Разбор задачи «Символы»

В этой задаче достаточно было последовательно проверить применимость всех операций к заданному набору критериев. Поскольку ни одной из операций не отдавалось предпочтение (и последний пример в условии это явно иллюстрировал), то порядок проверки не являлся существенным. После отыскания первого же подходящего варианта можно было сразу его выводить и прерывать выполнение программы.

Разбор задачи «Точка на карте»

В этой задаче было необходимо отыскать квадрат, имеющий наименьшую площадь пересечения с прямоугольником, описывающим город. При этом нужно было следить за тем, чтобы квадрат касался прямоугольника хотя бы в одной точке.

Поскольку стороны квадрата и прямоугольника параллельны, их пересечением будет прямоугольник. Площадь пересечения квадрата с центром в (x_i, y_i) и стороной $2 \cdot d$ и прямоугольника с левым нижним углом в $(0, 0)$ и верхним правым углом в (a, b) может быть вычислена следующим образом.

Левая граница прямоугольника, являющегося их пересечением, будет лежать на прямой, параллельной оси OY и проходящей через точку $x_{left} = max(x_i - d, 0)$; правая граница также будет лежать на прямой, параллельной оси OY и проходящей через точку $x_{right} = min(x_i + d, a)$; нижняя и верхняя границы будут лежать на прямых, параллельных оси OX и проходящих через точки $y_{bottom} = max(y_i - d, 0)$ и $y_{top} = min(y_i + d, b)$.

Площадь прямоугольника-пересечения (если он существует) вычисляется как $S_{int} = (x_{right} - x_{left}) \cdot (y_{top} - y_{bottom})$. Разумеется, перед вычислением площади следует выполнить проверку, что каждый из сомножителей является неотрицательным числом.

Собственно задача выбора прямоугольника-пересечения минимальной площади из некоторого набора таких прямоугольников является тривиальной (эквивалентной отысканию минимума в массиве).

Разбор задачи «Очень скоростной трамвай»

Рассмотрим пассажира $\#j$, который в настоящее время садится на остановке s_j и выходит на остановке f_j и выясним, продолжит ли он пользоваться трамваем, если единственной оставшейся остановкой будет остановка r_k.

Согласно условию задачи, чтобы пассажир воспользовался трамваем, должно выполняться неравенство $|r_k - s_j| + ((n + 1) - f_j) \leqslant (f_j - s_j)$. Для упрощения выделим два случая — когда остановка r_k будет находиться левее остановки s_j и когда остановка r_k будет находиться правее остановки s_j (случай совпадения остановок можно включить в любой из этих двух).

Если $r_k \leqslant s_j$, то неравенство можно записать как $s_j - r_k + n + 1 - f_j \leqslant f_j - s_j$ и получить оценку на r_k:

$$\begin{cases} r_k \leqslant s_j \\ n + 1 - 2 \cdot (f_j - s_j) \leqslant r_k \end{cases}$$

Таким образом, если r_k больше или равно минимальному значению $r_k^{(left)} = n + 1 - 2 \cdot (f_j - s_j)$, то пассажиру выгодно воспользоваться трамваем (конечно, при соблюдении первого неравенства).

Теперь выполним оценку справа. Пусть $s_j \leqslant r_k$, тогда $r_k - s_j + n + 1 - f_j \leqslant f_j - s_j$, откуда

$$\begin{cases} s_j \leqslant r_k \\ r_k \leqslant 2 \cdot f_j - (n + 1) = r_k^{(right)} \end{cases}$$

В результате получаем диапазон для r_k: $\left[r_k^{(left)}, r_k^{(right)}\right]$, приемлемый для пассажира $\#j$. Конечно, нужно учитывать, что для некоторых пассажиров приемлемых диапазонов не будет существовать.

Вычислим левые и правые границы описанных диапазонов для всех пассажиров, для которых такие диапазоны в принципе существуют. Это позволяет свести задачу к задаче о скобочной последовательности. На всякий случай кратко напомним, как решается такая задача.

Классический подход выглядит следующим образом. Будем описывать каждую границу диапазона парой значений — собственно номером остановки и типом границы (является ли она левой или правой). Поместим эти пары значений в список (или массив) и отсортируем их по возрастанию; равные значения будем упорядочивать по принципу «левая граница предшествует правой». Также будем поддерживать суммарное количество пассажиров, для которых текущий номер остановки является приемлемым. Когда встречается очередная левая граница, суммарное количество надо увеличить, когда встречается очередная правая граница — уменьшить. Если левая и правая граница приходятся на одну и ту же остановку, количество пассажиров, конечно, будет равно сумме после обработки левой границы (всех левых границ, приходящихся на эту остановку), но до обработки правой границы (всех правых границ, приходящихся на эту остановку).

Описанная процедура требует достаточно аккуратной реализации, хотя идейно является не особенно сложной. Асимптотика описанного решения $O(n \log n)$ (за счет сортировки).

Существует более элегантное решение, не требующее сортировки. Оно использует тот факт, что количество остановок не превосходит $3 \cdot 10^5$.

Рассмотрим пассажира $\#j$. Выше мы уже выполняли расчёт для некоторой остановки r_k — будет ли она приемлема для этого пассажира, и нашли верхнюю и нижнюю границы значения r_k для пассажира $\#j$. Поскольку логика рассмотрения будет несколько иной, введём такие обозначения: $left[j]$ для самой левой приемлемой остановки для пассажира $\#j$ и $right[j]$ — для самой правой приемлемой остановки для пассажира $\#j$.

Было бы удобно, если бы у нас имелся массив $counts[i]$, в котором мы для каждой остановки $\#i$ хранили бы количество пассажиров, которые воспользуются этой остановкой, если останется именно она. Элементы этого массива и служили бы ответами на запросы.

Тогда при рассмотрении очередного пассажира (пассажира $\#j$) для всех остановок от $left[j]$ до $right[j]$ необходимо увеличить значение элементов массива $counts[k]$ ($k = left[j], left[j] + 1, \ldots, right[j]$) на 1. Понятно, что такое увеличение непосредственно для каждого элемента потребует много времени, но сейчас мы увидим, как выполнить его эффективно.

Пусть мы вычислили значение $counts[1]$ для самой первой остановки. Рассмотрим теперь остановку #2. Часть пассажиров, которых устраивала посадка на остановке #1, будет устраивать и посадка на остановке #2, а какую-то часть пассажиров посадка на остановке #2 устраивать уже не будет. Это те пассажиры, для которых значение $rigth[j] = 1$.

Кроме того, могут появиться новые пассажиры, которых будет устраивать посадка на остановке #2 — это те, для которых $left[j] = 2$. Таким образом, чтобы получить из значения $counts[1]$ значение $counts[2]$, нужно вычесть из значения $counts[1]$ количество пассажиров, для которых $right[j] = 1 (= 2 - 1)$ и добавить количество пассажиров, для которых $left[j] = 2$.

Аналогичным образом мы можем провести вычисления для любых двух соседних остановок $\#i$ и $\#(i+1)$. Чтобы получить количество пассажиров, которые сядут в трамвай на остановке $\#(i+1)$, вычтем из количества пассажиров, садящихся в трамвай на остановке $\#i$ тех, для кого остановка $\#i$ последняя в приемлемом диапазоне, и добавим тех, для кого остановка $\#(i + 1)$ — первая в приемлемом диапазоне.

В программе мы выполним описанные действия в ином порядке. Изначально заполним все элементы массива $counts[i]$ нулями. Последовательно рассмотрим всех пассажиров и вычислим для каждого из них границы приемлемого диапазона остановок. Когда мы рассматриваем очередного пассажира $\#j$, будем увеличивать на единицу величину $counts[left[j]]$ и уменьшать на единицу величину $counts[right[j] + 1]$. Тем самым мы сохраним в каждом элементе $counts[i]$ величину, на которую этот элемент отличается от предыдущего.

Заметим, что после рассмотрения всех пассажиров в элементе $counts[1]$ окажется как раз количество пассажиров, которые будут пользоваться трамваем, если на линии останется только остановка #1. Всё, что нам нужно сделать, чтобы получить правильно заполненный массив $counts[i]$ — это выполнить суммирование его элементов нарастающим итогом:

```
for i = 2 to n do
    counts[i] = counts[i] + counts[i - 1]
```

Теперь для ответа на запрос останется просто вывести соответствующий элемент массива.

Разбор задачи «Постепенность»

По условию, мэр полагает, что каждый пассажир с остановки, которая будет ликвидирована, будет проходить ровно одну остановку (до ближайшей). Таким образом, необходимо, соблюдая описанные в задаче требования, убрать остановки с наибольшим суммарным количеством пассажиров.

Эта задача решается методом динамического программирования. Пусть $dp[i]$ — это количество пассажиров, которое мы можем лишить их привычных остановок, при условии, что остановка i сохранится в маршруте. Поскольку последняя остановка из маршрута исчезнуть не должна, то в $dp[n]$ в итоге будет содержаться ответ.

Как несложно видеть, $dp[i] = max(dp[i-1], max(dp[i-2], (dp[i-3]+p[i-2]))+p[i-1])$, где $p[j]$ — количество пассажиров, привыкших садиться на соответствующей остановке.

Однако недостаточно отыскать максимальное количество пассажиров, требуется еще и построить новый маршрут. Для этих целей можно поддерживать массив $from[i]$, в котором хранить номера остановок $(i-1)$, $(i-2)$ или $(i-3)$ соответственно — в зависимости от того, какая из них позволила сформировать максимальное значение.

Асимптотика данного решения $O(n)$.