Два проспекта

Сначала поймем как для двух ребер e_1, e_2 найти количество тугриков. Уберем оба ребра e_1, e_2. Каждая из пар, вершины которой в разных компонентах связности, заплатит хотя бы один тугрик. Если вершины какой-то пары оказываются в разных компонентах связности и при удалении только ребра e_1, и при удалении только ребра e_2, то такая пара заплатит два тугрика в любом случае. Таким образом, такие пары заплатят ровно два тугрика, а те пары, вершины которых только при удалении обоих ребер попадают в разные компоненты связности, заплатят ровно один тугрик. Количество этих пар каждого вида можно вычислить за $O(m+k)$, сделав поиск компонент связности с помощью dfs в каждом из случаев.

Этот алгоритм подсчета количества тугриков для заданной пары ребер дает нам решение первой подгруппы за $O(m^2(m+k))$ — давайте просто переберем все пары ребер.

Заметим, что граф при удалении обоих ребер должен стать несвязным, иначе ответ для этой пары ребер 0. Значит хотя бы одно ребро должно попасть на любое остовное дерево, например дерево dfs. Значит мы можем перебрать $O(nm)$ кандидатов пар ребер (одно из ребер перебираем на дереве dfs), что дает нам решение второй подгруппы за $O(nm(m+k))$.

Зафиксируем ребро дерева dfs, которое будет в паре. Удалим его. Заметим, что другое ребро, которое мы выберем в пару, должно быть мостом оставшегося графа (а иначе вообще не важно что это за ребро). Найдем все мосты этого графа. Для каждого моста можно легко посчитать ответ при его выборе суммарно за $O(m+k)$ для всех мостов. Это дает нам решение третьей подгруппы за $O(n(m+k))$.

Пусть ответ это пара ребер e_1, e_2, где e_1 лежит на дереве dfs. Давайте поймем, какими могут быть ребра e_1 и e_2. Если e_1 это мост, то e_2 тоже должно быть мостом (иначе при удалении e_1 ребро e_2 не будет мостом). Если e_1 это не мост, то каким может быть ребро e_2? Мы знаем, что e_2 должно быть мостом в графе без ребра e_1. Если e_2 не лежит на дереве dfs, то это должно быть единственное ребро, которое накрывает e_1, так как иначе e_2 не будет мостом графа без ребра e_1. Если e_2 лежит в дереве dfs, то множества внешних ребер, накрывающих e_1 и накрывающих e_2 должны совпадать. На самом деле, случай того, что e_2 это не мост графа без e_1 возможен, но тогда неважно какое ребро e_2 взять и количество тугриков будет равно количеству пар, вершины которых будут в разных компонентах связности после удаления e_1. Поэтому e_1 должно быть мостом.

Таким образом есть следующие случаи того, какими могут быть ребра e_1, e_2:

1. Оба ребра e_1, e_2 являются мостами графа.
2. Ребро e_1 является мостом графа, ребро e_2 не важно.
3. Ребро e_1 лежит на дереве dfs, а ребро e_2 это единственное внешнее ребро, которое его накрывает.
4. Ребра e_1, e_2 лежат на дереве dfs, для них множества внешних ребер, которые их накрывают совпадают.

Для каждого случая научимся находить пару ребер e_1, e_2, которая дает максимальный ответ и затем выберем среди них максимальную.

Сопоставим каждой паре вершин путь в дереве dfs между ее вершинами.

В случае 1 заметим, что для пары мостов e_1, e_2 ответ равен сумме количества путей, содержащих ребро e_1 и количества путей, содержащих ребро e_2. Давайте тогда для каждого ребра e дерева dfs посчитаем c_e — количество путей, содержащих e. Это можно сделать прибавив на каждом из k путей 1, что можно сделать с помощью LCA и прибавления ± 1 в концах вертикальных путей и префиксных сумм. Далее надо выбрать два моста e_1, e_2, таких что $c_{e_1} + c_{e_2}$ максимально, для этого надо просто найти два моста с максимальным значением c_e.

Заметим, что в случае 2 для моста e_1 ответ равен c_{e_1}. Тогда в этом случае оптимально взять мост с максимальным значением c_e. В качестве e_2 можно взять любое ребро.

В пятой подгруппе граф является деревом, поэтому все его ребра это мосты. Поэтому возможны только случаи 1, 2, которые мы уже научились решать за $O((n+k)\log n)$.

Для каждого ребра e дерева dfs посчитаем f_e, h_e — количество внешних ребер графа, накрывающих e, хеш множества внешних ребер графа, накрывающих e. Для того, чтобы посчитать хеш, можно каждому внешнему ребру сопоставить случайное 64-битное число, а хеш будет равен сумме этих чисел по всем накрывающим ребрам. Эта часть делается за $O(n)$.

Тогда в случае 3 нам интересны ребра e_1, для которых $f_{e_1} = 1$. Ответ для пары ребер e_1, e_2 тогда будет равен c_{e_1}. Переберем тогда все ребра дерева dfs, проверим критерий и найдем максимальный ответ.

Остался случай 4. Этот случай гораздо более сложный, чем случаи 1, 2, 3. В этом случае нам интересны пары ребер e_1 e_2, такие что $h_{e_1} = h_{e_2}$. Заметим, что если $h_{e_1} = h_{e_2}$, то ребро e_1 лежит на вертикальном пути от корня дерева dfs до e_2 (Н.У.О. ребро e_1 выше). Значит все ребра с одинаковым хешом лежат на некотором вертикальном пути. Тогда ответ для пары ребер e_1, e_2 будет равен количеству путей, которые накрывают ровно одно ребро среди e_1 и e_2. Заметим, что мы можем разделить каждый путь на два вертикальных и от этого ничего не поменяется. Поэтому мы можем считать, что все пути вертикальные. Началом пути будем называть более высокий конец, концом более низкий.

Будем делать обход в глубину дерева dfs и поддерживать дерево отрезков с операциями прибавления на отрезке и максимума на отрезке. При этом его ячейки будут соответствовать ребрам на пути от корня до текущего ребра e_2. В ячейке, соответствующей ребру e_1 будет записан ответ для пары ребер e_1, e_2, если $h_{e_1} = h_{e_2}$. Поймем, как пересчитывать это ДО и как бороться с парами ребер $h_{e_1} \neq h_{e_2}$.

Когда мы стоим в вершине v, последнее ребро e и мы переходим по ребру s нам для каждого ребра e_1 надо в ячейке для e_1 поменять число с ответа для (e_1, e) на ответ для (e_1, s). Какие пути могли накрывать/не накрывать ровно одно из e_1, e, но стать не накрывать/накрывать ровно одно из e_1, s?

1. Путь, начало которого это v и который накрывает ребро s, прибавляет 1 во все ячейки.

2. Путь, конец которого это v или конец которого лежит в поддереве v, но не в том куда идет ребро s, вычитает 1 для всех ребер e_1, которых этот путь не накрывал и прибавляет 1 для всех ребер e_1, которых этот путь накрывал. Назовем такие пути сложными.

Случай 1 простой: можно просто перебрать такие пути и сделать прибавление 1 на всем ДО.

Случай 2 намного сложнее. Количество сложных путей во всех v может быть большим и суммарно быть $O(nk)$. Заметим, что в случае, когда наш граф это цикл, количество сложных путей во всех v будет суммарно $O(k)$, это просто пути, которые кончаются в v. Поэтому такой же перебор как в случае 1 сработает. Поскольку у всех ребер хеши одинаковые в этом случае, это дает нам решение шестой подзадачи за $O((n + k) \log n)$.

Поймем, что делать с условием равенства хешей. Назовем кластером ребра дерева dfs с одинаковым хешом. Сопоставим каждому кластеру вертикальный путь от первого ребра кластера до последнего. Заметим, что на каждом таком пути внутри могут быть ребра с другим хешом. Заметим тогда, что либо пути для двух кластеров совсем не пересекаются (по ребрам), либо один из путей вложен в другой и при этом располагается между соседними вхождениями хеша для большего пути.

Тогда когда мы переходим по ребру e_2 в ДО будем искать минимум на отрезке от самого высокого ребра, с таким же хешом, что у ребра e_2. Но что делать с ребрами между ними, которые имеют другой хеш? Заметим, что делая такой обход, мы можем прибавить $-\infty$ на отрезке от последнего ребра с таким же хешом, что у e_2, до ребра e_2. Тогда мы перестанем рассматривать эти ребра в ответ. И при этом никакие из хешей этих ребер никогда больше не встретятся из-за устройства кластеров, поэтому мы дальше ничего не испортим.

Надо понять, как обновлять с помощью сложных путей из случая 2. Посмотрим на величину $c_e - c_s$. Это в точности количество сложных путей минус количество путей, которые начинаются в v и содержат s. Последнее количество мы знаем, тогда мы находим x — количество сложных путей. Добавим x ко всему ДО. Теперь нам нужно вычесть 2 на некоторых префиксах, заканчивающихся в началах сложных путей. Заметим, что мы можем этого не делать для сложного пути, если хеши с соответствующего префикса больше не встретятся ниже в обходе. Заметим, что каждый путь P

будет сложным и требующим вычитания на префиксе ровно в одной вершине v. Это так, потому что если выделить пути кластеров, которые ни в кого не вложены, то это будут не пересекающиеся вертикальные пути. И то, в какой внешний путь кластеров попало начало пути P, определяет единственную вершину v, в которой этот путь будет сложным и потребует прибавления на префиксе: эта вершина v это LCA конца внешнего пути кластера и конца пути P. Можно отдельным dfs для каждой вершины v найти какие сложные пути потребуют прибавления на префиксе в ней. Суммарное количество таких путей будет $O(k)$. В итоге мы разобрали случай 2 с помощью $O(n+k)$ операций с ДО.

В итоге получается полное решение за $O(m + (n+k)\log n)$.