Задача 1. Папины дочки

Построим ориентированный граф дочерей, в котором ребро из дочери x в дочь y означает, что есть требование, согласно которому у дочери y должно быть не меньше конфет, чем у дочери x. Заметим, что если несколько дочерей образуют цикл, то у всех дочерей в этом цикле количество конфет должно быть одинаково.

Во второй и четвертой подзадачах нет одинаковых коробок конфет. Это значит, что циклы в графе запрещены. Для этих подзадач достаточно было написать алгоритм топологической сортировки. Во второй подзадаче можно было использовать неэффективный алгоритм. Для четвертой подзадачи следовало написать, например, алгоритм Тарьяна, работающий за $O(N + M)$. При наличии цикла ответ `NO`. Если циклов нет, раздаем коробки по возрастанию конфет в порядке, полученном после топологической сортировки. Асимптотика такого решения — $O(N + M + K \log K)$.

Для решения третьей и пятой подзадач найдем компоненты сильной связности и построим конденсацию графа. Аналогично, для третьей подзадачи можно было использовать неэффективный алгоритм, для последней следовало написать, например, алгоритм Косарайю, работающий за $O(N + M)$. Внутри одной компоненты количество конфет у всех должно быть одинаково. По условию нам гарантируется, что компоненты сильной связности будут линейно упорядочены, то есть порядок их обхода по возрастанию будет единственным. Разобьем коробки на группы с одинаковым количеством конфет и упорядочим их по возрастанию количества конфет. Далее можно идти двумя указателями по упорядоченным компонентам сильной связности и по группам коробок. Пока не встретим группу коробок, количество коробок в которой не меньше размера компоненты, двигаем указатель коробок. Далее раздаем всей компоненте одинаковые коробки. Если еще остались коробки в группе, указатель коробок не сдвигаем, так как в следующей компоненте у дочерей может быть такое же количество конфет. Асимптотика такого решения — $O(N + M + K \log K)$.

Задача 2. Антошка, Антошка, пойдём копать картошку

Для решения этой задачи достаточно было сделать несколько наблюдений:

— Забирать больше одного клубня из куста никогда не выгодно.

Потому что единственная польза от убирания клубней из куста — смена четности. Но, чтобы поменять четность, достаточно взять один клубень. Если же мы достанем больше, то просто потеряем в итоговом ответе.

— Забирать клубни из «нечетных кустов» никогда не выгодно.

Забрав клубень из «нечетного куста», мы перескочим следующий и перейдем к кусту через один (таким образом, посетив i-ый и $(i + 2)$-ой кусты). Но зачем это делать, если можно прийти в следующий, и, если он четный, забрать клубень из него? Тогда мы посетим i-ый, $(i+1)$-ый, $(i+2)$-ой кусты, также потратив всего один клубень, что, определенно, лучше.

— Забирать клубень из «четного куста» можно всегда, без потери в ответе. Исключение составляет только «четный куст», стоящий в самом конце ряда.

Таким образом, решение задачи — сумма всех чисел минус количество четных чисел, не считая последний куст в ряду.

Задача 3. Прогулка в парке

Зафиксируем значение мекс множества X. Для того, чтобы его достичь, в множестве должны присутствовать все числа от 0 до $x - 1$ включительно. Значит, «стоимость» получения такого значения мекс будет равна $\frac{x(x-1)}{2}$. Тогда в качестве последнего числа можно добавить $K - \frac{x(x-1)}{2}$ (нет смысла добавлять больше одного числа, поскольку это не увеличит мекс или наибольшее число в множестве). Тогда в качестве максимального значения будет либо $x - 1$, либо $K - \frac{x(x-1)}{2}$.

Первый случай можно разобрать отдельно, взяв в качестве x наибольшее число, для которого $\frac{x(x-1)}{2} \leqslant K$. Сделать это можно, например, с помощью двоичного поиска.

Во втором случае, целевая функция будет равна $\left(K - \frac{x(x-1)}{2}\right)x = Kx - \frac{x^2(x-1)}{2}$. Это кубическая функция с отрицательным коэффициентом перед x^3, а, значит, максимум этой функции достигается в области экстремума функции (из-за ограничения на целые значения x). Для этого найдём производную функции и приравняем к 0: $\left(Kx - \frac{x^2(x-1)}{2}\right)' = K - \frac{3x^2}{2} + x = 0$. Поскольку нас интересует положительные значения x, то значение, область вокруг которой мы должны рассмотреть, равно $\frac{1+\sqrt{6K+1}}{3}$.

Останется только взять максимум из двух вариантов. В качестве области для проверки можно выбрать $[x-10, x+10] \cap [1, K]$.

Задача 4. Шахматный турнир

Для решения второй подзадачи можно было полностью промоделировать турнир за $O(2^N)$.

Для решения третьей и четвертой подзадач заметим, что все туры можно разделить на четыре группы:

1. туры верхней сетки;
2. первый тур нижней сетки, а также туры нижней сетки, в которых играют только победители прошлых туров нижней сетки;
3. туры нижней сетки, в которых играют победители прошлого тура нижней сетки с проигравшими одного из прошлых туров верхней сетки;
4. финальный тур.

Заметим, что в нижней сетке туры второй и третьей группы чередуются по дням. В первой группе N туров, во второй и третьей по $N-1$ туров, в четвертой — 1 тур.

Всего игровых дней $\max(N, 2(N-1)) + 1 = 2(N-1) + 1 = 2N$.

Теперь научимся, зная номера двух игроков, определять, играют ли они в туре какой-нибудь из четырех групп.

В финальном туре всегда играют только игроки 1 и 2. Они же всегда играют в первый день. И это единственная пара, у которой будет две игры. У остальных будет либо 1 игра, либо 0.

Для нахождения критериев для первых трех групп введем переменную $\texttt{diff} = y - x$, где x и y номера игроков, идущие по возрастанию.

Для первой группы заметим, что в день i у всех игроков $\log(\texttt{diff}) = i - 1$ и меньшие номера игроков в играх идут через $2^i = 2 \cdot \texttt{diff}$, начиная с 1. То есть, критерием наличия игры в первой группе будут условия:

1. $\log(\texttt{diff})$ — целое число, $\log(\texttt{diff}) + 1$ — номер дня;
2. $x - 1$ делится нацело на $2 \cdot \texttt{diff}$.

У второй группы в день i у всех игроков $\log(\texttt{diff}) = i/2$ (эти туры проходят только в четные дни) и меньшие номера игроков в играх идут через $2^{i/2+1} = 2 \cdot \texttt{diff}$, начиная с 2. То есть, критерием наличия игры во второй группе будут условия:

1. $\log(\texttt{diff})$ — целое число, большее нуля, $2 \cdot \log(\texttt{diff})$ — номер дня;
2. $x - 2$ делится нацело на $2 \cdot \texttt{diff}$.

У третьей группы в день i у всех игроков $\log(\texttt{diff} + 1) = (i-1)/2$ (эти туры проходят только в нечетные дни) и меньшие номера игроков в играх идут через $2^{(i-1)/2+1} = 2 \cdot (\texttt{diff} + 1)$, начиная с 2. То есть, критерием наличия игры в третьей группе будут условия:

1. $\log(\texttt{diff} + 1)$ — целое число, $2 \cdot \log(\texttt{diff} + 1) + 1$ — номер дня;

2. $x - 2$ делится нацело на $2 \cdot (\texttt{diff} + 1)$.

Если пара не подошла ни под один критерий, она в турнире не пересечется.

Для решения четвертой подзадачи не забудьте использовать 64-битные типы данных.

Задача 5. Стол и вода

Так как скорость распространения воды постоянна и не меняется при прохождении через лист, можно заметить, что область распространения воды в момент времени t представляет собой круг радиуса t сантиметров с центром в начальном положении стакана. Остаётся определить, с какими листами данный круг имеет общие точки. Для этого для каждого листа достаточно проверить расстояния от центра (точки) до каждой из его сторон (отрезка), а также учесть случай, когда центр находится внутри листа (тогда круг будет пересекаться с листом в любой момент времени)

Как найти расстояние от точки до отрезка? Понятно, что оно не больше, чем минимальное из расстояний от точки до концов отрезка. Меньше оно может быть, когда основание перпендикуляра, проведенного из точки на прямую, на которой лежит отрезок, находится в пределах отрезка. Обозначим точку как O, а отрезок — AB. Тогда это условие соответствует тому, что $\vec{BO} \cdot \vec{BA} \geqslant 0$ и $\vec{AO} \cdot \vec{AB} \geqslant 0$ (соответствующие вектора образуют острый или прямой угол). Длину перпендикуляра, которая и будет расстоянием от точки до отрезка, можно найти как $\frac{|[\vec{AO} \times \vec{AB}]|}{|AB|}$

Подзадача 2

Для каждого запроса можно проверить каждый прямоугольник. Асимптотика такого решения $O(N \cdot Q)$

Подзадача 3

На плоскости есть четыре области, при нахождении точки в которых расстояние до стороны прямоугольника будет меньше, чем до любой его вершины. В случае, если стороны параллельны осям координат, эти области задаются независимыми неравенствами на x и y, а расстояние сводится к разности координат.

Подзадача 4

Заметим, что для каждого листа существует момент времени t, начиная с которого для каждого $t' \geqslant t$ область распространения воды будет касаться его. Можно вычислить такие моменты времени для каждого листа и отсортировать. Тогда можно отвечать на каждый запрос бинарным поиском. Асимптотика решения $O(Q \cdot logN)$.

Задача 6. Эксперимент

Не трудно заметить, что структура построенного лабиринта представляет собой неориентированное дерево.

Подзадача 2

Для решения второй подзадачи можно было для каждого запроса модификации запускать обход в глубину (dfs) по всему дереву и изменять уровни теплоты вершин в соответствии с условием задачи. Асимптотика такого решения $O(N \cdot Q)$

Подзадача 3

Давайте заведем переменную sum, которая будет хранить сумму всех x из запросов второго типа. Тогда ответ на запрос первого типа это $sum - distances$, где $distances$ — сумма расстояний от вершины запроса до всех вершин, на которых были модификации. Благодаря удобной структуре дерева в этой подзадаче можно легко вычислить $distances$:

Если v центральная: $distances = cntall - cnt[v]$;

Если v не центральная: $distances = 1 \cdot cnt[center] + 2 \cdot (cntall - cnt[center] - cnt[v])$; где $center$ — индекс вершины дерева, соединенной со всеми остальными; $cnt[v]$ — количество запросов модификации, сделанных в вершине v; $cntall$ — количество вообще всех запросов модификации. Асимптотика решения $O(N + Q)$.

Подзадача 4

В данной подгруппе задача вырождается к операциям на массиве. Аналогично предыдущей подзадаче для запросов первого типа нам нужно находить сумму расстояний от вершины запроса до всех когда-либо измененных (только теперь на массиве). Заметим, что для всех элементов, меньших текущего, такое расстояние можно посчитать как $dist1 = cur \cdot cntless - sumless$. А для всех

элементов, больших текущего, как $dist2 = sumgreater - cur \cdot cntgreater$. Таким образом, нам нужно вычислять количество чисел и их сумму, больших/меньших данного. Для этого можно использовать структуру данных, которая умеет прибавлять в точке и находить сумму на префиксе/суффиксе, например, дерево Фенвика. Заведем две такие структуры. В первую структуру на каждом запросе модификации будем прибавлять 1 на позицию вершины, во вторую структуру будем прибавлять саму позицию на позицию. Тогда с помощью первой структуры мы можем посчитать количество чисел меньших/больших данного, а с помощью второй структуры, суммы чисел меньших/больших данного. Асимптотика решения $O(Q \cdot logN)$.

Подзадачи 5, 6

В этой подзадаче можно было использовать корневую декомпозицию по запросам. Будем хранить ответ для каждой вершины — $ans[v]$ и обновлять его каждые $sqrt(Q)$ запросов. Внутри блока запросов можно запоминать все вершины, на которых были изменения, а при ответе на запрос находить вклад каждой из них с помощью подсчета расстояния, используя LCA, и получать общий ответ, суммируя с $ans[v]$. Эта часть будет работать за $O(Q \cdot sqrt(Q) \cdot LCA)$.

Когда блок запросов подходит к концу, нужно обновлять массив ans. Для этого зафиксируем корень дерева и для каждого поддерева с помощью dfs вычислим количество модификаций в нем — $sz[v]$ (внутри текущего блока запросов, разумеется). Далее посчитаем суммарное изменение теплоты в корневой вершине (операциями текущего блока) — $diff$. Далее можно заметить, что для любого ребенка to изменение теплоты будет $diff + sz[to] - (all - sz[to])$, где all — количество запросов модификации в текущем блоке. Благодаря этой формуле и обходу в глубину можно обновить значения на всем дереве за $O(N)$.

В зависимости от способа нахождения наименьшего общего предка, решение может работать за $O(N \cdot sqrt(Q) + Q \cdot sqrt(Q) \cdot logN)$ или за $O((N + Q) \cdot sqrt(Q))$ и проходить (или не проходить) 5 и 6 подзадачи.

Подзадачи 7, 8

В этой подзадаче можно было использовать центроидную декомпозицию. Построим дерево центроидов. Пусть каждая вершина в дереве центроидов помнит количество модификаций в своем поддереве — $cnt[v]$ (в дереве центроидов), а также знает суммарный вклад в себя всех модификаций из поддерева — $val[v]$. Помимо суммарной статистики также нужно помнить, сколько каждый конкретный из детей-центроидов вносит количества и теплоты — $cnt[v][child]$, $val[v][child]$. При запросе модификации это легко обновляется рекурсивным подъемом по дереву центроидов за непосредственно высоту этого дерева — $O(logN)$.

Для ответа на запрос первого типа также можно рекурсивно подниматься по дереву центроидов и вычислять ответ. На каждом уровне рекурсии будем прибавлять в ответ вклад всех других детей-центроидов, кроме того, откуда мы пришли. Благодаря устройству дерева центроидов в итоге мы получим нужный нам ответ. При подъеме по дереву центроидов к ответу на каждом шаге нужно прибавлять следующую величину: $val[c] - val[c][prev] - dist(v, c) \cdot (cnt[c] - cnt[c][prev])$, где v — вершина запроса, c — текущий центроид, $prev$ — предыдущий центроид. Здесь $val[c] - val[c][prev]$ — вклад всех модификаций из других детей-центроидов, кроме того, из которого мы пришли. Также мы отнимаем $dist(v, c) \cdot (cnt[c] - cnt[c][prev])$ — расстояние от центроида до вершины запроса, умноженное на количество запросов из других детей-центроидов, чтобы учесть потерю теплоты при отдалении от источника.

В зависимости от способа нахождения наименьшего общего предка решение может работать за $O(Q \cdot logN)$ или за $O(Q \cdot logN \cdot logN)$ и проходить (или не проходить) 7 и 8 подзадачи.

Задача 7. Тестирование уровней

Пусть при проходе через i-ые врата корабль находится в отрезке $[A, B]$, и при этом гравитация направленна вниз. Тогда, если время для того, чтобы добраться до следующих врат равно T, корабль сможет оказаться в отрезке $[A - \frac{a_y T^2}{2}, B + \frac{a_y T^2}{2}]$, в зависимости от момента переключения гравитации.

Посчитаем с помощью динамического программирования на каком отрезке и с каким направлением гравитации может оказаться корабль при пересечении очередных врат dp[номер врат][направление гравитации] = отрезок. Тогда, если отрезки последнего слоя динамики не пустые — ответ существует. Восстановим его следующим образом.

Зафиксируем нижнюю достижимую точку последних врат x и направление гравитации (например, отрицательное). Тогда, если гравитация не изменялась, в эту точку можно было попасть из точки $x + \frac{a_y T^2}{2}$ (в случае с положительным направлением гравитации был бы минус). Если это точка принадлежит отрезку динамики для предыдущего слоя — перейдём в него. Иначе — нужно изменить гравитацию. Если точка x находится и внутри отрезка предыдущих врат, то достаточно поменять гравитацию на середине пути. Иначе, положим отрезок предыдущих врат за $[L, R]$. Если $x < L$, то выберем такое время, что корабль попадёт в x из L. Иначе, будем считать что корабль попал в x из R.

Для того, чтобы найти время, достаточно решить следующее уравнение относительно t_1: $x_0 + \frac{a_y t_1^2}{2} + \frac{a_y (T-t_1)^2}{2} = x$, где x_0 — начальная позиция (знаки зависят от направления гравитации).

Восстановив таким образом ответ, останется только вывести его.