4-AMALIY ISH **Saralash masalasini formal qo‘yilishi. Saralashning qat’iy va yashilangan usullari**

Ishning maqsadi: Saralash algoritmlarini o’rganish va amalda qo’llash.

Ish tartibi:
- Nazariy qismdagi ma’lumotlarni o’rganib chiqish.
- Amaliy qismda keltirilgan amaliyot ishini bajarish.
- *Topshiriqlar* bo’limida keltirilgan masalalarni yechish
- Amalga oshirilgan ish bo’yicha hisobot tayyorlash va topshirish

NAZARIY QISM VA AMALIY QISM

SARALASH USULLARI

Saralash to’plamdagi elementlarni ishlashga qulay ko’rinishga olib keladi. Agar sonli massivdagi sonlar kamayish tartibida saralansa uning eng birinchi elementi doimo eng kattasi bo’lib hisoblanadi. Shu sababli ma’lumotlarni saralangan formada saqlagan ma’qul.

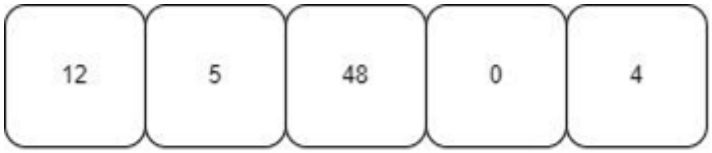
Ushbu amaliy ishda quyidagi saralash usullari realizatsiyasi qarab chiqiladi:

- Pufakchali saralash (Bubble sort)
- Tanlash orqali saralash (Selection sort)
- O’rniga qo’yish orqali saralash (Insertion sort)
- Tez saralash (Quick sort)
- Birlashtirish orqali saralash (Merge sort)
- Shell saralash usuli (Shell sort)

1. PUFAKCHALI SARALASH

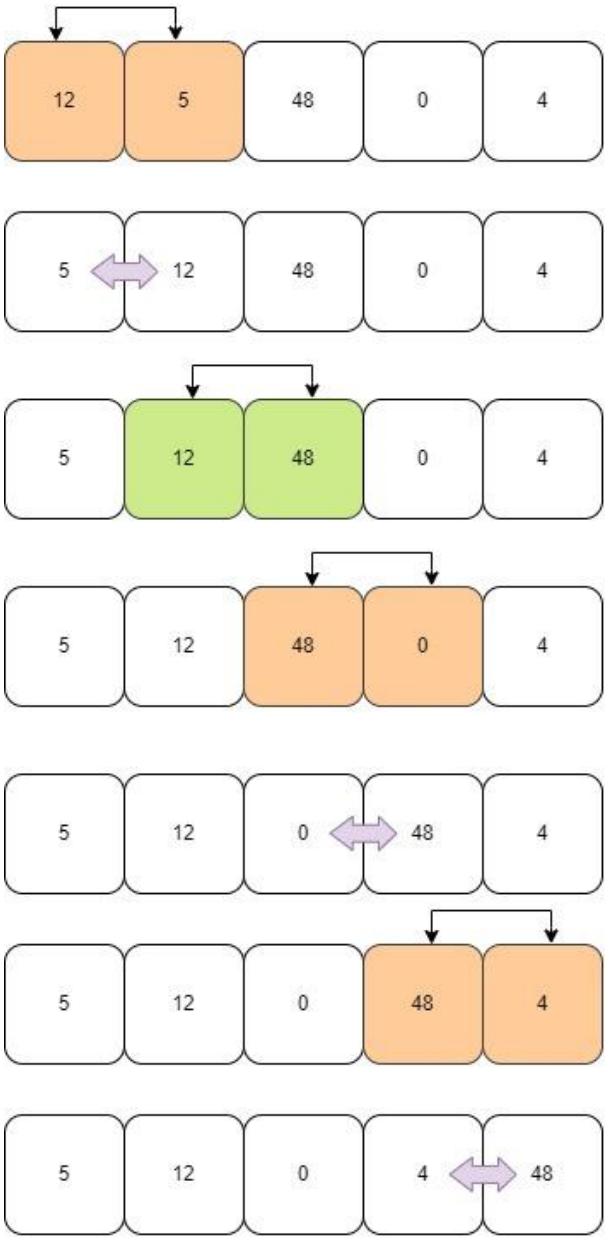
Bu saralash usulida **har bir element** keyingi element bilan solishtiriladi. Agar bu elementlar kerakli tartib joylashmagan bo’lsa ular o’rni almashtiriladi. Har bir iteratsiya oxirida **eng katta yoki kichik** element ro’yxat oxiriga joylashtiriladi.

Quyidagi massiv berilgan.



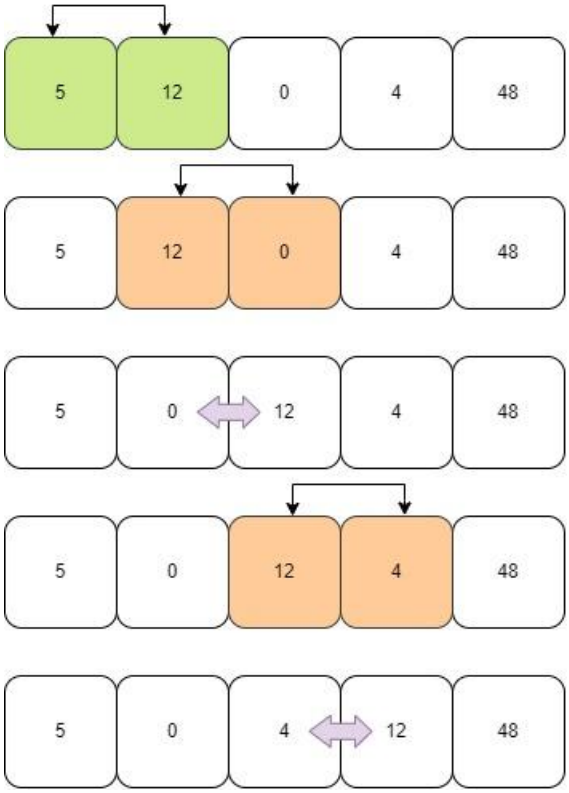
1-iteratsiya.

Massivni o’sish tartibida saralash kerak bo’lsin. Har bir element qo’shni element bilan solishtiriladi. Agar chap element o’ng elementdan katta bo’lsa u holda ular almashtiriladi. Rasmda sariq rangda almashtiriladigan elementlar ko’rsatilgan.



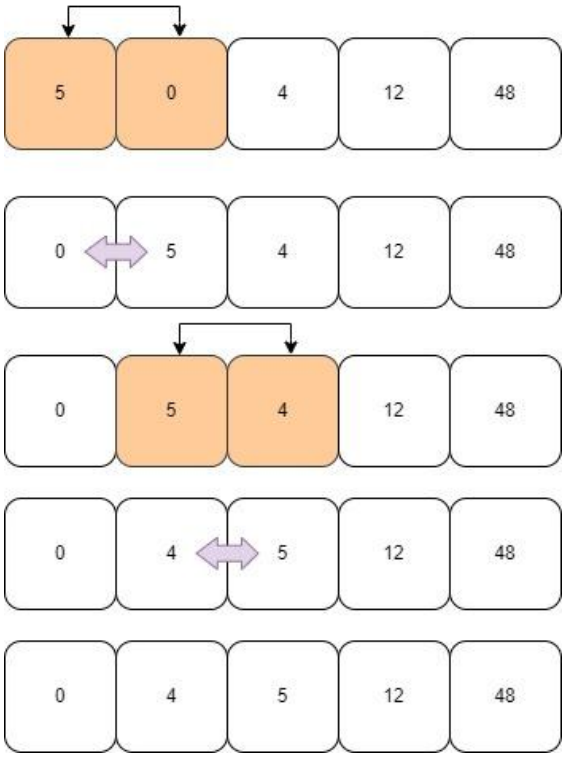
Eng katta element ro’yxat oxirida joylashgan.

2-iteratsiya.

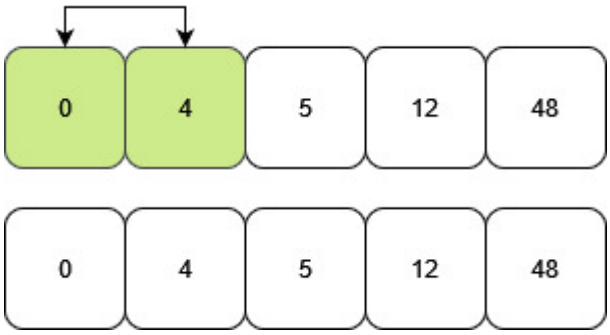


Eng katta element ro’yxat oxirida joylashganligi sababli iteratsiya 4 chi indeksgacha amalga oshirish yetarli.

3-iteratsiya.



4-iteratsiya.



4 iteratsiyadan keyin saralangan ro'yxatni olamiz.

Algoritm murakkabligi

- eng yaxshi holatda: O(n)
- o'rtacha holatda $O(n^2)$
- eng yomon holatda $O(n^2)$

Dastur kodi

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

void bubbleSort(int list[], int listLength)
{
	while(listLength--)
	{
		bool swapped = false;

		for(int i = 0; i < listLength; i++)
		{
			if(list[i] > list[i + 1])
			{
				swap(list[i], list[i + 1]);
				swapped = true;
			}
		}

		if(swapped == false)
			break;
	}
}

int main()
{
	int list[5] = {3,19,8,0,48};
	cout << "Input array ..." << endl;
	for(int i = 0; i < 5; i++)
		cout << list[i] << '\t';
	cout << endl;

	bubbleSort(list, 5);

	cout << "Sorted array ..." << endl;
	for(int i = 0; i < 5; i++)
		cout << list[i] << '\t';
```

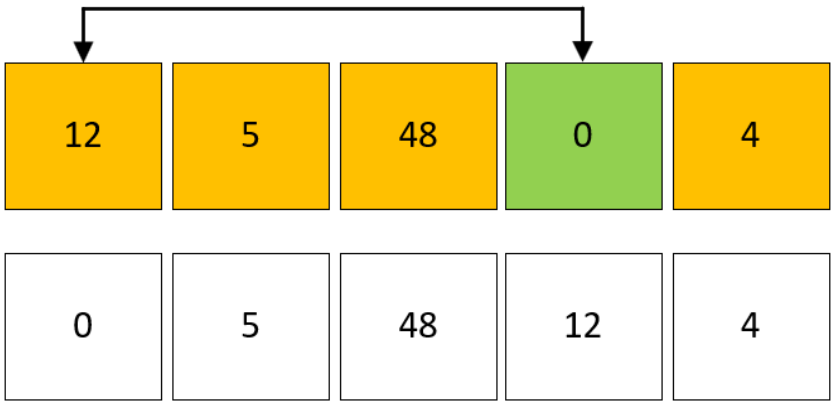
```
        cout << endl;
}
```

2. TANLASH ORQALI SARALASH

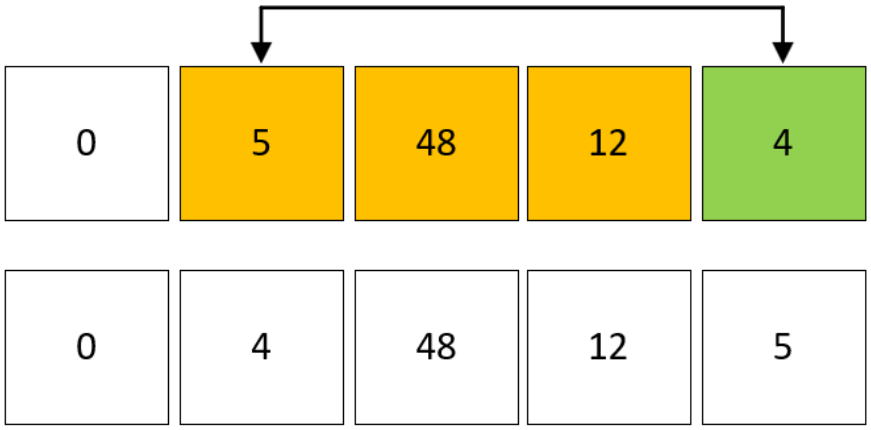
Ro’yxatda eng kichik element qidiriladi va uni joriy indeks bilan almashtiriladi.

1-iteratsiya.

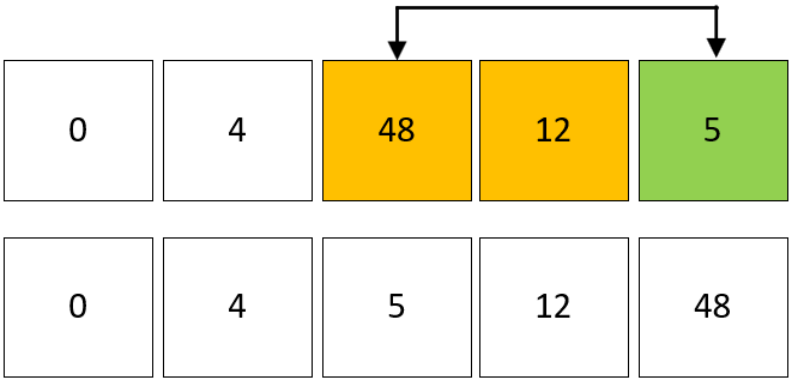
Yashil rangda eng kichik element ko’rsatilgan. Sariq rangda qarab chiqiladigan elementlar ko’rsatilgan.



2-iteratsiya.



3-iteratsiya.



Algoritm murakkabligi

- xohlagan holatda $O(n^2)$

Dastur kodi

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

int findSmallestPosition(int list[], int startPosition, int listLength)
{
	int smallestPosition = startPosition;

	for(int i = startPosition; i < listLength; i++)
	{
		if(list[i] < list[smallestPosition])
			smallestPosition = i;
	}
	return smallestPosition;
}

void selectionSort(int list[], int listLength)
{
	for(int i = 0; i < listLength; i++)
	{
		int smallestPosition = findSmallestPosition(list, i,
listLength);
		swap(list[i], list[smallestPosition]);

	}
	return;
}

int main ()
{
	int list[5] = {12, 5, 48, 0, 4};

	cout << "Input array ..." << endl;
	for(int i = 0; i < 5; i++)
		cout << list[i] << " ";
	cout << endl;
```

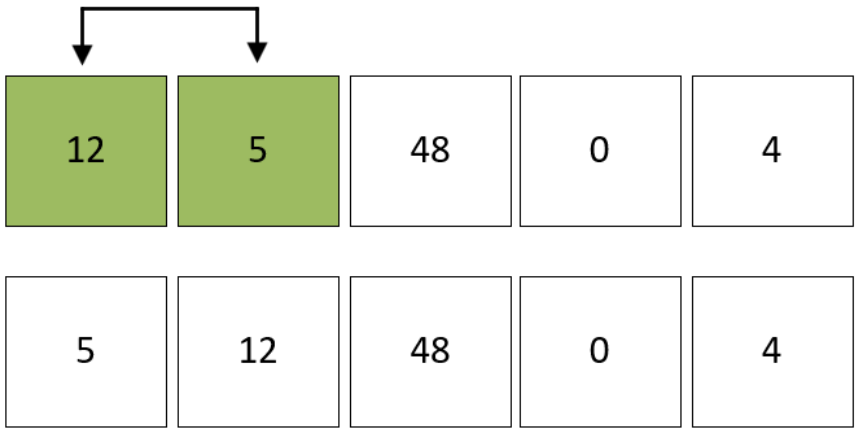
```
        selectionSort(list, 5);

        cout << "Sorted array ..." << endl;
        for(int i = 0; i < 5; i++)
                cout << list[i] << " ";
        cout << endl;
}
```

3. O’RNIGA QO’YISH ORQALI SARALASH

Bu usul ikkinchi elementda boshlanadi. Elementdan oldingi qo’shni elementlar solishtirib chiqiladi.

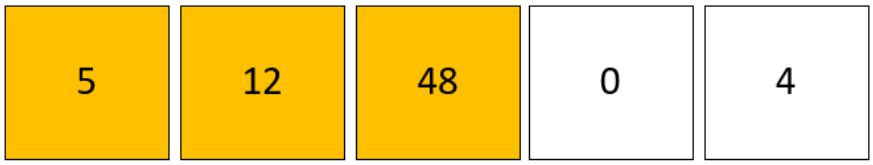
1-iteratsiya.

2 chi elementdan boshlab tekshiriladi. 1 va 2 elementlar almashtiriladi.



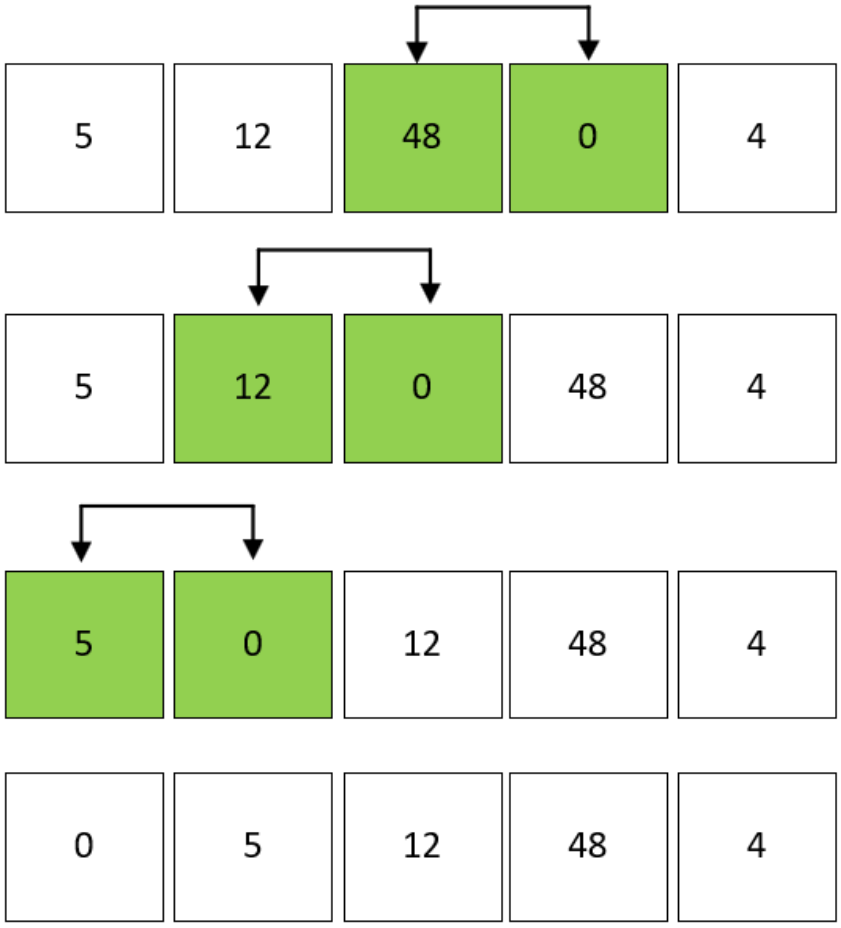
2-iteratsiya.

3 chi elementdan boshlab tekshiriladi. Element to’g’ri ketma-ketlikda bo’lgani uchun almastirish bajarilmaydi.



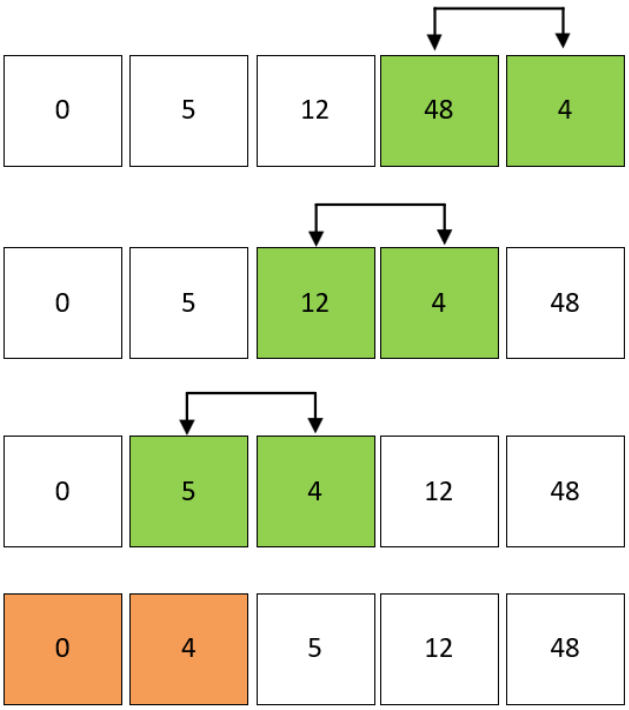
3-iteratsiya.

Ushbu iteratsiyada 3ta almashtirish amalga oshiriladi.



4-iteratsiya.

Ushbu iteratsiyada 3ta almashtirish amalga oshiriladi.



Va saralash tugatiladi.

Algoritm murakkabligi

- eng yaxshi holatda O(n)
- eng yomon holatda $O(n^2)$

Dastur kodi

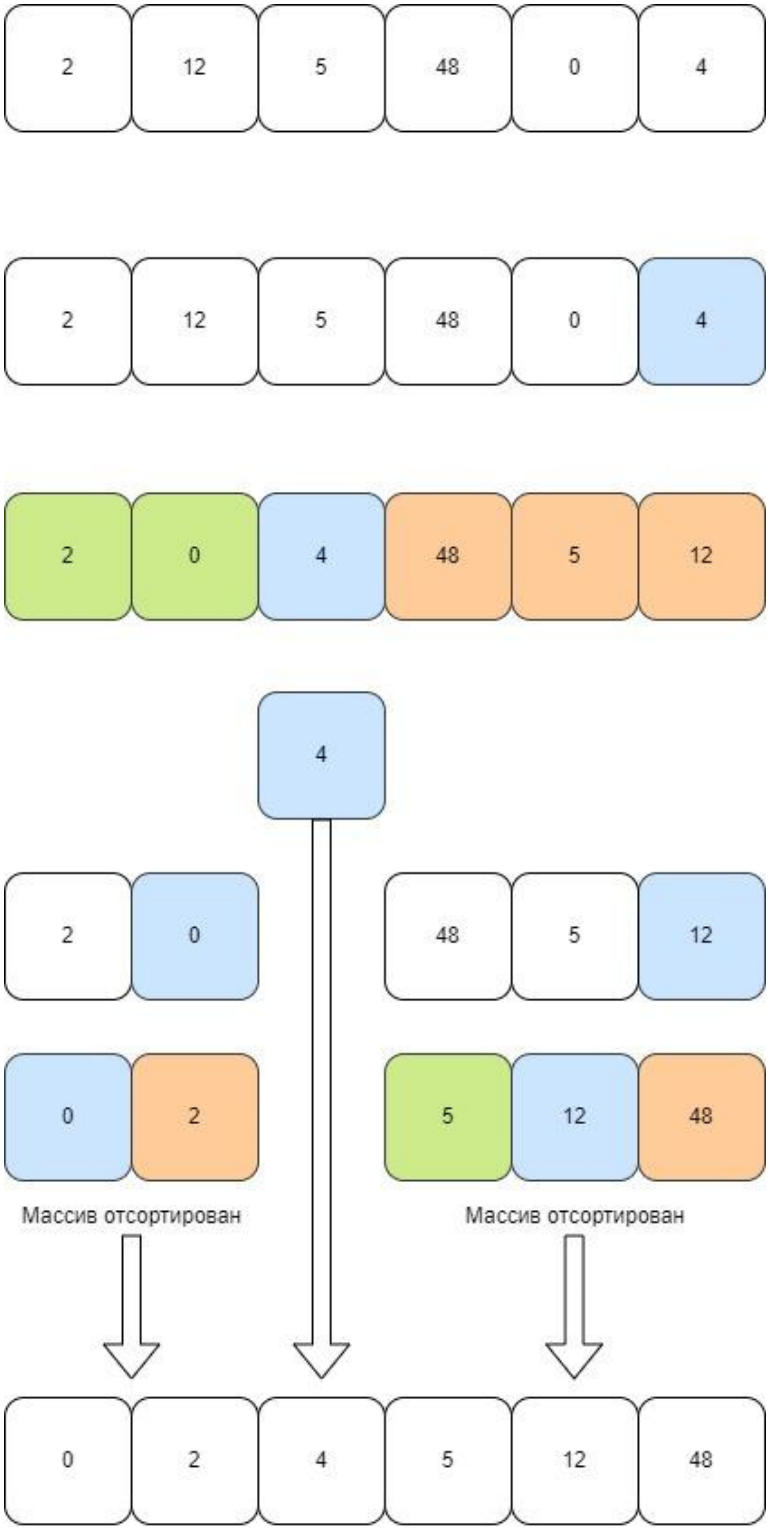
```
#include <iostream>
using namespace std;

void insertionSort(int list[], int listLength)
{
        for(int i = 1; i < listLength; i++)
        {
                int j = i - 1;
                while(j >= 0 && list[j] > list[j + 1])
                {
                        swap(list[j], list[j + 1]);

                        j--;
                }
        }
}

int main()
{
        int list[5]={12,5,48,0,4};
        cout<<"Input array ...\n";
        for (int i = 0; i < 5; i++)
        {
```

```
            cout<<list[i]<<" ";
        }
        cout<<endl;

        insertionSort(list, 5);

        cout<<"\n\nSorted array ... \n";
        for (int i = 0; i < 5; i++)
        {
            cout<<list[i]<<" ";
        }
        cout<<endl;

        return 0;
}
```

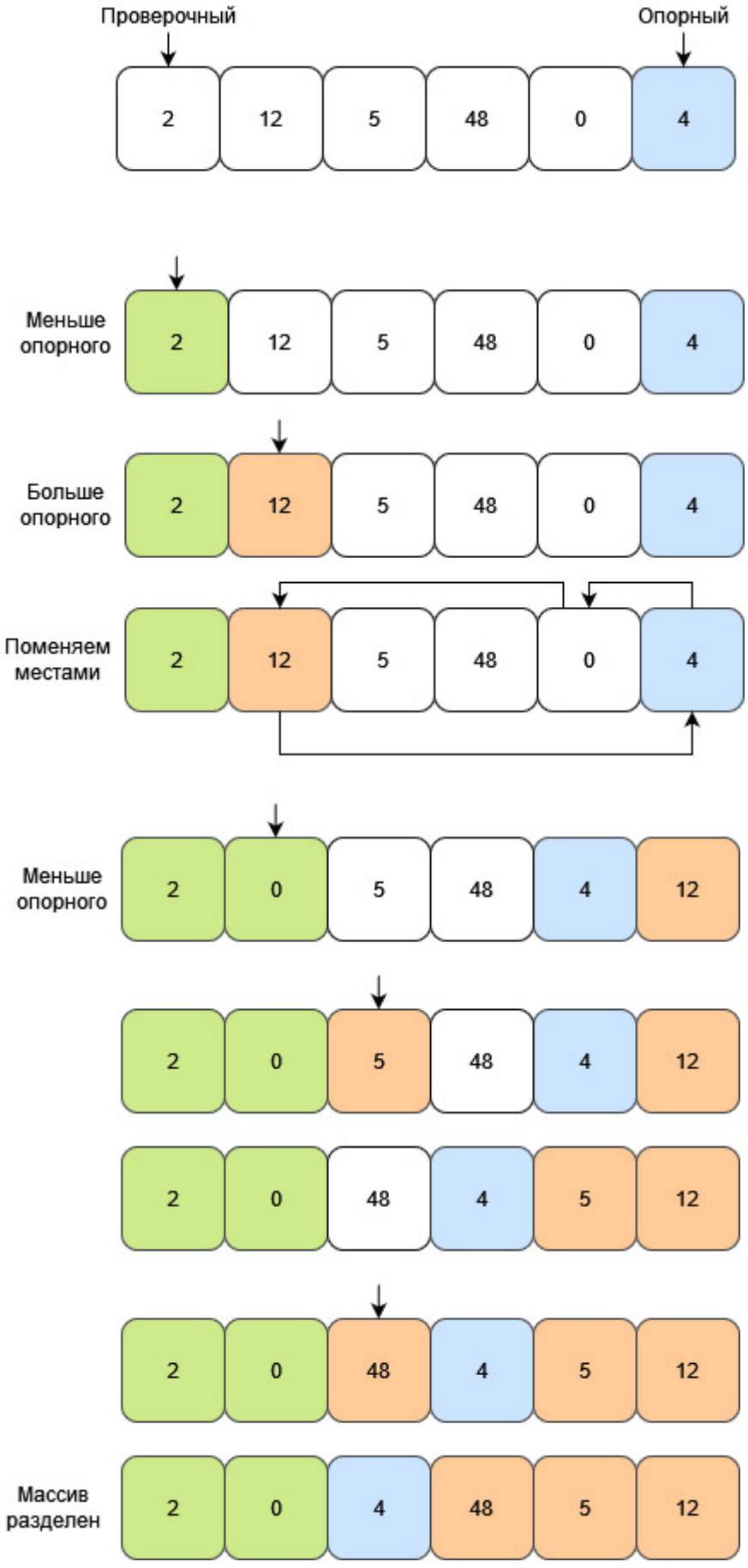
4. TEZ SARALASH

Bu usulda masala kichik masalalarga bo’linadi.

Birinchi navbatda tayanch element tanlab olinadi. Bun eng oxirgi element, ko’k rangda berilgan. Ekin tayanch elementdan kichik elementlar chap tarafga, katta elementlar o’ng tarafga ko’chiriladi.



Elementlarni ko’chirish tartibi quyidagi rasmda keltirilgan.



Algoritm murakkabligi

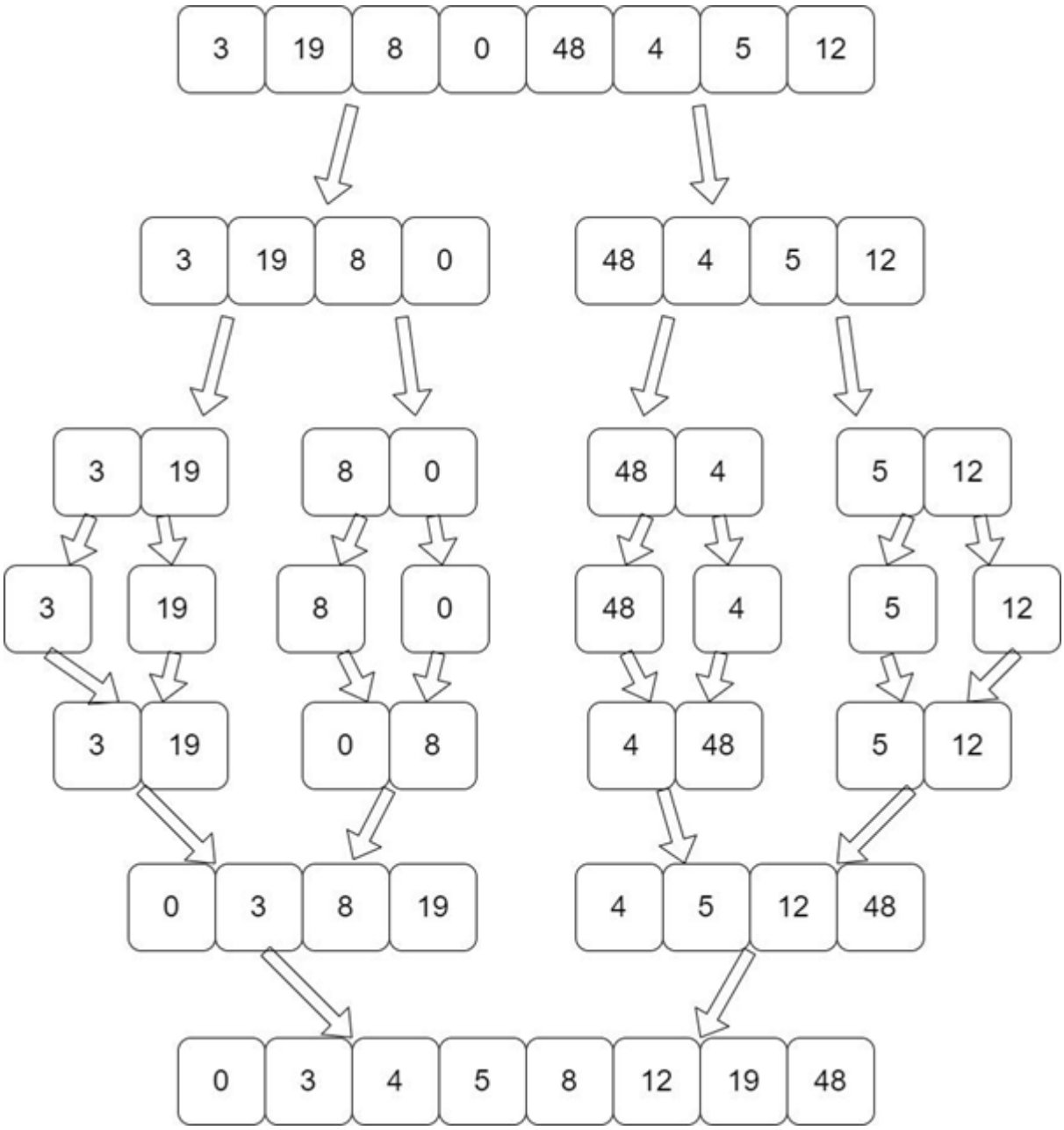
- eng yaxshi holatda O(n*logn)
- eng yomon holatda $O(n^2)$

Dastur kodi

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

int partition(int a[], int start, int end)
{
    int pivot = a[end];
    int pIndex = start;

    for (int i = start; i < end; i++)
    {
        if (a[i] <= pivot)
        {
            swap(a[i], a[pIndex]);
            pIndex++;
        }
    }

    swap (a[pIndex], a[end]);

    return pIndex;
}

void quicksort(int a[], int start, int end)
{
    if (start >= end) {
        return;
    }

    int pivot = partition(a, start, end);

    quicksort(a, start, pivot - 1);
    quicksort(a, pivot + 1, end);
}

int main()
{
    int a[] = { 2, 12, 5, 48, 0, 4 };
    int n = sizeof(a)/sizeof(a[0]);

    quicksort(a, 0, n - 1);

    for (int i = 0; i < n; i++) {
        cout << a[i] << " ";
    }

    return 0;
}
```

5. BIRLASHTIRISH ORQALI SARALASH

Bu usulda massivni ikki teng massivga bo'linadi. Bo'lish davom ettiriladi va saralangan holda qayta yig'iladi.



Algoritm murakkabligi

- barcha holatda O(n*logn)

Dastur kodi

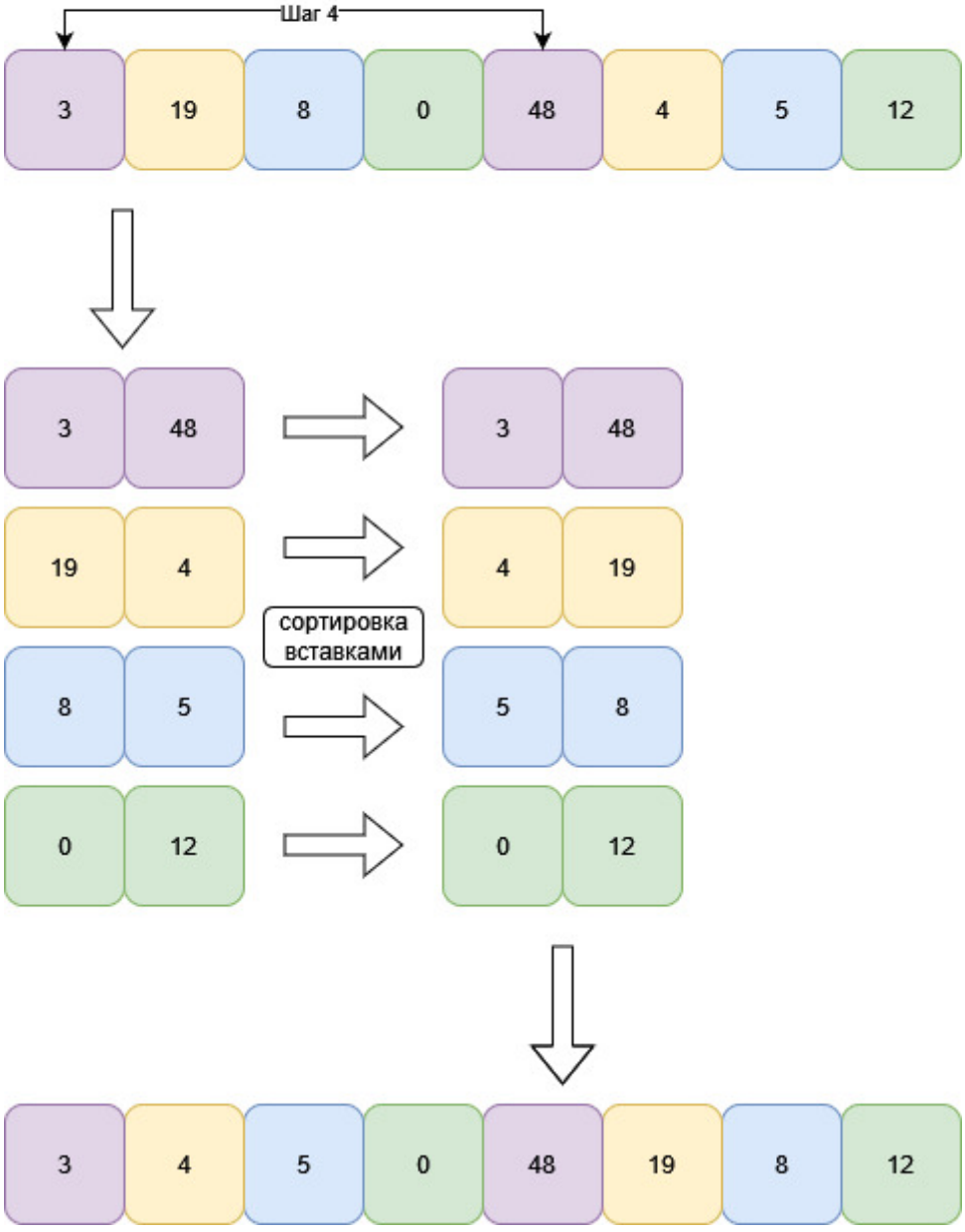
```
#include <iostream>
using namespace std;

void merge(int list[],int start, int end, int mid);

void mergeSort(int list[], int start, int end)
{
```

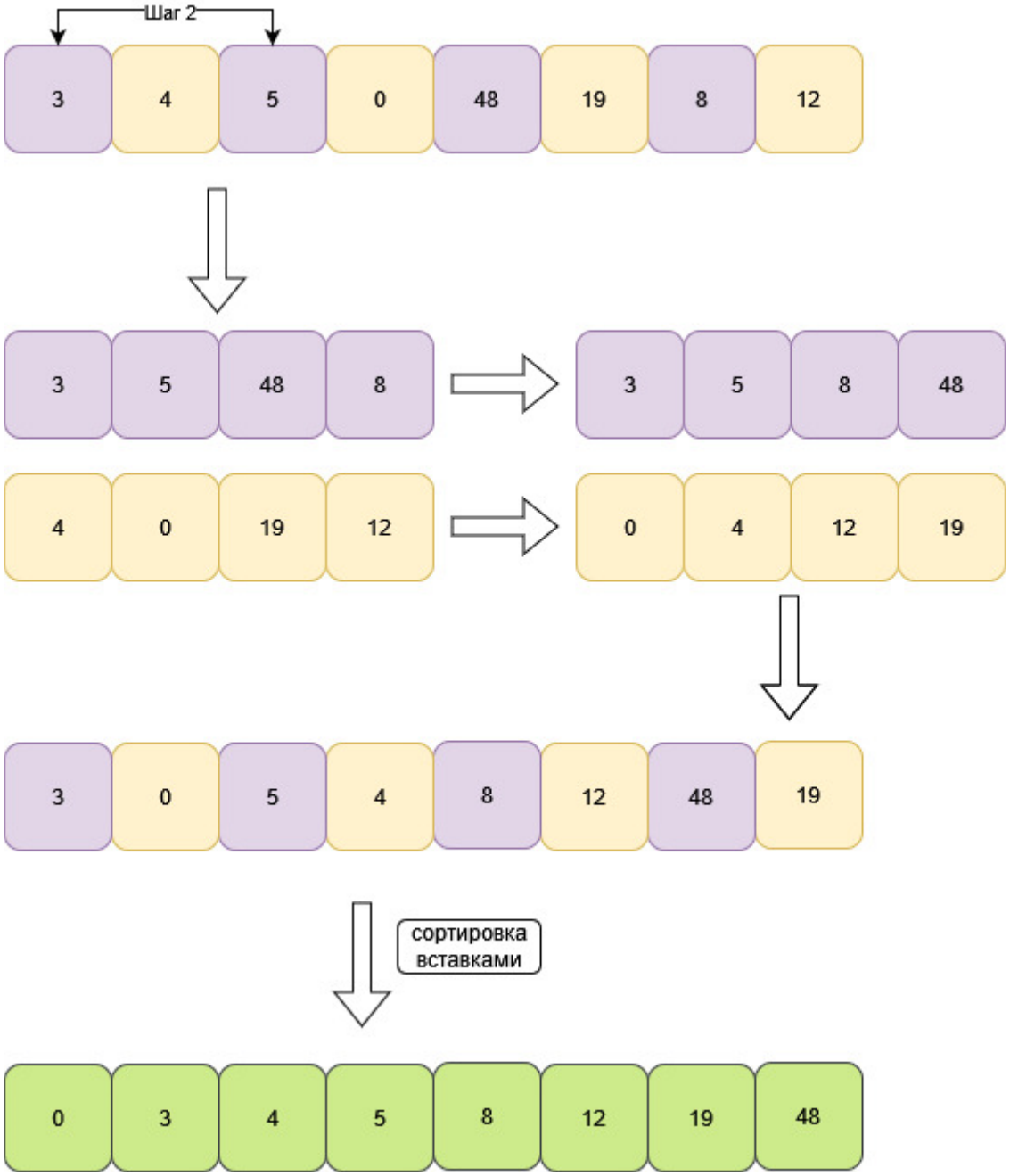
```
        int mid;
        if (start < end){

                mid=(start+end)/2;
                mergeSort(list, start, mid);
                mergeSort(list, mid+1, end);
                merge(list,start,end,mid);
        }
}

void merge(int list[],int start, int end, int mid)
{
        int mergedList[8];
        int i, j, k;
        i = start;
        k = start;
        j = mid + 1;

        while (i <= mid && j <= end) {
                if (list[i] < list[j]) {
                        mergedList[k] = list[i];
                        k++;
                        i++;
                }
                else {
                        mergedList[k] = list[j];
                        k++;
                        j++;
                }
                for (int l=0;l<k;l++)
                        cout<<mergedList[l]<<" ";
                cout<<endl;
        }

        while (i <= mid) {
                mergedList[k] = list[i];
                k++;
                i++;
        }

        while (j <= end) {
                mergedList[k] = list[j];
                k++;
                j++;
        }

        for (i = start; i < k; i++) {
                list[i] = mergedList[i];
        }
}

int main()
{
        int list[8]={3,19,8,0,48,4,5,12};
        cout<<"Input array ...\n";
        for (int i = 0; i < 8; i++)
        {
           cout<<list[i]<<"\t";
        }
        mergeSort(list, 0, 7);
        cout<<"\n\nSorted array ... \n";
        for (int i = 0; i < 8; i++)
        {
           cout<<list[i]<<"\t";
```

```
        }
}
```

6. SHELL SARALASH USULI

Bu algoritmda o’rniga qo’yish saralash usuli ishlatiladi. N o’lchamli massiv N/2 qadamlar bilan kichik massivlarga ajratiladi. Kichik massivlar o’rniga qo’yish usulida saralanadi va birlashtiriladi. Keyin N/4 qadam bilan kichik massivga bo’linadi va yuqoridagi amallar qaytalanadi. Jarayon qadam 1 ga teng bo’lganicha davom ettiraladi.





Algoritm murakkabligi

- eng yaxshi holatda O(n*logn)
- eng yomon holatda O(n^2)

Dastur kodi

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

void shellSort(int list[], int listLength)
{
        for(int step = listLength/2; step > 0; step /= 2)
        {
                for (int i = step; i < listLength; i += 1)
        {
                        int j = i;
```

```
                        while(j >= step && list[j - step] > list[i])
                        {
                                swap(list[j], list[j - step]);
                                j-=step;
                                cout<<"\ndid";
                        }
        }
        }
}

int main()
{
        int list[8]={3,19,8,0,48,4,5,12};
        cout<<"Input array ...\n";
        for (int i = 0; i < 8; i++)
        {
           cout<<list[i]<<"\t";
        }

        shellSort(list, 8);

        cout<<"\n\nSorted array ... \n";
        for (int i = 0; i < 8; i++)
        {
           cout<<list[i]<<"\t";
        }
}
```

TOPSHIRIQLAR

1. Qidirish usullariga oid masalalar

1. N o'lchamli massiv berilgan. Massiv elementlarini pufakchali saralash usulida saralang.
2. N o'lchamli massiv berilgan. Massiv elementlarini tanlash orqali saralash usulida saralang.
3. N o'lchamli massiv berilgan. Massiv elementlarini o'rniga qo'yish orqali saralash usulida saralang.
4. N o'lchamli massiv berilgan. Massiv elementlarini tez saralash usulida saralang.
5. N o'lchamli massiv berilgan. Massiv elementlarini birlashtirish saralash usulida saralang.
6. N o'lchamli massiv berilgan. Massiv elementlarini Shell saralash usulida saralang.
7. N o'lchamli massiv berilgan. Massiv elementlarini pufakchali saralash usulida saralang.
8. N o'lchamli massiv berilgan. Massiv elementlarini tanlash orqali saralash usulida saralang.
9. N o'lchamli massiv berilgan. Massiv elementlarini o'rniga qo'yish orqali saralash usulida saralang.
10. N o'lchamli massiv berilgan. Massiv elementlarini tez saralash usulida saralang.
11. N o'lchamli massiv berilgan. Massiv elementlarini birlashtirish saralash usulida saralang.

12. N o’lchamli massiv berilgan. Massiv elementlarini Shell saralash usulida saralang.

13. N o’lchamli massiv berilgan. Massiv elementlarini pufakchali saralash usulida saralang.

14. N o’lchamli massiv berilgan. Massiv elementlarini tanlash orqali saralash usulida saralang.

15. N o’lchamli massiv berilgan. Massiv elementlarini o’rniga qo’yish orqali saralash usulida saralang.

16. N o’lchamli massiv berilgan. Massiv elementlarini tez saralash usulida saralang.

17. N o’lchamli massiv berilgan. Massiv elementlarini birlashtirish saralash usulida saralang.

18. N o’lchamli massiv berilgan. Massiv elementlarini Shell saralash usulida saralang.

19. N o’lchamli massiv berilgan. Massiv elementlarini pufakchali saralash usulida saralang.

20. N o’lchamli massiv berilgan. Massiv elementlarini tanlash orqali saralash usulida saralang.

21. N o’lchamli massiv berilgan. Massiv elementlarini o’rniga qo’yish orqali saralash usulida saralang.

22. N o’lchamli massiv berilgan. Massiv elementlarini tez saralash usulida saralang.

23. N o’lchamli massiv berilgan. Massiv elementlarini birlashtirish saralash usulida saralang.

24. N o’lchamli massiv berilgan. Massiv elementlarini Shell saralash usulida saralang.

25. N o’lchamli massiv berilgan. Massiv elementlarini pufakchali saralash usulida saralang.

26. N o’lchamli massiv berilgan. Massiv elementlarini tanlash orqali saralash usulida saralang.

27. N o’lchamli massiv berilgan. Massiv elementlarini o’rniga qo’yish orqali saralash usulida saralang.

28. N o’lchamli massiv berilgan. Massiv elementlarini tez saralash usulida saralang.

29. N o’lchamli massiv berilgan. Massiv elementlarini birlashtirish saralash usulida saralang.

30. N o’lchamli massiv berilgan. Massiv elementlarini Shell saralash usulida saralang.