Алгоритмы. Домашнее задание №5

Горбунов Егор Алексеевич

16 октября 2015 г.

Задача №1 (Максимальное по весу паросочетание за $\mathcal{O}(n)$)

Дан граф $G(V, E)$ и $w : E \to \mathbb{Z}$. Нужно придумать алгоритм нахождения веса W такого паросочетания M, что $W = \sum_{e \in M} w(e) \to \max$, если:

(a) G – дерево

Дерево G можно подвесить за любую вершину и получить корневое дерево (поиск в глубину от любой вершины это и делает). Будем тогда считать, что G – подвешенное дерево с корнем $root$. Решать поставленную задачу будем последовательно для всех поддеревьев дерева G начиная с листьев. Введём:

$A[v] =$ размер наибольшего паросочетания M в поддереве с корнем в вершине v, причём M содержит ребро (v, u), где $u \in Children(v)$

$B[v] =$ размер наибольшего паросочетания M в поддереве с корнем в вершине v, причём M не содержит рёбер (v, u), где $u \in Children(v)$

Тут $Children(v)$ — множество детей вершины v в подвешенном дереве G. А размер максимального паросочетания в поддереве с корнем в v тогда находится так: $\max(A[v], B[v])$ (определения A и B дополняют друг друга).

Ясно, что если v — лист, то $A[v] = B[v] = 0$. Если же v — не лист, то посчитаем $B[v]$ и $A[v]$ на основе его детей:

$$B[v] = \sum_{u \in Children(v)} \max(A[u], B[u])$$

Т.к. из определению $B[v]$ максимальное паросочетание, не содержащее рёбер (v, u), где $u \in Children(v)$, будет просто складываться из паросочетаний в поддеревьях с корнями в детях v.

С $A[v]$ немного сложнее: будем добавлять к паросочетанию ребро (v, u), где $u \in Children(v)$, так же, т.к. после этого в паросочетании уже есть ребро, инцидентное u, то к весу паросочетания нужно добавить $B[u]$ и ещё добавить сумму размеров максимальных паросочетаний в поддеревьях с корнями в $Children(v) \setminus \{u\}$. И это всё нужно помаксимизировать, используя u как параметр:

$$A[v] = \max_{u \in Children(v)} \Big(w(v,u) + B[u] + \sum_{x \in Children(v) \setminus \{u\}} \max(A[x], B[x]) \Big)$$

Тут $w(v,u)$ — вес ребра (v,u). Т.к. $B[v] = \sum_{u \in Children(v)} \max(A[u], B[u])$, то

$$\sum_{x \in Children(v) \setminus \{u\}} \max(A[x], B[x]) = B[v] - \max(A[u]B[u])$$

А значит:

$$A[v] = B[v] + \max_{u \in Children(v)} \Big(w(v,u) + B[u] - \max(A[u]B[u]) \Big)$$

Таким образом мы считаем $A[v]$ и $B[v]$ для всех вершин дерева G, причём для того, чтобы посчитать $B[v]$ мы тратим $\mathcal{O}(|Children(v)|)$ операций, как и для подсчёта $A[v]$, что видно из приведённых выше формул. А значит для того, чтобы посчитать $A[root]$ и $B[root]$ нам понадобиться $\mathcal{O}(E)$ операций, а т.к. в дереве на n вершинах $n-1$ ребро, то мы сможем сделать это за $\mathcal{O}(n)$. В качестве ответа выдаём $\max(A[root], B[root])$. ■

(b) G – цикл

Выберем в цикле вершину x. Ей инцидентны ровно 2 ребра: $e_1 = (x,u)$ и $e_2 = (v,x)$. Ясно, что если максимальному по весу паросочетанию M ребро e_1 принадлежит, то e_2 точно не принадлежит и наоборот. Рассмотрим граф $G \setminus \{e_1\}$ — это дерево. Тогда найдём в нём за $\mathcal{O}(n)$ паросочетание максимальное по весу. Таким образом мы получили некоторое паросочетание M_1. Аналогичным образом, но удаляя из графа не e_1, а e_2 получим некоторое паросочетание M_2 (всё за $\mathcal{O}(n)$, т.к. паросочетание ищутся в графе без цикла). Тогда ответом на задачу будет: $\max(W(M_1), W(M_2))$. Действительно, пусть $W(M_1) > W(M_2)$. Предположим, что существует паросочетание $M: \; W(M) > W(M_1)$. Невозможно, чтобы M не содержало e_1, т.к. иначе оно было бы максимальным паросочетанием в графе $G \setminus \{e_1\}$, но тогда оно было бы равно M_1, значит $e_1 \in M$, а $e_2 \notin M$, т.е. M – максимальное паросочетание в графе $G \setminus \{e_2\}$, т.е. $M = M_2$, но $W(M_1) > W(M_2)$. Пришли к противоречию, а значит паросочетание, построенное приведённой выше процедурой максимально. И построено оно за $\mathcal{O}(n)$, т.к. мы всё что мы сделали — это поискали дважды максимальное паросочетание в дереве на n вершинах. ■

(c) G – связный граф на n вершинах и n рёбрах

Заметим, что граф G отличается от дерева одним ребром и в графе G есть один единственный простой цикл. Рассмотрим обход в глубину графа G начиная с какой-либо вершинки, которую обозначим как $root$. Мы получим дерево, причём при обходе в глубино мы пройдём по $n-1$ прямому ребру и один единственный раз встретим обратное ребро, которое и образует цикл в графе G. Пусть это ребро e и в графе этому ребру смежны лишь 2 других ребра e_1 и e_2, которые принадлежат обходу в глубину. Заметим, что максимальному паросочетанию по весу в графе G могут одновременно лишь принадлежать либо e_1 и e_2 (и то, если они не смежны), либо e. Удаление любого из этих рёбер из графа делает граф деревом. Тогда аналогичными пункту (b) рассуждениями легко понять, что ответом на вопрос о максимальном паросочетании будет: $\max(maxMatch(G\setminus\{e\}), maxMatch(G\setminus\{e_1\}), maxMatch(G\setminus\{e_2\}))$. Т.е. выбирается максимальное по весу среди трёх максимальных паросочетаний в 3-х различных деревьях, что асимптотически работает за $\mathcal{O}(n)$ ■

Задача №2 (Про кактус)

Должно быть, это как-то аккуратно сводится к поиску паросочетания в каких-то деревьях...=)

Задача №3 (Число перестановок на n элементах без неподвижных точек за $\mathcal{O}(n^2)$)

Будем находить последовательно число перестановок без неподвижных точек для множества из $1, 2, \ldots, n$ элементов. Хранить ответы будет в массиве $A[1..n]$, т.к. ответ на вопрос задачи будет лежать в $A[n]$.
Число перестановок с k неподвижными точками равно: $\binom{n}{k}A[n-k]$, т.е. мы закрепляем k неподвижных точек, а оставшаяся часть перестановки не должна содержать таковых. Тогда число перестановок с как минимум 1 неподвижной точкой равно: $\sum_{k=1}^{n}\binom{n}{k}A[n-k]$. А тогда число перестановок на n элементах без неподвижных точек равно:

$$A[n] = n! - \sum_{k=1}^{n}\binom{n}{k}A[n-k]$$

Тут нужно положить, что $A[0] = 1$. Запишем для нескольких n:

$$\begin{aligned}
&A[1] = 1! - \binom{1}{1}A[0]\\
&A[2] = 2! - \binom{2}{1}A[1] - \binom{2}{2}A[0]\\
&A[3] = 3! - \binom{3}{1}A[2] - \binom{3}{2}A[1] - \binom{3}{3}A[2]\\
&\quad\ldots
\end{aligned}$$

Заметим, что биномиальные коэффициенты можно пересчитывать, например, по треугольнику паскаля: $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ и тратить на каждый следующий биномиальный коэффициент $\mathcal{O}(1)$ операций. Аналогично можно пересчитывать факториал $n!$ запоминая значение с предыдущего шага (т.е. подсчёта предыдущего $A[i]$) или же просто предподсчитав их за $\mathcal{O}(n)$ и сохранив в массиве. Видим таким образом, что на подсчёт $A[k]$ нам требуется $\mathcal{O}(k)$ операций, а значит, что т.к. для подсчёта ответа — $A[n]$ нужно посчитать все $A[k]$, $k \in \{1, \ldots, n-1\}$ то суммарно на получение ответа уйдёт $\mathcal{O}(n^2)$. ■

Задача №4 (Вероятность выпадения k орлов за $\mathcal{O}(nk)$)

Даны n монет с вероятностью выпадения орла $p_i,\ i \in \{1, \ldots, n\}$ каждая. Все монетки подкидываются и что-то выпадает...нужно найти вероятность выпадения k орлов за $\mathcal{O}(nk)$.
Разложим все монетки подряд и будем считать следующую величину:

$P[n, k]$ — вероятность выпадения k орлов используя первых n монет

Ясно, что:

$$P[n, k] = P[n-1, k](1 - p_n) + P[n-1, k-1]p_n$$

Т.е. либо среди первых $n-1$ монеты уже выпало ровно k орлов, тогда нам нужно домножить вероятность этого на вероятность того, что n-ая монета даст нам решку, либо среди первых $n-1$ монеты выпал лишь $k-1$ орёл, тогда нам нужно, чтобы n-ая монета упала орлом. Других вариантов нет.
Итого, нам нужно заполнить таблицу размера $(n+1) \times (k+1)$, причём:

$$P[n, k] = 0,\ k > n$$
$$P[n, 0] = \prod_{i=1}^{n} (1 - p_i)$$

И считаем идя слева направо, сверху вниз. Ответ будет лежать в правом нижнем углу: $P[n, k]$.
Таким образом нам нужно посчитать nk элементов таблицы, а это как раз делается за $\mathcal{O}(nk)$, т.к. каждый элемент уходит $\mathcal{O}(1)$ времени. ■

Задача №5 (Про группоид)

Дан группоид — множество M из g элементов с заданной на нём бинарной операцией «произведения» и замкнутый относительно неё. Дано произведение из n элементов M:

$$a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_n$$

Нужно, расставляя скобки, определить за $\mathcal{O}(n^3 g^2)$, какие различные элементы можно получить считая это произведение. Будем поступать почти так же, как в задаче на нахождение порядка произведения матриц произведение матриц:

$D[i, j]$	$=$	$Bool[1..g]$ — массив, что, если $D[i, j][g_k] = True$, то $g_k \in M$ можно получить как результат произведения $a_i \cdot a_{i+1} \cdot \ldots \cdot a_j$, а если $D[i, j][g_k] = False$, то нельзя

Считать $D[i, j]$ будем по мере увеличения $(j - i)$. Все массивы $D[i, j]$ заполнены $False$ изначально, только $D[i, i]$ такие, что $D[i, i][a_i] = True$.
Ясно, что зная все возможные различные результаты произведения $a_i \cdot \ldots \cdot a_k$ и все возможные результаты произведения $a_{k+1} \cdot \ldots \cdot a_j$ для всех k между i и j мы получаем, при помощи процедуры, приведённой ниже, все возможные результаты произведения $a_i \cdot \ldots \cdot a_j$.

```
for k from i to j − 1 do
    for x ∈ M do
        for y ∈ M do
            if D[i,k][x] = True AND D[k+1,j][y] = True then
                D[i,j][x · y] = True
```

Заметим, что данная процедура занимает $\mathcal{O}\left(ng^2\right)$, т.к. внешний цикл работает за $\mathcal{O}\left(n\right)$ и тело цикла перебирает всевозможные произведения за $\mathcal{O}\left(g^2\right)$.

Так нам нужно посчитать все $D[i,j]$ выше главной диагонали, т.к. ответ будет находится в ячейке $D[1,n]$, а подсчёт нужно производить по диагоналям, начиная с главной диагонали и поднимаясь выше и выше...Таким образом нам нужно найти $D[i,j]$ в $\frac{n^2}{2}$ ячейках, а значит суммарная сложность алгоритма $= \mathcal{O}\left(n^3g^2\right)$. ∎

PS: замечу, что все выделения памяти под D, очевидно, влезают во временные рамки. А в силу конечности группоида мы можем это производить индексацию по его элементам за константное время.