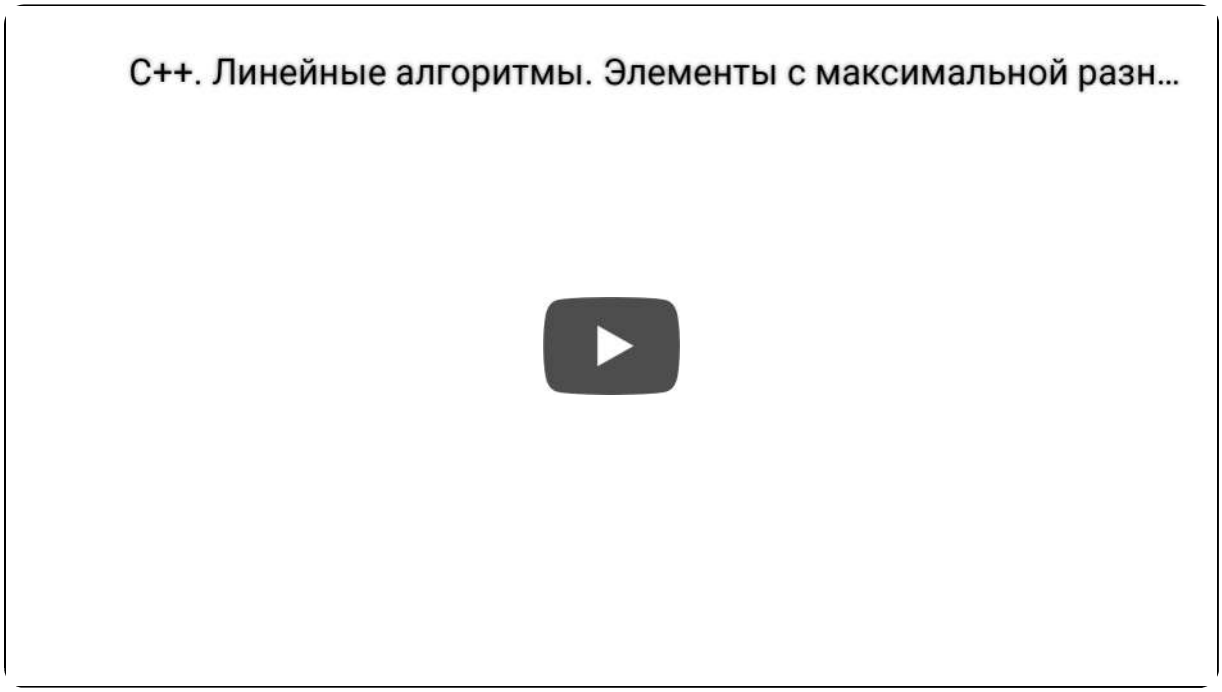
Элементы с максимальной разностью



Линейные алгоритмы представляют собой класс алгоритмов, позволяющих решать задачи за линейную сложность $O(n)$, где n — размерность задачи.

Рассмотрим интересные примеры таких задач.

Задача. Дан массив $a_0, a_1, \ldots a_{n-1}$. Необходимо выбрать в нём два элемента a_i и a_j такие, что $i < j$ и разность $(a_j - a_i)$ максимальна.

Заметим, что утверждение о том, что элемент a_j — максимальный элемент в массиве, неверно. Например, для массива $\{10, 6, 3, 4, 5, 1, 0\}$ необходимо выбрать разность $a_4 - a_2$, и ни один из выбранных элементов сам по себе не обладает какими-либо особыми свойствами в массиве.

Очевидно, что можно перебрать все пары a_i и a_j, но тогда алгоритм не будет линейным.

Рассмотрим следующую идею. Будем перебирать правый элемент пары a_j, а левый элемент a_i подбирать наилучшим образом. Заметим, что для фиксированного a_j наилучшим элементом a_i будет минимальный элемент слева от него. Во время перебора правого элемента пары все элементы слева

от него уже просмотрены, а значит, при переборе мы можем поддерживать информацию про минимальный элемент (обозначим его индекс $imin$) на отрезке $[0; j-1]$. Пересчитывать $imin$ при увеличении j на 1 будем за $O(1)$. Вычислительная сложность всего алгоритма составляет $O(n)$.

Реализация

```
int imin = 0;
int ibest = 0;
int jbest = 1;
for (int j = 2; j < n; ++j)
{
    if (a[j - 1] < a[imin])
        imin =  j - 1;
    if (a[j] - a[imin] > a[jbest] - a[ibest])
    {
        jbest = j;
        ibest = imin;
    }
}
cout << ibest << " " << jbest << endl;
```