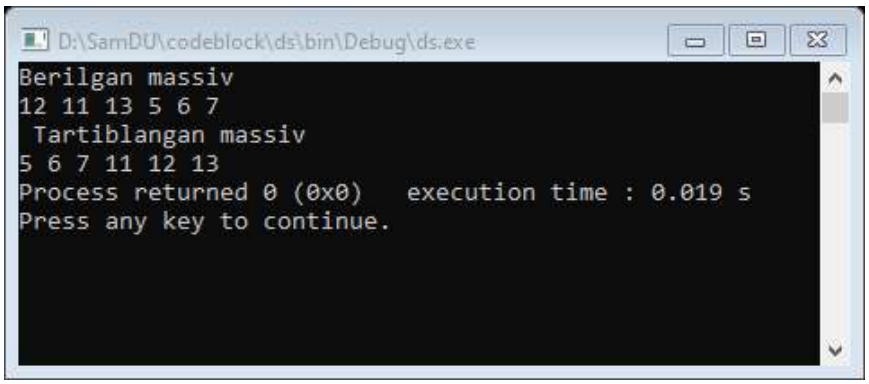
Ikkinchi massivda boshqa elementlar qolmaganligi va ikkala massiv ham ishga tushirilganda saralanganligini bilganimiz uchun qolgan massivlarni to‘g‘ridan-to‘g‘ri birinchi massivdan nusxalashimiz mumkin.

Birlashtirib saralash algoritmi uchun dastur kodi

```
#include <iostream>
using namespace std;

// Array[] ikkita ichki massivni birlashtiradi.
//Birinchi ichki massiv - Array[l..m]
//  Ikkinchi ichki massiv Array[m+1..r]
void merge(int Array[], int l, int m, int r)
{
        int n1 = m - l + 1;
        int n2 = r - m;

        // Vaqtinchalik massivlarni yaratish
        int L[n1], R[n2];

        // Ma’lumotlarni vaqtinchalik L[] va R[] massivlariga nusxalash
        for (int i = 0; i < n1; i++)
                L[i] = Array[l + i];
        for (int j = 0; j < n2; j++)
                R[j] = Array[m + 1 + j];

        // Vaqtinchalik massivlarni yana arr [l..r] ga birlashtirish.

        // Birinchi ichki massivning boshlang‘ich ko‘rsatkichi
        int i = 0;
```

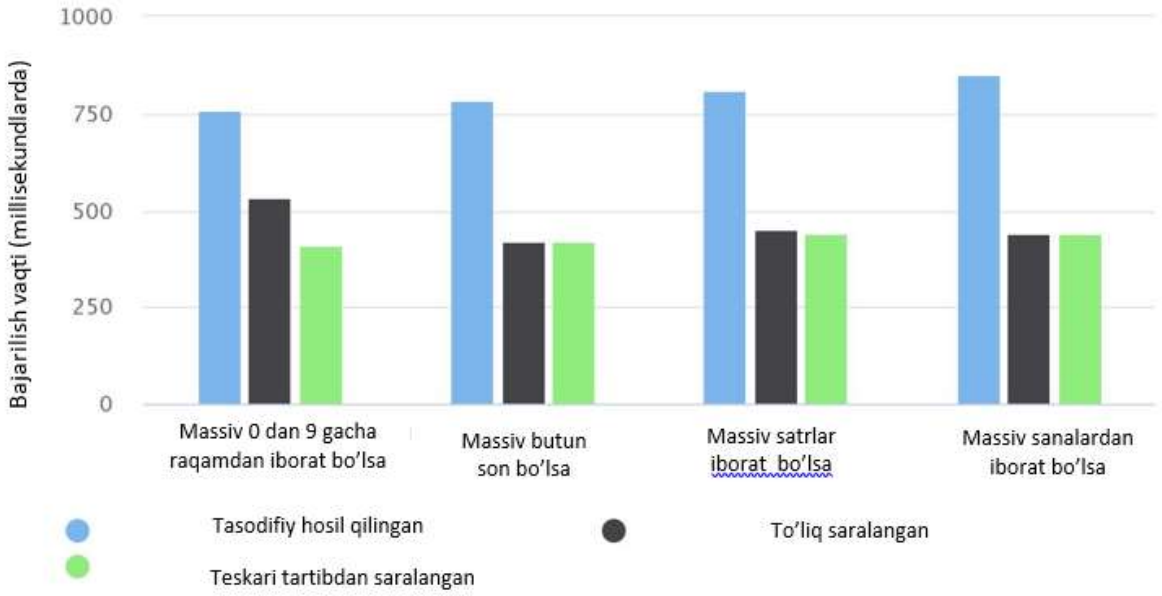
```
        // Ikkinchi kichik massivning boshlang‘ich ko‘rsatkichi
        int j = 0;

        // Birlashtirilgan ichki massivning dastlabki ko‘rsatkichi
        int k = l;

        while (i < n1 && j < n2) {
                if (L[i] <= R[j]) {
                        Array[k] = L[i];
                        i++;
                }
                else {
                        Array[k] = R[j];
                        j++;
                }
                k++;
        }

        // L [] ning qolgan elementlarini nusxalash,
        //agar mavjud bo‘lsa
        while (i < n1) {
                Array[k] = L[i];
                i++;
                k++;
        }

        // Agar mavjud bo‘lsa, R [] ning
        //qolgan elementlarini nusxalash
        while (j < n2) {
                Array[k] = R[j];
                j++;
                k++;
        }
}

// l chap indeks uchun,
//r esa tartiblangan ichki massivning o‘ng indeksidir
void mergeSort(int Array[],int l,int r){
```

```
        if(l>=r){
                return;//rekursiv ravishda qaytadi
        }
        int m =l+ (r-l)/2;
        mergeSort(Array,l,m);
        mergeSort(Array,m+1,r);
        merge(Array,l,m,r);
}

// Massivni chop etish funksiyasi
void printArrayay(int A[], int size)
{
        for (int i = 0; i < size; i++)
                cout << A[i] << " ";
}
int main()
{
        int Array[] = { 12, 11, 13, 5, 6, 7 };
        int Array_size = sizeof(Array) / sizeof(Array[0]);

        cout << "Berilgan massiv \n";
        printArrayay(Array, Array_size);

        mergeSort(Array, 0, Array_size - 1);

        cout << "\n Tartiblangan massiv \n";
        printArrayay(Array, Array_size);
        return 0;
}
```



14-rasm. Merge Sort algoritmi dastur natijasi

Birlashtirib saralash algortimini baholash. Algoritmning murakkabligini taxmin qilaylik. Har qanday rekursiv funksiya chaqiruvi daraxtga o‘xshaydi (Izoh: “Daraxtlar” haqida keyingi ma’ruzalarda to‘xtalib o‘tiladi). Bunday daraxtni rekursion daraxt deb ham atashadi. Daraxtning har bir darajasi bir yoki bir nechta funksiya chaqiruvlarini aks ettiradi. Shoxlari bo‘lmagan daraxt tugunlari rekursiyani tugatadigan funksiya chaqiruvlarini anglatadi. Birlashtirish tartibida daraxtning balandligi $\log_2 n$ ga teng, chunki har bir qadamda boshlang‘ich massiv n/2 uzunlikdagi ikkita ichki massivga bo‘linadi. Ajratishdan so‘ng, birlashtirish operatsiyasi amalga oshiriladi. Birlashtirish jarayoni n taqqoslashni, navbati bilan *log*n marta, ya’ni daraxtning har bir darajasi uchun 1 marta takrorlashni talab qiladi. Keyin algortim asimptotikasi O (nlogn) bo‘ladi.



15-rasm. Merge Sort algoritmining turli xil tiplar uchun ishlash vaqti (elementlar soni 50000 ta)

1. Merge sort algoritmining murakkabliklarini baholang
2. Merge sort algoritmida “Bo‘lib tashlash”da nimalarga e’tibor berish kerak