Алгоритмы и структуры данных. Линейные алгоритмы 2

densine, w1nSkEeper

6 декабря 2025 г.

Содержание

1. Разминка. MEX

1.1. Определение

MEX – Minimum Excluded (исключенный минимум), т.е. минимальное неотрицательное (в данном случае) число, не лежащее в каком-нибудь массиве.

$$mex(1, 2, 3) = 0$$

$$mex(0, 1, 2, 5, 6) = 3$$

1.2. Задача

$p_1, p_2, \dots p_n$ – перестановка чисел от 0 до $n-1$. Поступает q запросов вида: $l, r : 1 \le l \le r \le n$. Найти $mex(p_l, p_{l+1}, \dots p_r)$. Требование – $O(1)$ на запрос. Пусть x – ответ. Тогда если $l = 1, r = n$ то ответ – n. Иначе:

$$0 \le x \le r - l + 1 \le n - 1$$

x не лежит в $p_l, \dots p_r$ по определению. Тогда ответ лежит либо в $p_1, \dots _{l-1}$, либо в $p_{r+1}, \dots p_n$. Пусть минимум на них это a и b соответственно. Тогда ответом, логично, является $min(a, b)$. Тогда за $O(1)$ мы можем отвечать посчитав заранее суффиксные и префиксные минимумы!

2. Минимум в скользящем окне

Даны $k, n : k \le n$. $a_1, \dots a_n$ – какие-то числа. Требуется найти b, для которого выполняется:

$$b_1 = min(a_1, \dots a_k)$$

$$b_2 = min(a_2, \dots a_{k+1})$$

$$b_3 = min(a_3, \dots a_{k+2})$$

...

Будем поддерживать дек кандидатов в минимумы. В нем храним сам минимум и его индекс. В момент, когда нужно добавлять новый элемент, смотрим – меньше ли он последнего элемента в деке? Пока да – удаляем верхний элемент. Затем добавляем текущий, и удаляем первый если его индекс не входит по ограничению временных (i, j). Код:

```
deque<pair<int, int>> d;
for (int i = 0; i < n; i++) {
    while (!d.empty() && d.front().second <= i - k) d.pop_front();
    while (!d.empty() && d.back().first > a[i]) d.pop_back();
    d.push_back({a[i], i}); // добавляем элементы всего n раз,
                            // а значит и удалим всего n раз!
    if (i + 1 >= k) d.front().first // это ответ
}
```

Таким образом сложность такого алгоритма – $O(n)$.

3. Ближайший меньший справа

Дано n, и $a_1, \dots a_n$ – какие-то числа. Найти для каждого a_i найти $a_j < a_i : j > i$ и $j - min$. Будем поддерживать дек чисел "без ответа". Когда встречаем число, меньшее чем те, что какие-то лежащие в деке, их убираем (записываем куда-то ответ). Продолжаем до конца. Числа оставшиеся в деке не имеют меньшего справа.

```
vector<pair<int, int>> d;

for (int i = 0; i < n; i++) {
    while (!d.empty() && d.back().first > a[i]) {
        ans[d.back().second] = i;
        d.pop_back();
    }
    d.push_back({a[i], i});
}
```

4. Гистограммы

Дано n и массив $a_1, \dots a_n : 0 \leq a_i \leq 10^9$. Найти максимальной по площади прямоугольник из башен гистограммы.

Будем перебирать a_i, и пытаться прямоугольник, образованный это башенкой влево и вправо. И вот так выходит, что максимально влево можно расширить до ближайшего меньшего слева, и аналогично для правой части (очевидно).

5. Максимальный прямоугольник

Аналогичное задание для матрицы n x m, причем $n \cdot m \leq 10^6$. Перебираем столбцы и считаем "высоты". Затем перебираем строки – тут решение сводится к перебору конкретного элемента как в гистограммах.