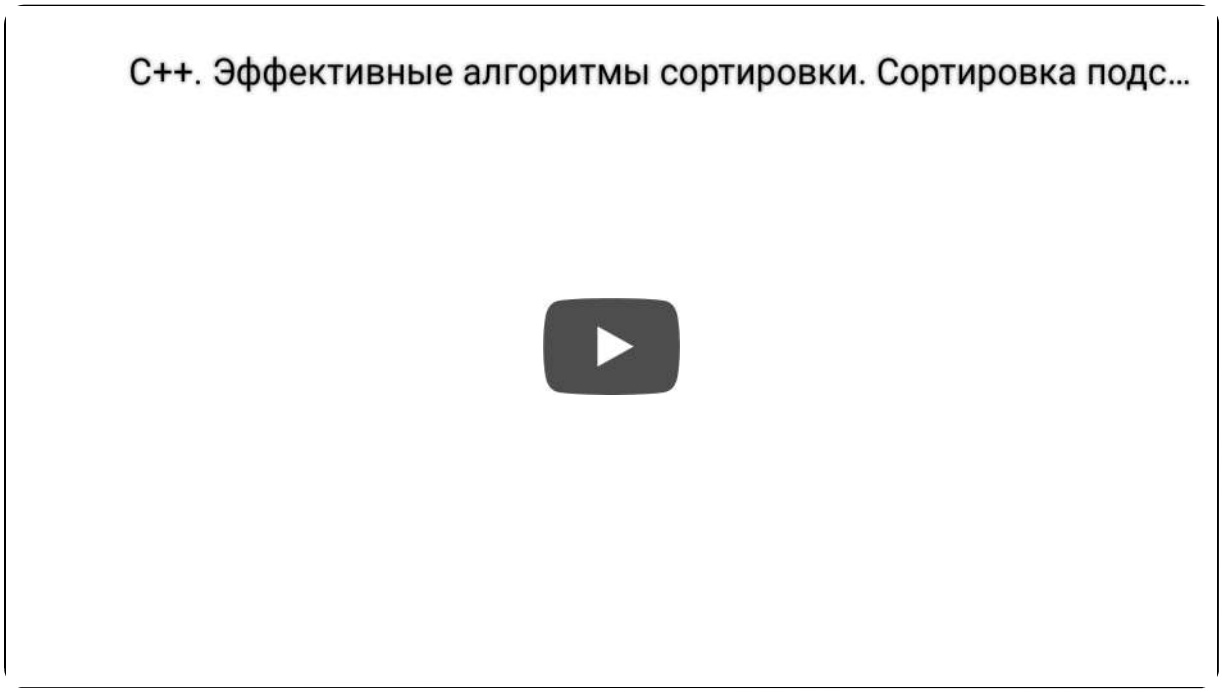
Сортировка подсчётом



Пусть нам надо отсортировать массив, состоящий только из десятичных цифр. Оказывается, в этом случае существует более эффективный алгоритм сортировки по сравнению с рассмотренными ранее. А именно подсчитаем во вспомогательном массиве cnt количество вхождений каждой из цифр в исходный массив a. Сделать это можно за один проход массива a. А затем заполним массив a заново согласно значениям массива cnt.

Рассмотрим на примере массива:

$$[2, 3, 5, 1, 3, 2, 0, 9]$$

После подсчёта количества каждой из цифр в массиве cnt будут храниться следующие значения :

i	0	1	2	3	4	5	6	7	8	9
$cnt[i]$	1	1	2	2	0	1	0	0	0	1

Далее, пройдя с помощью индекса i по массиву cnt, заполним массив a заново, а именно запишем в него $cnt[i]$ раз число i. Получим:

$$[0, 1, 2, 2, 3, 3, 5, 9]$$

Можно использовать рассмотренную идею не только для сортировки

десятичных цифр, но и для сортировки массива, элементам которого можно поставить в соответствие числа от 0 до $M-1$. Данный алгоритм называют **сортировкой подсчётом**.

Очень удобно использовать метод подсчёта для массивов символов. В таком случае каждому символу можно поставить в соответствие его ASCII-код. Если это строчные латинские символы, то их можно отображать в массив от 0 до 25.

Несложно понять, что вычислительная сложность полученного алгоритма — $O(N+M)$, где N — количество элементов в массиве a, а M — количество возможных значений, которые встречаются в нём. Также алгоритм зависит от M и по количеству используемой памяти, поэтому его полезно использовать, только когда M сравнительно небольшое (в первую очередь когда $M \leq N$).

Реализация

Приведём основной фрагмент программы сортировки массива цифр:

```
vector<int> cnt(10, 0);
for (auto el: a)
    ++cnt[el];
a.resize(0);
for (int d = 0; d < cnt.size(); ++d)
    a.insert(a.end(), cnt[d], d);
```