Разбор задачи «Самокат»

Подзадача 1 решается просто. Поскольку в маршруте присутствуют только участки одного типа, то при условии, что весь запас энергии в аккумуляторе будет истрачен, минимальное и максимальное количество усилий будут совпадать: Кеше не придётся делать выбор. Чтобы вычислить количество усилий, нужно выяснить, какое расстояние Кеше останется проехать. В случае, когда $p \neq 0$, количество усилий будет определяться как $max(0, p - q) \cdot a$, а в случае, когда $m \neq 0$ — как $max(0, m - q/2) \cdot b$.

Подзадача 2. Ограничения в подзадаче таковы, что можно просто перебрать все возможные варианты. Конечно, перебор имеет смысл, если ёмкости аккумулятора не хватит на весь путь Кеши — но это легко проверить.

Далее можно действовать, например, следующим образом. Сначала предположим, что Кеша будет включать электродвигатель всегда, когда поднимается в гору. Если ему не хватит ёмкости аккумулятора на все такие участки, то ему придётся приложить $(m - q/2) \cdot b$ усилий на преодоление оставшейся части маршурта в гору и $p \cdot a$ усилий на преодоление ровной части маршрута. Если же ёмкости аккумулятора будет достаточно, чтобы проехать все участки, идущие в гору, то Кеша потратит $(p - (q - 2 \cdot m)) \cdot a$ усилий на преодоление ровной части маршрута.

Эти формулы несложно объединить, если ввести величину qm, которая будет совпадать с q, если $q < 2 \cdot m$, и будет равна $2 \cdot m$ в противном случае. Тогда количество усилий будет записано как $(m - qm/2) \cdot b + (p - (q - qm)) \cdot a$. Пока будем считать это количество и максимально возможным, и минимально возможным.

Теперь предположим, что одну единицу длины маршрута в гору Кеша проедет с выключенным электродвигателем. Это «высвободит» 2 единицы ёмкости аккумулятора, которые мы «передадим» на движение по ровному участку. Теперь количество усилий может быть вычислено как $(m - (qm - 2)/2) \cdot b + (p - (q - (qm - 2))) \cdot a$. Сравним полученный результат с текущими минимумом и максимумом и обновим одно из текущих значений.

Будем продолжать этот «процесс» передачи очередных 2 единиц ёмкости с участка, идущего в гору, на ровный участок и будем каждый раз вычислять, какое количество усилий понадобится, обновляя при необходимости минимум и максимум. Процесс прекратится либо когда на ровный участок будет передана вся ёмкость, либо когда ёмкости на ровный участок будет передано столько, чтобы покрыть его весь.

Заметим, что передавать менее 2 единиц ёмкости смысла не имеет — если передача улучшает ситуацию, то передавать надо как можно больше.

Подзадача 3.

В общем случае, конечно, перебор всех вариантов не уложится по времени. Однако, рассматривая процесс, описанный в подзадаче 2, можно попробовать ответить на вопрос «Когда выгодно передавать ёмкость?»

Если 2 единицы ёмкости тратятся на преодоление одной единицы длины пути в гору, это экономит для Кеши b усилий. Но при этом они не будут потрачены на преодоление ровного участка, т.е. Кеша потратит на нём $2 \cdot a$ усилий. Если $b > 2 \cdot a$, то Кеша потратит меньше усилий, если будет включать электродвигатель при подъёме в гору, если же неравенство не выполняется, то меньше усилий Кеша потратит, если будет включать электродвигатель на ровных участках. Эти рассуждения верны и для случая, когда мы хотим узнать максимальные затраты усилий.

Поскольку нам требуется вычислить и максимум, и минимум, то достаточно рассмотреть две крайние ситуации. Во-первых, надо посчитать, какое количество усилий потратит Кеша, если будет всегда (пока хватает ёмкости) включать электродвигатель при подъёме в гору, а на ровных участках будет включать его, если такая возможность останется. Во-вторых, надо посчитать, какое количество усилий потратит Кеша, если будет всегда (пока хватает ёмкости) включать электродвигатель на ровных участках, а при подъёме в гору — если такая возможность останется.

Теперь остаётся определить минимальное и максимальное из этих двух чисел и вывести их в качестве ответа.

Разбор задачи «Лавочки»

В подзадаче 1, очевидно, выбор направления, в котором сначала поплывёт Кеша, не имеет зна-

чения, и выводить эту часть ответа можно, даже выбирая `L` или `R` случайным образом.

Далее, как понятно, Кеща проплывёт чётное количество раз расстояние влево и расстояние вправо (эти количества могут быть равны, а могут отличаться друг от друга на единицу). Впрочем, нас в данной ситуации интересует только расстояние, которое фактически проплывёт Кеша, а его можно вычислить следующим образом. Посчитаем для каждой из лавочек остаток от деления $d_j = m \bmod (2 \cdot l_j)$ (вместо l_j можно было бы писать r_j). Если он окажется равным нулю, значит, Кеша сможет проплыть в точности то расстояние, которое он задумал. Если же остаток отличен от нуля, Кеше придётся проплыть ещё $2 \cdot l_j$, и при этом он превысит желаемое расстояние на $2 \cdot l_j - d_j$.

Далее останется только перебрать все лавочки и найти, для какой из них превышение расстояния окажется минимальным.

Теперь опишем полное решение задачи.

Сначала посчитаем количество полных кругов, которые гарантированно нужно будет проплыть Кеше. Это количество для лавочки $\#j$ составляет $k_j = m/(2 \cdot (l_j + r_j))$ (деление выполняется нацело).

Теперь выясним, какое расстояние Кеше ещё необходимо проплыть. Оно определяется остатком от деления: $d_j = m \mod (2 \cdot (l_j + r_j))$. Если этот остаток оказался равным нулю, задача уже решена, а Кеша может выходить из воды. Направление движения в начале плавания можно выбрать любым.

Если же остаток от деления оказался не равным нулю, то Кеше потребуется проплыть какое-то дополнительное расстояние.

Вообще говоря, у него три варианта на выбор: плыть ещё один полный круг, плыть влево, после чего вернуться и выйти из воды, плыть вправо, после чего вернуться и выйти из воды.

Добавление ещё одного полного круга приведёт к тому, что Кеша проплывет «лишнее» расстояние, определяемое разностью $(k_j + 1) \cdot 2 \cdot (l_j + r_j) - m$. Начальное направление движения при этом может быть любым.

Каким будет «лишнее» расстояние, если Кеша сначала поплывёт влево? Оно вычисляется как $k_j \cdot 2 \cdot (l_j + r_j) + 2 \cdot l_j - m$. Заметим, что, в отличие от ситуации с дополнительным полным кругом нет гарантии, что эта разность окажется неотрицательной. Но если это так, то результат из предыдущего пункта с полным кругом улучшится.

Аналогично можно посчитать «лишнее» расстояние, если Кеша сначала поплывёт вправо. Если разность $k_j \cdot 2 \cdot (l_j + r_j) + 2 \cdot r_j - m$ будет неотрицательной, следует проверить, нельзя ли получить ещё более близкий к m результат.

Выполним такие вычисления для каждой лавочки, сохраняя в соответствующих массивах наилучшее приближение к желаемому расстоянию и символ, означающий направление движения Кеши при старте. Далее отыщем среди всех лавочек наиболее подходящую — для которой разность между желаемым и фактическим расстояниями будет минимальной.

Разбор задачи «Форточка»

В задаче требовалось выполнить аккуратную симуляцию описанного процесса. Ограничения позволяют анализировать состояние окна и накопленный дискомфорт действующих лиц каждую единицу времени.

Определим три возможных состояния окна — «открыто настежь», «открыто как фрамуга» и «закрыто». Изначально окно находится в состоянии «открыто как фрамуга».

Заведём переменную, содержащую количество времени, прошедшего от начала рабочего дня. При увеличении её на единицу будем обновлять значения дискомфорта Любителя Свежего Воздуха и Противника Сквозняков. Если дискомфорт кого-либо из них достигнет предельного значения, это может привести к изменению состояния окна, но лишь в том случае, если к окну отправится только один из сотрудников. Поэтому, прежде чем изменять состояние окна в очередной момент времени, нужно проверить, что сотрудники не придут к нему одновременно (и тогда уже состояние окна изменяться не будет).

При изменении состояния окна будем фиксировать, кто из сотрудников выполнил это изменение (для этого удобно использовать четыре переменные). Когда процесс будет завершён (по любой из причин: оба сотрудника подошли к окну одновременно или же закончился рабочий день), останется только вывести накопленные в них значения.

Разбор задачи «Танцевальный марафон»

В этой задаче можно воспользоваться техникой «двух указателей».

Сначала посчитаем, какого результата может добиться Кеша, если начнёт танцевать в момент начала первой композиции. Этот подсчёт выполняется непосредственно, проходом по соответствующим массивам.

Будем считать, что первый «указатель» находится сейчас на первой композиции (с неё Кеша начал), а второй «указатель» находится на последней композиции, которую Кеша сможет исполнить полностью (но не левее первого «указателя»).

Предположим, что Кеша исполнит k танцев, а в $k+1$ танце продержится какое-то количество минут. Эти данные мы можем использовать при подсчёте результата, которого сможет достичь Кеша, если начнёт танцевать в момент начала второй композиции. Действительно, мы знаем, сколько энергии потратил Кеша на исполнение первого танца, поэтому «вернём» эту энергию Кеше. Также (технически) удобно «вернуть» Кеше энергию, которую он израсходовал на неполный танец.

Понятно, что, начав со второй композиции, Кеша в худшем случае вновь остановится на композиции $\#k$. А энергия, которую он потратит, исполняя танцы со 2 по $\#k$, вычисляется так, как описано выше. После этого можно перераспределить энергию, которую мы «вернули» Кеше: вполне возможно, её хватит ещё на некоторое количество танцев и минут в неполном танце.

Таким образом, мы переместим первый «указатель» на вторую композицию, а второй «указатель» — на какую-то из композиций, начиная с k. Полученный результат сравним с результатом для первой композиции, и будем сохранять лучший из них.

Далее будем действовать аналогично, каждый раз перемещая на одну композицию вперёд первый «указатель», а второй — на какую будет получаться. Будем сохранять лучший из результатов и номер композиции, для которой этот результат достигается. Определённой аккуратности требует обработка танцев, исполненных не полностью.

Заметим, что это не единственно возможный способ решения (можно использовать бинпоиск).

Разбор задачи «Мороженое»

Будем решать задачу методом динамического программирования.

Каждый участник (кроме первого и последнего) может обменяться мороженым либо с соседом слева, либо с соседом справа, либо же ничего не будет делать. В действительности, нам достаточно рассматривать только обмен с соседом слева: обмен между участниками $\#k$ и $\#(k+1)$ можно рассматривать и как обмен с соседом справа для участника $\#k$, и как обмен с соседом слева для участника $\#(k+1)$. Таким образом, количество возможных действий для каждого участника будет равным двум (а для первого участника останется возможным лишь одно действие, поскольку соседа слева у него нет).

Будем хранить в $dp[0][i]$ минимально возможное максимальное недовольство среди всех участников до $\#i$ включительно, если участник $\#i$ не стал меняться мороженым со своим соседом слева. А в $dp[1][i]$ будем хранить минимально возможное максимальное недовольство среди всех участников до $\#i$ включительно, если участник $\#i$ поменялся мороженым с участником $\#(i-1)$.

Теперь мы можем вычислить для участника $\#(i+1)$ значения $dp[0][i+1]$ и $dp[1][i+1]$:

$$dp[0][i+1] = \max\left(\min(dp[0][i],\, dp[1][i]),\, |m[i+1] - s[i+1]|\right)$$

$$dp[1][i+1] = \max\left(\min(dp[0][i-1],\, dp[1][i-1]),\, |m[i] - s[i+1]|,\, |m[i+1] - s[i]|\right)$$

.

Действительно, если мы предполагаем, что в оптимальном ответе участник $\#(i+1)$ не будет меняться мороженым, то нам нужно выбрать лучший возможный ответ для всех участников до $\#i$ включительно, после чего сравнить его с недовольством участника $\#(i+1)$. Если же мы предполагаем, что участник $\#(i+1)$ поменяется мороженым со своим соседом слева, то возможным ответом для нас может быть лишь тот, что хранится в $dp[0][i]$: для реализации обмена участник $\#i$ не должен участвовать в другом обмене.

Поскольку нам нужно ещё и предъявить план обменов, заведём дополнительный массив $from[][]$. В элемент $from[0][i+1]$ будем записывать 0, если при вычислении $dp[0][i+1]$ был выбран элемент

$dp[0][i]$, и 1, если был выбран элемент $dp[1][i]$. Аналогично, в элемент $from[1][i+1]$ будем записывать 0, если при вычислении $dp[1][i+1]$ был выбран элемент $dp[0][i-1]$, и 1, если был выбран элемент $dp[1][i-1]$.

По завершении вычислений пройдём по массиву $from[][]$ с конца и восстановим порядок обменов.

Заметим, что это не единственный способ получить план обменов.