Оптимизация Матрицы

Автор задачи и разработчик: Константин Бац

Формализуем условие задачи. Нам дан некоторый массив $w_1, w_2, \ldots, w_n$. Мы должны выбрать один раз некоторую позицию j, и после этого значение w_i для всех i между $\max(j-k, 1)$ и $\min(j+k, n)$ заменятся на значение w_j. Нам требуется найти такое j, после которого сумма w_i по всему массиву будет максимальной, и вывести эту сумму.

Для решения первой и, возможно, второй подзадачи можно было разобрать случаи для всех возможных n и k, прописав каждый случай в коде явно. Ниже приведен код для $n \leqslant 3$. В случае $k = 0$ замены вообще не производятся, а при $k \geqslant 2$ выбранное значение покрывает весь массив, так что достаточно выбрать максимальное.

```
if k == 0:
    print(sum(w))
elif k == 1:
    if n <= 2:
        print(max(w) * n)
    elif n == 3:
        print(max(w[0] * 2 + w[2], w[1] * 3, w[0] + w[2] * 2))
else:
    print(max(w) * n)
```

В третьей подзадаче $k = 0$, а это значит, что выбор j не меняет значение никаких w_i. Следовательно, вне зависимости от j ответ на задачу будет равен сумме w_i. На самом деле, как нетрудно убедиться, приведенный выше код также решает и эту подзадачу.

В четвертой подзадаче не было дополнительных ограничений на k, однако $n \leqslant 1000$ позволяло просто перебрать все j, для каждого посчитать сумму w_i в k-окрестности вокруг него и выбрать максимальный из них. Ответ на задачу можно записать в виде формулы

$$\max_{j=1}^{n} \left\{ \sum_{i=1}^{n} \begin{cases} w_j, & \text{если } j-k \leqslant i \leqslant j+k \\ w_i, & \text{иначе} \end{cases} \right\}.$$

В коде это представляется в виде двух вложенных циклов: первый цикл по j, второй по i. Время работы такого решения равно $\mathcal{O}(n^2)$.

Для прохождения последней подзадачи давайте немного оптимизируем нашу формулу. Записанную выше сумму можно разбить на случаи, когда $i < j-k$, $i > j+k$ и i находится на расстоянии не больше k от j. Запишем эту сумму следующим образом:

$$\left(\sum_{i<j-k} w_i\right) + \left(\sum_{j-k\leqslant i\leqslant j+k} w_j\right) + \left(\sum_{i>j+k} w_i\right) = \left(\sum_{i<j-k} w_i\right) + \left(\sum_{i>j+k} w_i\right) + (d \cdot w_j),$$

где за d обозначено количество i между $j-k$ и $j+k$, то есть $\max(j-k, 1) - \min(j+k, n) + 1$.

Для того, чтобы для выбранного j находить такую сумму за $\mathcal{O}(1)$, достаточно посчитать массив префиксных сумм $\texttt{pref}[t] = \sum_{i=1}^{t} w_i$. Это можно сделать за линейное время, использовав тот факт, что $\texttt{pref}[0] = 0$, а для $t > 0$ верно $\texttt{pref}[t] = \texttt{pref}[t-1] + w_t$. Тогда для подсчета ответа для фиксированного j можно пользоваться следующими равенствами:

$$\sum_{i<j-k} w_i = \texttt{pref}[\max(1, j-k) - 1]$$

$$\sum_{i>j+k} w_i = \texttt{pref}[n] - \texttt{pref}[\min(j+k, n)]$$

Учитывая, что массив pref предподсчитывается один раз за линейное время, осталось перебрать все возможные j, для каждого найти результат за $\mathcal{O}(1)$ описанным выше способом, и выбрать максимальный. Таким образом, общее время работы программы — $\mathcal{O}(n)$.