Башни 3.0

В первой подзадаче можно было явно построить граф, где ребро (u, v) проведено, если из u-й башни можно прыгнуть на v-ю. После этого достаточно из каждой вершины запустить DFS и посчитать множество посещённых вершин. Пусть m – количество рёбер в графе, тогда n DFS-ов работают за $O(nm)$. Чтобы ответить на запрос, достаточно перебрать индекс вершины на отрезке $[l, r]$ и проверить, сколько из них подходят под условие, используя предпосчитанную достижимость. Решение работает за $O(nm + qn)$, поскольку $m = \Theta(n^2)$ (нижняя оценка когда высоты башен возрастают), алгоритм работает за $O(n^3 + qn)$ и проходит при ограничениях $n \leqslant 100$, $q \leqslant 1000$.

Для решения второй подзадачи мы можем для каждой вершины v хранить битсет b_v достижимых вершин. Более формально, битсет вершины v имеет размер n, и i-й бит равен 1, если вершина i достижима из вершины v, 0 иначе. Сначала научимся эффективно строить граф. Заметим, что ребро (u, v) существует тогда и только тогда, когда v-я башня является одним из максимумов на отрезке, в котором u — середина, а v является одним из концов. Мы можем предпосчитать максимумы на каждом отрезке за $O(n^2)$, после этого можно проверить существование ребра за $O(1)$. Поскольку все башни имеют разную высоту, если мы будем рассматривать их в порядке убывания высоты, все рёбра из очередной башни будут вести в предыдущие. Это значит, что мы можем поддерживать инвариант что множество достижимых из всех рассмотренных вершин посчитано корректно, после чего для вычисления множества достижимых из очередной вершины v достаточно взять OR битсетов вершин, куда ведут рёбра из v. После того, как мы посчитали все битсеты, мы хотим эффективно отвечать на запросы. Для каждого l создадим битсет $filterL_l$, который содержит 1 в позициях, которые не меньше l. Аналогично создадим битсеты $filtetR$. Тогда ответ на запрос (u, v, l, r) — это количество единиц в AND b_u, b_v, $filterL_l$, $filterR_r$. Асимптотика такого решения — $O\left(\frac{n^3}{w}\right)$, где w — длина машинного слова.

В следующих четырёх подзадачах для того, чтобы отвечать на запросы, достаточно посчитать количество достижимых из каждой вершины.

В третьей подзадаче все высоты равны либо 1, либо 2. Из любой вершины существуют рёбра во все вершины со значением 2, поэтому их количество можно добавить к ответу для всех вершин. Кроме этого, нам нужно посчитать для каждой вершины v со значением 1 количество других вершин со значением 1, которые из неё достижимы. Можно заметить, что если рядом с v в массиве есть значение 2, мы не сможем попасть из v в другую вершину со значением 1. В противном случае, мы можем, используя рёбра длины 1, добраться до всех вершин в максимальном по включению отрезке из 1, содержащем v, и ни одна другая вершина со значением 1 не достижима. Используя эти наблюдения, можно получить линейное решение.

Четвёртая подзадача была идентична третьей, но у нас дополнительно могли быть башни высотой 3. Сначала заметим, что все такие башни достижимы из всех остальных. Для каждой башни высотой 2 предпосчитаем следующую башню высоты 2 слева и справа, а также ближайшую башню высоты 3. Пусть башни u и v — башни высоты 2, и существует ребро из u в v. Пусть $w_1, w_2, \ldots, w_k$ — башни высоты 2 на отрезке от u до v. Заметим, что вместо ребра от u до v, мы могли бы пойти по пути $u \to w_1 \to w_2 \to \ldots \to w_k \to v$. Это значит, что из всех рёбер между башнями высоты 2 можно оставить только те, которые ведут в ближайшую слева и справа башню, которые мы предпосчитали. Это позволяет легко посчитать для каждой вершины v высоты 2 значения l_v — количество достижимых слева, и r_v — справа, а также итоговый ответ. Теперь рассмотрим башни v высоты 1. Используя критерий из предыдущей подзадачи, мы можем понять какое множество башен высоты 1 достижимо — это может быть либо только исходная башня, либо максимальный по включению отрезок башен высоты 1, обозначим его за $[a, b]$. Нам осталось посчитать количество достижимых башен высоты 2. Заметим, что в любом пути из v первая башня высоты больше 1 должна быть либо $a - 1$-й, либо $b + 1$-й. Тогда достаточно добавить к ответу l_{a-1}, если вершина $a - 1$ достижима, и r_{b+1}, если вершина $b + 1$ достижима. Решение работает за $O(n)$.

Сделаем наблюдение, полезное для полного решения. Мы хотим понять из каких вершин есть ребро в вершину v. Пусть gl_v — ближайший элемент в массиве слева от v, больший v, gr_v — аналогичный элемент справа. Если к некоторой вершине u есть более близкий чем v элемент, больший v-го, его можно выбрать из множества $\{gl_v, gr_v\}$. Это значит, что множество вершин, из которых

проведено ребро в v — отрезок $\left[\left\lfloor\frac{gl_v+v}{2}\right\rfloor+1, \left\lceil\frac{gr_v+v}{2}\right\rceil-1\right]$.

Используя этот факт, можно получить решение пятой подзадачи. Пусть в вершину v ведут рёбра с отрезка $[c_v, d_v]$. Мы переберём v, посчитаем отрезок вершин, из которых достижима v, и добавим 1 к ответу на этом отрезке. Чтобы посчитать отрезок вершин, откуда достижима v, будем хранить текущий ответ (изначально это $[c_v, d_v]$), и отрезок рассмотренных вершин (изначально это $[v, v]$). После этого, пока отрезок рассмотренных вершин и ответ не совпадут, будем расширять отрезок рассмотренных вершин на 1 (слева или справа). Пусть u — вершина, которую мы только что добавили в отрезок рассмотренных, она должна принадлежать отрезку ответа, и мы можем объединить отрезок ответа с отрезком $[c_u, d_u]$. Когда отрезок ответа и рассмотренных вершин совпадут, не будет существовать вершин вне отрезка, из которых есть ребро в отрезок, следовательно это все вершины, из которых достижима v. Решение работает за $O(n^2)$. Это решение можно соптимизировать до решения шестой подзадачи: чтобы расширить текущий отрезок $[l, r]$, мы можем найти минимум c_v и максимум d_v на $[l, r]$ с помощью sparse table, и заменить границы на этот максимум и минимум. Можно доказать, что это решение работает за $O(n \log n)$.

Чтобы продвинуться дальше, нам понадобится ещё несколько наблюдений. Заметим, что поскольку ребро (i, j) проведено тогда, когда j-й элемент является максимумом на отрезке, где i является серединой, а j – одним из концов, из наличия ребра (i, j) следует наличие рёбер (k, j) для $i < k < j$. Аналогичное утверждение можно сделать для ребра (i, j) когда $j < i$. Пусть в некотором пути есть два соседних ребра (u, v) и (v, w), такие, что $w < u < v$. Тогда, используя свойство выше, мы получаем, что есть ребро (u, w). Делая такие замены для произвольного пути, можно добиться того, чтобы все рёбра пути вели либо только налево, либо только направо. Посчитаем колво достижимых по рёбрам налево из каждой вершины, аналогично по рёбрам направо, и, сложив результаты, получим ответ. Пусть L_i – самая правая вершина левее i, в которую есть ребро из i. Докажем, что можно оставить только рёбра $i \to L_i$, и количество достижимых слева не изменится. Действительно, пусть в каком-то пути мы пошли по ребру $i \to j$ и $j < L_i$. Мы знаем что тогда должно быть ребро $L_i \to j$, и $i \to j$ можно заменить на $i \to L_i$, $L_i \to j$. Делая такие замены, мы можем удалить все «плохие» рёбра. Тогда ответ для вершины i — это ответ для L_i плюс 1. Это значит, что для решения задачи достаточно уметь эффективно находить L_i для каждого i.

Мы умеем за линейное время посчитать gl_i и gr_i для всех i, используя стандратный алгоритм со стеком. Поскольку входящие рёбра в каждую вершину ведут из отрезка, который мы можем вычислить, задача о подсчёте L_i сводится к следующей: даны n отрезков $[l_i, r_i]$, для каждой позиции нужно вычислить минимальный индекс отрезка, который её покрывает. Существует множество решений этой задачи. Мы можем сделать сканлайн, который поддерживает set отрезков, которые покрывают данную позицию, тогда для каждого отрезка нужно создать событие «добавить i» в позции l_i и «удалить i» в позиции $r_i + 1$. Такое решение работает $O(n \log n)$. Также можно решать задачу с помощью дерева отрезков (тоже за $O(n \log n)$) или с помощью системы непересекающихся множеств (за $O(n \cdot \alpha(n))$).

Теперь нам нужно отвечать на запросы, используя знание о том, что можно посчитать массивы L и R, такие, что из вершины v достижимы вершины $v, L_v, L_{L_v}, \ldots, R_v, R_{R_v}, \ldots$. Мы можем представлять это как два дерева: построим «левый лес деревьев» из рёбер $v \to L_v$ и «правый лес деревьев» из рёбер $v \to R_v$. Тогда, множество достижимых из вершины v — это объединение пути до корня в левом лесу и в правом лесу.

В седьмой подзадаче есть дополнительное ограничение на запросы $u_i = v_i$. Мы можем разделить отрезок запроса на две части: то, что левее u и то, что правее u, и решать задачу отдельно для каждой из частей. Чтобы ответить на запрос про левую часть, посчитаем двоичные подъёмы на левом лесу. Чтобы найти количество вершин на пересечении пути до корня и отрезка $[l, r]$, найдём двоичными подъёмами первую вершину на пути, не большую r, и первую, строго меньшую l (по индексу). Тогда ответ — это разность глубин этих вершин. Решение работает за $O(n \log n)$.

В восьмой подзадаче дополнительное ограничение $r \leqslant \min(u, v)$ или $l \geqslant \max(u, v)$. Будем считать, что $l \leqslant r \leqslant u \leqslant v$, остальные случаи аналогичны. Заметим, что на отрезке $[l, r]$ нет вершин, которые находятся правее u или v, поэтому мы можем забыть про существование правого леса. В левом лесу пересечение множества достижимых — это путь от $\mathrm{lca}(u, v)$ до корня. Чтобы ответить на запрос, найдём LCA, а затем посчитаем пересечение пути до корня и отрезка аналогично

предыдущей подзадаче. Решение работает за $O(n \log n)$.

В девятой подзадаче отрезок каждого запроса содержит все позиции. Пусть $u < v$. Мы разделим его на три части: левее u, между u и v, и правее v. Для первой и третьей части мы уже умеем считать ответ как в прошлой подзадаче. Чтобы посчитать количество общих достижимых между u и v, понадобится наблюдение:

Пусть есть три вершины $i \leqslant j < k$, и $a_k < a_j$. Тогда k не достижима из i.

Доказательство: докажем это по индукции по убыванию j. Пусть для какого-то j путь из i до k нашёлся. Рассмотрим первую вершину w в этом пути, большую j. Заметим, что поскольку мы пришли в w из вершины не правее j, j-я позиция будет к этой вершине ближе, чем w-я, так что должно выполняться неравенство $a_w \geqslant a_j > a_k$. Применяя предположение индукции для $j := w$, получаем, что пути от i до k не существует, противоречие.

Из этого следует, что элементы пересечения множеств достижимых из u и из v в середине являются максимумами на отрезке $[u, v]$. Действительно, если это не так, пусть w достижима из u и из v, и на $[u, v]$ есть вершина s, такая что $a_s > a_w$. Пусть $s < w$, тогда w не может быть достижима из u по предыдущему утверждению, противоречие. Случай $s > w$ аналогичен.

Рассмотрим множество максимумов на отрезке $[u, v]$. Заметим, что подмножество, состоящее из максимумов, достижимых из u – это подотрезок пути из u в правом лесу. Это также подотрезок в массиве индексов максимумов, поскольку ребро не может переходить «через максимум». С помощью RMQ найдём максимум, с помощью двоичных подъёмов по лесам найдём отрезки достижимых максимумов, пересечём их, и вычислим количество элементов как разность глубин. Решение работает за $O(n \log n)$.

Десятая подзадача не сильно сложнее. Теперь у нас есть нетривиальные отрезки, мы будем делить их на три части аналогично девятой подзадаче. Обработку левой и правой части можно сделать как в восьмой подзадаче, подсчёт ответа для части посередине такой же, но нужно пересечь ответ с отрезком запроса (например, используя двоичные подъёмы). Решение по-прежнему работает за $O(n \log n)$.

BONUS: на самом деле при решении задачи для отрезка между u и v можно не использовать факт про максимумы и решать более общую версию задачи:

Даны два корневых дерева на вершинах $1, 2, \ldots, n$, нужно уметь быстро пересекать два пути до корня, исходящие из вершин запроса u и v, и выводить количество вершин в пересечении.

Это можно делать следующим образом: запустим dfs по первому дереву, поддерживая массив w, в котором в позиции i находится 1 тогда и только тогда, когда вершина i является предком текущей вершины в dfs и 0 иначе. При обходе дерева у нас происходят точечные изменения в этом массиве. Когда мы находимся в вершине u, мы можем ответить на все запросы вида (u, v): достаточно взять сумму на вертикальном пути второго дерева по текущему состоянию массива w. Мы свели задачу к задаче на дереве, где в вершинах меняются веса и нужно искать сумму на пути. Эта задача эквивалента задаче, где значения прибавляются ко всем вершинам в поддереве и запрос — найти значение в вершине. Это можно решать с помощью дерева отрезков на эйлеровом обходе. С учётом того, что во втором дереве все рёбра ведут направо, это решение можно адаптировать для последней подзадачи, где отрезок — это не весь массив.