Разбор задачи «Собеседования»

Попробуем отправить Кешу сначала в AAA, потом в BBB, а затем — наоборот. Что нам это даст?

Если сначала мы отправим его в AAA, нам надо выяснить, верно ли, что $s_1 + t_1 + r \leqslant f_2$, т.е. успеет ли Кеша при самом «неудачном» раскладе пойти к началу собеседования в AAA, а потом успеть к концу собеседования в BBB. Если верно, выводим AB.

Если это неравенство не выполняется, попробуем поступить наоборот и проверим, что $s_2 + t_2 + r \leqslant f_1$. Если это неравенство выполнено, выводим BA.

Если оба неравенства не выполняются, значит, на два собеседования Кеша никак не успевает. Поэтому посчитаем, после какого он освободится раньше, т.е. выясним, что меньше — $s_1 + t_1$ или $s_2 + t_2$, и, соответственно, выведем A или B.

Разбор задачи «Первый день стажировки»

В данных ограничениях можно просто просимулировать описанный процесс.

Одно из возможных решений может быть таким. Очевидно, что из каждой вазочки Кеша съест хотя бы одно печенье. Поэтому посчитаем, сколько он съест сверх этого количества.

Будем хранить номер вазочки, печенье из которой в текущий момент нравится Кеше больше всего, а также количество печений, которое он уже съел из этой вазочки. Кроме того, будем поддерживать в актуальном состоянии максимум количества печений, съеденных из одной вазочки (и ее номер, поскольку его тоже нужно вывести в качестве ответа).

В начальный момент времени наиболее вкусным Кеша будет считать печенье из вазочки #1 (другого он просто не пробовал). Будем просматривать вазочки слева направо до тех пор, пока не встретим вазочку, печенье в которой будет более вкусным (или таким же), как и в ней. Количество таких вазочек и будет равно количеству печений, которое Кеша съест из этой вазочки «дополнительно».

Когда печенье в очередной вазочке окажется таким же или более вкусным, обновим номер «самой вкусной» вазочки и установим в 0 количество дополнительно съеденных из нее печений. Разумеется, при обновлении номера вазочки не следует забывать обновлять (при необходимости) максимум.

Разбор задачи «Удобный момент»

Опишем один из возможных способов решения задачи.

Выполним некоторые подготовительные действия. Для каждого простого задания, которое будет выполнять Кеша, посчитаем, в какой момент времени он его завершит. Нам также будет удобно дополнить набор заданий Кеши фиктивным нулевым заданием с нулевой длительностью выполнения. Будем хранить вычисленные значения в массиве (списке) $preA[]$.

Точно также посчитаем, в какие моменты времени Потап будет завершать выполнение очередного своего задания. Эти величины будем хранить в массиве (списке) $preC[]$.

Теперь применим технику двух указателей. Предположим, что Кеша сразу же примется за сложное задание (т.е. установим первый указатель на позицию 0 — после фиктивного нулевого задания). Это значит, что спустя b единиц времени Кеша будет готов задавать вопросы Потапу. Будем двигать второй указатель по массиву $preC[]$ до тех пор, пока не найдем первое значение, не меньшее b. Пусть это значение $preC[j_0]$ Разность между этим значением и b и будет составлять время, которое Кеше придется подождать в этом случае.

Посмотрим, что произойдет, если Кеша сначала выполнит первое простое задание, а потом возьмется за сложное. Пусть в этом случае он сможет задать вопрос Потапу в момент времени $preC[j_1]$. Очевидно, что $j_1 \geqslant j_0$, поэтому, чтобы получить j_1, нужно просматривать элементы массива $preC[]$, начиная с j_0. Посчитаем разность $preC[j_1] - (preA[1] + b)$, сравним ее с полученной на предыдущем этапе $preC[j_0] - (preA[0] + b)$ и будем хранить минимум из этих величин.

Точно так же посчитаем, сколько придется ожидать Кеше, если он начнет выполнять сложное задание после второго простого, после третьего простого и т.д., и найдем минимум среди этих значений.

Разбор задачи «Новые особенности»

Эту задачу можно решать бинпоиском по ответу или методом двух указателей.

Метод двух указателей можно применить следующим образом.

Предварительно посчитаем время s, которое потратил бы Кеша на написание всех функций, если бы не использовал нововведения вовсе.

Будем хранить описание каждой функции в виде пары (w_j, d_j). Упорядочим все пары по неубыванию w. Будем описывать запросы парой (*номер запрос*, t_i) и упорядочим эти пары по неубыванию t_i. Установим первый указатель на первый (в упорядоченной последовательности) запрос i_0 и посчитаем сумму всех w_j, не превосходящих t_{i_0} (пусть это будет sw_{i_0}), а также сумму всех соответствующих им d_j (пусть это будет sd_{i_0}. Вычислить время, которое потребовалось бы Кеше на написание функций в этом случае, вычисляется как $s - sd_{i_0} + sw_{i_0}$.

Как понятно, для следующего запроса i_1 верно $t_{i_0} \leqslant t_{i_1}$, поэтому, чтобы узнать, сколько времени потратит Кеша в этом случае, нет необходимости пересчитывать все заново: достаточно посчитать сумму тех w_j, которые будут больше t_{i_0}, но при этом не будут превосходить t_{i_1} (и, разумеется, сумму соответствующих d_j).

Когда вычисления для всех запросов будут завершены, останется переупорядочить их результаты по номерам запросов.

Разбор задачи «Конференция»

Эта задача решается методом динамического программирования.

Пусть $dp[v][i]$ — суммарная интересность последовательности докладов, если последним Кеша слушает доклад $\#i$ на секции v (где v принимает значения 1 или 2). Тогда можно провести вычисления по формулам:

$$dp[1][i] = max(dp[1][i-1] + f[i], dp[2][i-2] + s[i])$$
$$dp[2][i] = max(dp[2][i-1] + s[i], dp[1][i-2] + f[i])$$

и выбрать максимальное из $dp[1][n]$ и $dp[2][n]$.

Останется только восстановить путь, по которому мы пришли к этому значению, это можно сделать обратным просмотром.