Задача. Оптические каналы связи

Автор задачи: Андрей Станкевич

Задачу можно сформулировать так: требуется удалить часть ребер из дерева, чтобы степень каждой вершины была не больше k. При этом требуется оставить как можно больше ребер, а при равном количестве оставить ребра с максимальной суммой.

Большинство подзадач этой задачи требуют до той или иной степени применения технологии динамического программирования на дереве. Основная идея следующая: состояние — это вершина дерева, для поддерева которой решается задача, а также информация о решении в поддереве, которая необходима для использования решения для этого поддерева снаружи от него.

Подзадачи 1, 2 и 3

В этих подзадачах можно реализовать перебор всех ребер, которые мы оставляем за $2^n \cdot poly(n)$.

Подзадачи 1, 4 и 5

В этих подзадачах требуется найти паросочетание в дереве. Невзвешенная задача решается жадным алгоритмом, а для подзадачи 5 надо использовать динамическое программирование. Состояние динамического программирования для поиска максимального взвешенного паросочетания: (u, b), где u — вершина дерева, а b — флаг, можно ли задействовать вершину u в паросочетании с её родителем.

Подзадачи 6 и 7

В этих подзадачах оставшиеся ребра образуют пути. Каждый путь в дереве состоит из двух частей: вверх и вниз (одна из них может отсутствовать). Используем динамическое программирование, состояние: (u, b), где u — вершина дерева, а b — флаг, является можно ли продлить из вершины u путь вверх.

Рассмотрим теперь решения для произвольного k.

Заметим общую тенденцию решения подзадач для $k = 1$ и $k = 2$: флаг b в состоянии динамического программирования показывает, были ли все k ребер для корня поддерева u уже сохранены в выборанном оптимальном решении для поддерева, либо можно выбрать ребро из u в родителя.

Обобщим это для полного решения: состояние — пара (u, b), где флаг b показывает, были ли все k ребер для корня поддерева u сохранены для решения в поддереве.

Как пересчитать значение динамического программирования. Рассмотрим вершину u. Чтобы посчитать $(u, 1)$ надо выбрать не более $k-1$, а чтобы посчитать $(u, 0)$ — не более k ребер в поддерево из корня. Для каждого выбранного ребра uv к значению прибавляется $(v, 0)$, а для невыбранного $(v, 1)$.

Для вычисления значений воспользуемся вспомогательным динамическим программированием, подобным задаче о рюкзаке. Посчитаем $opt[i][j]$ — рассмотрели первые i ребер и взяли из них j, какое оптимальное решение можно получить таким образом. Переход выполняется за $O(1)$, мы либо берем очередное ребро, либо нет.

Вычисление массива opt для вершины, у которой t детей, работает за $O(tk)$, суммируя по всем вершинам получаем время $O(nk)$.

Значение основного динамического программирования для состояния $(u, 0)$ равно максимуму по j от 0 до k, а для состояния $(u, 1)$ — от 0 до $k - 1$.

Наконец, сформулируем четко, что мы храним в качестве значения динамического программирования. Если задача невзвешенная (подзадачи 1, 2, 4, 6, 8, 10, 12, где $w_i = 0$), значение динамического программирования — это количество взятых ребер. Если же задача взвешенная, то в качестве значения будем хранить пару из количества взятых ребер и суммарного их веса. Количество ребер будет первым критерием оптимизации, а их суммарный вес — вторым, при равенстве количества ребер.

Менее эффективные решения, которые реализуют динамическое программирование за $O(n^2k)$ или $O(nk^2)$ могут проходить подзадачи 8 и/или 9, и 10 и/или 11, соответственно.