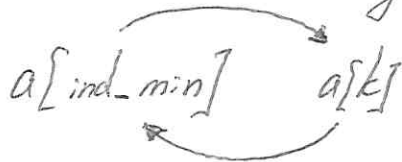


Сортировка. Метод простого выбора

Алгоритм

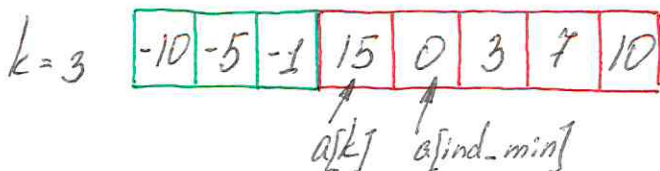
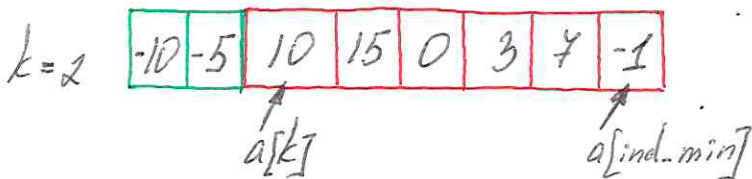
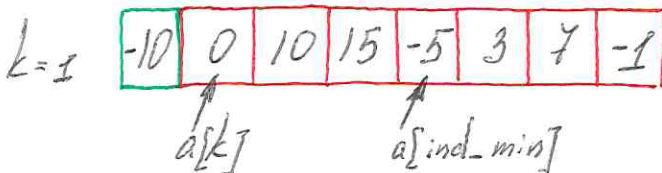
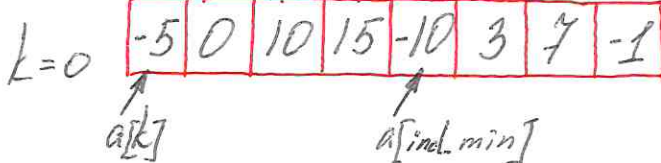
- ① Найти индекс минимального элемента в текущей области поиска. ^{k -индекс минимального элемента} Массив разделяется на 2 части:
- отсортированная часть $[0; k-1]$
 - область поиска min элемента $[k; N-1]$

- ② Переставить местами элемент в начале "области поиска" и минимальный элемент. При этом область поиска уменьшается на один элемент, а отсортированная часть увеличивается на один элемент.



- ③ Повторяем пункты ① и ② для $k \in [0; N-2]$

$N=8$



и т.д.

зеленым цветом обозначена
отсортированная часть

красным цветом обозначена
область поиска минимума

Сортировка. Метод пузырька.

12

Алгоритм

- ① Для всех элементов массива $[0; N-2]$ проверить условие $a_i > a_{i+1}$. Если условие выполняется, то поменять местами a_i и a_{i+1}
- ② Сделать подсчет количества обменов в пункте ①
- ③ Если количество обменов больше нуля, то повторить пункты ① и ②. Если нет - сортировка завершена.

$N = 8$

$i=0$

-5	0	10	15	-10	3	7	-1
----	---	----	----	-----	---	---	----

не меняем

$i=1$

-5	0	10	15	-10	3	7	-1
----	---	----	----	-----	---	---	----

не меняем

$i=2$

-5	0	10	15	-10	3	7	-1
----	---	----	----	-----	---	---	----

не меняем

$i=3$

-5	0	10	15	-10	3	7	-1
----	---	----	----	-----	---	---	----

меняем

count = 1

$i=4$

-5	0	10	-10	15	3	7	-1
----	---	----	-----	----	---	---	----

меняем

count = 2

.....

После первого прогона

-5	0	10	-10	3	7	-1	15
----	---	----	-----	---	---	----	----

count = 4

Всего понадобится 6 прогонов для этого массива

Сортировка Раёёёёёё

Алгоритм

- Для всех элементов массива $[0; N-1-q]$ проверить условие $a_i > a_{i+q}$. Если условие выполняется, то поменять местами a_i и a_{i+q}
- Сделать подсчет количества обменов в пункте 1
- Если количество обменов больше нуля, то повторить пункты 1 и 2. Если нет - уменьшить расстояние q в 2 раза ($q = q/2$) и повторить пункты 1 и 2
- Пункты 1, 2 и 3 повторяют до тех пор, пока $q \geq 1$
Изначально $q = N/2$ - группы элементов

$N=8$

-5 0 10 15 -10 3 7 1

-5, -10 -10, -5
0, 3 0, 3
10, 7 7, 10
15, 1 1, 15

сортировка
внутри группы

-10 0 7 1 -5 3 10 15

-10 7 -5 10 -10 -5 7 10
0 1 3 15 0 1 3 15

сортировка
внутри группы

-10 0 -5 1 7 3 10 15

-10 0 -5 1 7 3 10 15 → -10 -5 0 1 3 7 10 15

-10 -5 0 1 3 7 10 15

$q = 4$ (4 группы элементов)

a_0	a_4	a_2	a_6
a_1	a_5	a_3	a_7

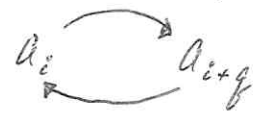
$q = 2$ (2 группы элементов)

a_0	a_2	a_4	a_6
a_1	a_3	a_5	a_7

$q = 1$ (1 группа элементов)

всего 8 сортировок
пузырьком

```

for (g = N/2; g > 0; g /= 2) {
    do {
        count = 0;
        for (i = 0; i < N - g; i++) {
            if (a_i > a_{i+g}) {
                
                count++;
            }
        }
    } while (count > 0);
}

```


Сортировка одномерного массива.

Метод простого выбора.

Метод заключается в последовательном нахождении минимального или максимального элемента (в зависимости от того сортируем ли мы массив по возрастанию или по убыванию) и перестановке его в начало массива.

```
for (k = 0; k < N - 1; k++) {  
    // нахождение индекса минимального элемента  
    ind_max = k;  
    for (i = 1 + k; i < N; i++)  
        if (arr[i] > arr[ind_max])  
            ind_max = i;  
  
    // обмен элементов  
    temp = arr[k];  
    arr[k] = arr[ind_max];  
    arr[ind_max] = temp;  
}
```

ind_max – индекс текущего максимального элемента;

k – индекс элемента, который обменивается с максимальным элементом.

На первом шаге мы должны поставить максимальный элемент всего массива на место 1го элемента. То есть поменять arr[0] и arr[ind_max]. Далее мы ищем новый максимальный элемент в хвостовой части массива (не учитывая arr[0]) и меняем его местами со вторым элементом массива. То есть меняем arr[1] и arr[ind_max]. Далее снова ищем максимальный элемент в хвостовой части массива (не учитывая arr[0] и arr[1]) и меняем его с третьим элементом. То есть меняем arr[2] и arr[ind_max]. Далее продолжаем находить максимальные элементы из хвостовой не отсортированной части массива и менять их с первым не отсортированным элементом пока не достигнем предпоследнего элемента.

Метод пузырька.

Сортировка методом пузырька предполагает многократное прохождение по массиву и обмен рядомстоящих элементов массива в том случае, если эти элементы стоят в неверном порядке.

В нашем случае, переменная `count` будет отвечать за количество обменов, совершенных при прохождении вдоль массива. Эта переменная обнуляется, затем происходит обход массива. Если во время обхода был сделан хотя бы один обмен элементов местами, то `count` снова обнуляют и повторяют обход. Если же переменная после обхода массива осталась равной нулю, то значит, массив уже отсортирован.

```
int count = 1; //переменная для подсчета количества обменов
```

```
while (count > 0) {
```

```
    //обнуление количества обменов
```

```
    count = 0;
```

```
    //прохождение по массиву
```

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

```
        if(arr[i] > arr[i + 1]) {
```

```
            //обмен элементов
```

```
            temp = arr[i];
```

```
            arr[i] = arr[i + 1];
```

```
            arr[i + 1] = temp;
```

```
            //подсчет количества обменов
```

```
            count++;
```

```
        }
```

```
    }
```

```
}
```