Распределенная Матрица

Автор задачи и разработчик: Владимир Рябчун

Для решения первой подзадачи достаточно аккуратно реализовать то, что написано в условии. Для этого достаточно хранить граф, для каждого узла хранить время его присоединения, и при каждом запросе заново делать `dfs`. Асимптотика решения будет $\mathcal{O}(nm)$.

Решение второй подзадачи отличается лишь тем, что для каждой вершины следует хранить флаг, находится ли она в рабочем состоянии, и время последнего восстановления. При обходе `dfs` достаточно считать сумму времен последнего восстановления и длину пути, чтобы после посчитать суммарную ненадежность узлов. Чуть более эффективное решение заключается в том, чтобы сразу пройти по всем запросам, целиком построить дерево, и хранить для каждого узла предка и его глубину в дереве. Тогда ответ на запрос может заключаться в постепенном подъеме из двух вершин до корня. Время работы решения все еще $\mathcal{O}(nm)$.

Для решения третьей подзадачи можно выписать в каждой вершине момент времени, когда она была добавлена. Обозначим за $\mathrm{lca}(x, y)$ узел, являющийся наименьшим общим предком узлов x и y, а за $\mathtt{s}[v]$ — сумму чисел в узлах на пути от v до корня. Будем считать, что в данный момент запрос номер t, а количество различных узлов, от которых зависят узлы x и y, равно p. Тогда ответ в случае, если обе вершины добавлены, равен $t \cdot p - (\mathtt{s}[x] + \mathtt{s}[y] - \mathtt{s}[\mathrm{lca}(x, y)])$. Поддерживать и считать суммы чисел на путях, наименьшего общего предка и количество узлов на пути можно при помощи любой удобной техники, которая отвечает за запрос за время порядка $\mathcal{O}(\log n)$, например с помощью двоичных подъемов. Итоговая асимптотика будет $\mathcal{O}(m \log n)$.

В четвертой подзадаче необходимо делать ровно то же самое, что и в третьей, но надо быстро понимать, получает ли узел энергию. Если узел v отказал, то он и все его потомки никогда больше не будут получать энергию. Можно проставить отметки об отсутствии энергии обычным обходом в глубину, который не будет заходить в уже помеченные вершины. Пометки с вершин не снимаются, и каждую мы пометим не более одного раза, поэтому суммарное время всех обходов в глубину будет $\mathcal{O}(n)$.

Для полного решения необходимо заметить прекрасное свойство обхода дерева. Будем обходить дерево в глубину и записывать вершину в список, когда заходим в нее. Тогда все вершины поддерева v образуют в этом массиве непрерывный отрезок (от вхождения v до вершины, которая была в его конце, когда `dfs` покидал v). Будем в каждом элементе массива обхода хранить, сколько узлов-предков v отказали. Отказ вершины v приводит к увеличению всех значений на отрезке ее поддерева на 1. И, наоборот, восстановление приводит к уменьшению значений на отрезке на 1. Для проверки того, получает ли вершина энергию, достаточно посмотреть на значение в ее ячейке массива обхода. Если там записано 0, то нет ни одной вершины выше нее, которая отказала.

Авторское решение для реализации этих операций использует дерево отрезков. Величину $\mathtt{s}[v]$ можно поддерживать так же: при восстановлении узла значения $\mathtt{s}[u]$ поменяются только у потомков v на одинаковую величину. В итоге решение будет отвечать на каждый запрос за $\mathcal{O}(\log n)$ времени, соответственно итоговая сложность будет равна $\mathcal{O}(m \log n)$.