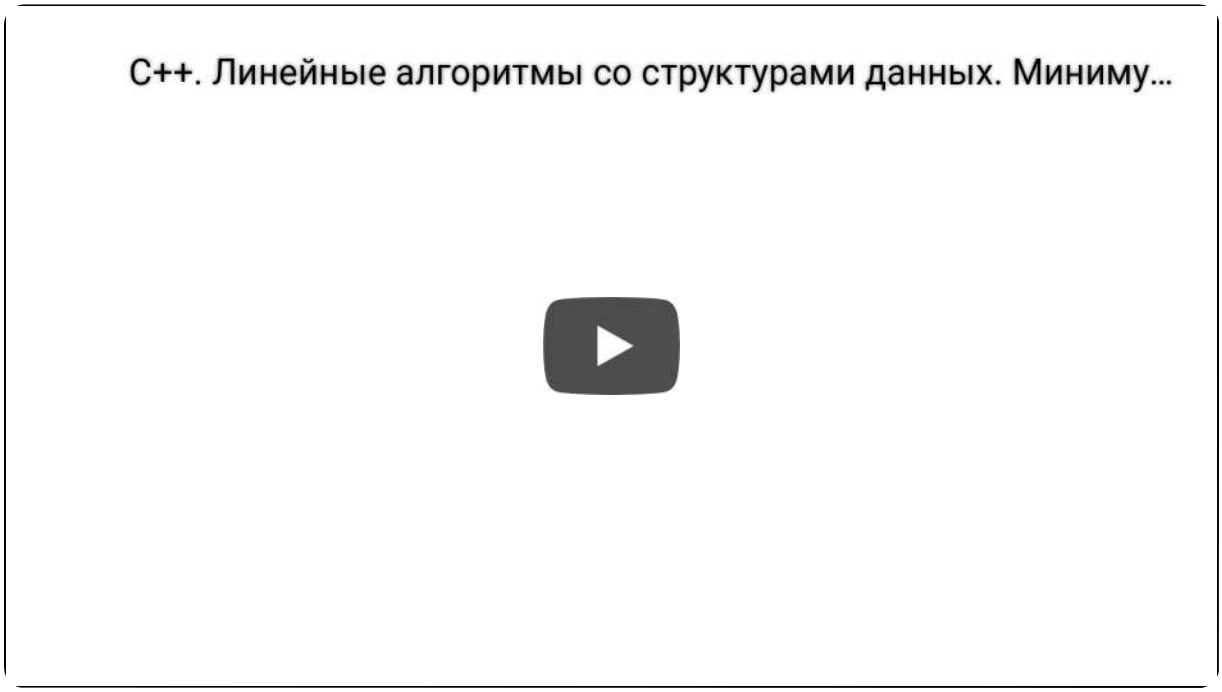
Минимум в скользящем окне с деком



Рассмотрим ещё один способ решения задачи поиска минимума в скользящем окне.

Будем перебирать все окна в цикле по переменной i. Будем хранить в деке d индексы так называемых "перспективных элементов", то есть тех, которые могут теоретически стать ответом для текущего окна или какого-либо окна правее текущего.

Заметим, что левее минимума в текущем окне перспективных элементов быть не может. Элементы, расположенные в рассматриваемом окне правее и большие текущего минимума могут оказаться перспективными, так как при рассмотрении окна, в котором не лежит текущий минимум, какой-то из таких элементов может оказаться минимумом в этом окне.

Заметим, что последовательность значений перспективных элементов одного окна всегда строго возрастает. Если в процессе алгоритма появился элемент меньший, чем последний перспективный элемент, то мы должны удалить все большие или равные этому элементу ранее перспективые элементы из конца дека, так как они перестали быть таковыми, и добавить в конец дека текущий

элемент. При этом минимум в текущем окне будет всегда храниться в начале дека. Также при переходе к следующему окну необходимо проверять, что первый элемент дека всё ещё принадлежит рассматриваемому окну. Если это не так, то его тоже необходимо удалить. Именно потому, что элементы в данном алгоритме удаляются как из начала, так и из конца нашей структуры, мы используем дек. Обратите внимание, что храним мы в нём индексы перспективных элементов, а не их значения.

Данный алгоритм лучше рассмотренного ранее, так как он является однопроходным, то есть не нужно предварительно насчитывать какую-либо информацию, и ответ находится за n действий.

Данный алгоритм является линейным, так как каждый элемент массива будет ровно один раз добавлен в дек и не более одного раза удалён из дека.

Реализация

```
deque <int> d;
for (int i = 0; i < n; ++i) {
    if (i >= k && d.front() == i - k)
        d.pop_front();

    while (d.size() && x[d.back()] >= x[i])
        d.pop_back();
    d.push_back(i);
    if (i >= k - 1)
        res.push_back(d.front());
}
```