

Иерархия Паучьего сообщества

Автор задачи: Даниил Голов, разработчики: Даниил Голов и Даниил Орешников

Переформулируем задачу чуть более коротко: дано подвешенное дерево, и требуется «пометить» (назначить мнение А) некоторое количество вершин, чтобы количество ребер, на которых верхняя вершина помечена, а нижняя — нет, было максимально.

Эта задача всем в своем условии намекает на динамическое программирование по поддеревьям, более того, она очень похожа на задачу о максимальном независимом множестве на дереве, которая также решается с помощью динамики. В этой задаче мыведем такие же состояния:

- $\text{dp}[v][0]$ — максимальное значение беспорядка в поддереве вершины v , если при этом сама вершина v не помечена (имеет мнение В);
- $\text{dp}[v][1]$ — максимальное значение беспорядка в поддереве вершины v , если сама v при этом помечена.

Для начала найдем максимальное возможное значение беспорядка, которое мы можем получить, а затем восстановим, какие вершины для этого надо было пометить. Для этого считаем все связи из ввода и сделаем обход в глубину из вершины номер 1 — так мы сможем определить для каждого ребра, какая вершина является родителем («ментором»), а какая — ребенком («подчиненным»).

Теперь (можно и на выходе из `dfs`) будем считать значения динамики. Заметим, что если мы зафиксируем, помечена ли вершина, дальше можно оптимизировать поддерева ее детей независимо: действительно, от изменений в одном поддереве значения беспорядка в других не изменятся. Рассмотрим оба варианта: и когда v не помечена, и когда помечена.

1. Если вершина v не помечена, то независимо от состояний ее детей u_i , ни одно ребро (v, u_i) не будет вносить вклад в ответ. Таким образом, ответ будет просто складываться из максимальных возможных беспорядков в поддеревьях u_i , то есть

$$\text{dp}[v][0] = \sum_{u_i - \text{ребенок } v} \max(\text{dp}[u_i][0], \text{dp}[u_i][1]).$$

2. Если вершина v помечена, то некоторые из ее детей будут образовывать с ней пару, дающую вклад $+1$ в ответ. А именно, непомеченные дети. Тогда помеченный ребенок u_i дает вклад $\text{dp}[u_i][1]$, а непомеченный — $\text{dp}[u_i][0] + 1$. Остается только выбрать максимум:

$$\text{dp}[v][1] = \sum_{u_i - \text{ребенок } v} \max(\text{dp}[u_i][0] + 1, \text{dp}[u_i][1]).$$

Здесь хороший момент, чтобы вспомнить, что нас еще просят восстановить, какие вершины должны быть помечены, а какие — нет. Воспользуемся стандартной техникой для восстановления ответа: запомним, какой выбор привел нас к максимальному значению. А именно,ведем $\text{down}[v][0]$ и $\text{down}[v][1]$, которые будут содержать номера всех детей v , которые должны быть помечены, чтобы $\text{dp}[v][0]$ и $\text{dp}[v][1]$, соответственно, были максимальны.

Для их заполнения в формулах пересчета dp вместо того, чтобы просто брать максимум из двух опций, посмотрим, какая из них больше, и отметим соответствующую информацию в down . Так, если $\text{dp}[u_i][0] + 1 < \text{dp}[u_i][1]$, добавим u_i в $\text{down}[v][1]$, так как его выгоднее пометить, чем не пометить.

После того, как все вершины будут обработаны, максимальное значение беспорядка можно получить как $\max(\text{dp}[1][0], \text{dp}[1][1])$. Более того, зная, какая из двух этих величин больше, мы знаем, стоит ли отмечать вершину 1, чтобы достичь такого значения беспорядка. Запомним эту информацию, и спустимся по дереву: если очередной u_i лежит в соответствующем выбранному для первой вершины состоянию $\text{down}[1]$, то u_i тоже надо пометить, иначе — не надо. Так продолжаем спускаться по дереву, каждый раз восстанавливая состояния вершин.

Суммарно на подсчет такой динамики и спуск по дереву мы потратили время, пропорциональное количеству ребер в графе (по каждому ребру мы прошли константное число раз), поэтому весь этот алгоритм работает за $\mathcal{O}(n)$