Практика по алгоритмам #11: DFS, динамика, реккурентность

Содержание

1. Разбор старых ДЗ

1.1. Динамика по подмножествам (практика #9)

1. *Покрасить граф в четыре цвета за $\mathcal{O}^*(2^n)$.* Перебираем A за 2^n и за $\mathcal{O}(m)$ проверяем, можно ли множества A и $\overline{A}$ покрасить в два цвета. $\mathcal{O}(2^n m)$.
2. *Унести вещи максимальной суммарной стоимости за 4 подхода.* Пусть у нас всего 2^k подходов. Решение за $\mathcal{O}(3^n k)$. $f_0[A]$ – можно ли унести A за один подход. $f_k[A]$ – можно ли унести A за k подходов, $f_k[A] = \bigvee\limits_{B \subseteq A} (f_{k-1}[B] \wedge f_{k-1}[A \setminus B])$. f_0 можно посчитать за $\mathcal{O}(2^n)$, переход от $k-1$ к k можно сделать за $\mathcal{O}(3^n)$. Используемая память: $\mathcal{O}(2^n)$.
3. *Количество способов разбить вершины неорграфа на циклы.* $f[A]$ – количество способов разбить множество вершин A на циклы, $g[A,x,y]$ – количество способов разбить множество вершин A на циклы и путь из x в y. Динамика вперед: $g[A|2^t,t,t]$+=$f[A]$, где t – первая вершина не из A, она обязательно покрыта каким-то циклом, будем его строить. $g[A|2^z,x,z]$+=$g[A,x,y]$, если $z \notin A$ и z смежна с y. $f[A]$+=$\frac{1}{2}(g[A,x,y] - f[A \hat{} 2^x \hat{} 2^y])$, если из y есть ребро в x. Вычитаем, чтобы не посчитать цикл из одного ребра. Время работы решения $\mathcal{O}(2^n nm)$.
4. *Количество совершенных паросочетаний в двудольном графе. В первой доли $m \le 15$, во второй доли и $n \le 1000$ вершин.* В отличии от поиска совершенного паросочетания, задача подсчета количества совершенных паросочетаний P#-трудна, надежды решить за полином нет. $f[i,A]$ – количество способов покрыть подмножество вершин A первой доли, используя первые i вершин второй доли. $f[i,A] = f[i-1,A] + \sum_x f[i-1, A \hat{} 2^x]$, где x – вершина из A, смежная с i-й вершиной второй доли. Время работы $2^m nm$ (nm – оценка сверху на число ребер в графе). Чтобы на практике решение работало быстро, важно хранить только последние две строки массива f: $f[i], f[i-1]$. Разница по памяти: $2^m n$ памяти – 32 mb, никуда не кешируется. 2^m памяти – 32 kb, отлично кешируется в кеш любого уровня. В задаче предполагалось, что или ответ помещается в 64-битный целый тип, или вычисления прооисходят по модулю.
5. *Составить расписание в школе.* Для каждого множества уроков $A \subseteq Q$ посчитаем $is[A]$ – можно ли все эти уроки провести одновременно. Для этого все преподаватели и учебные группы должны быть различны. Чтобы можно было распределить всех по классам, нужно проверить, что в двудольном графе (первая доля – уроки из A, вторая доля – классы, а ребро (a,b) есть, если можно провести урок a в классе b) есть паросочетание, покрывающее все уроки. Поскольку мы умеем находить паросочетание в двудольном графе за полиномиальное время, $is[A]$ можно посчитать для всех A за $\mathcal{O}^*(2^{|Q|})$. Теперь посчитаем целевую динамику $f[A]$ – минимальная длительность учебного дня, чтобы провести все уроки из множества A. $f[\emptyset] = 0$, $f[A] = 1 + \min\limits_{B \subseteq A \,:\, is[B]} f[A \hat{} B]$, что можно посчитать за $\mathcal{O}(3^{|Q|})$.
6. *Махинации.* Состояние – разбиение суммарного числа товаров A на n слагаемых, таких разбиений $\binom{n-1}{|A|+n-1} \le 2^{|A|+n-1}$. Разбиение на слагаемые $a_1, a_2, \ldots a_n$ будем кодировать так: $\underbrace{11\ldots1}_{a_1} 0 \underbrace{11\ldots1}_{a_2} 0 \ldots 0 \underbrace{11\ldots1}_{a_n}$, то есть битовой строкой длины $|A| + n - 1$.

   Как сделать переход (применить махинацию) за $\mathcal{O}(1)$? Будем рекурсивно перебирать старое и новое состояние: `go(i, A, B)`, где i – сколько типов товаров уже рассмотрели. A – битовый префикс старого состояния, B – битовый префикс нового состояния. Спуск в рекурсию работает за $\mathcal{O}(1)$, когда $i = n$, делаем переход из $[t, A]$ в $[t+1, B]$.

Возможность перехода отслеживаем при спуске. Итого фиксированную махинацию t мы попробуем применить ко всем состояниям A за $\mathcal{O}(\binom{n-1}{|A|+n-1})$ и итоговое время работы: $\mathcal{O}(\binom{n-1}{|A|+n-1}T)$.

д2. *Удаление минимального числа символов алфавита так, чтобы на каждом отрезке длины k данной строки s хотя бы один символ был удален.* Сперва за $\mathcal{O}(n)$ пройдемся по строке и для каждого отрезке $[i, i+k)$ посчитаем множество символов A_i, встречающихся на этом отрезке. Для этого идем двумя указателями, поддерживаем текущее A_i и, чтобы пересчитывать его, для каждого символа c поддерживаем $cnt[c]$ – сколько раз c встречается на отрезке $[i, i+k)$. Здесь мы предполагаем, что k не больше длины машинного слова w, то есть A_i можно закодировать 64-битным целым числом. Нам нужно теперь удалить такое множество символов B, что пересечение с каждым из A_i не пусто и $|B|$ минимален. Дальнейшая часть решения пока не разобрана.

1.2. Поиск в глубину (практика #10)

1. *Цикл, максимальный вес ребра минимален.* Сортировка весов ребер, бинарный поиск по этому массиву, dfs для проверки наличия цикла. Альтернативное решение а $\mathcal{O}(\texttt{sort}(E) + (V+E)^*)$: сортируем ребра и в порядке возрастания веса делаем `join`-ы в СНМ, пока не получим компоненту связности, не являющуюся деревом. Звездочкой обозначена обратная функция Аккермана.
2. *Оранжево-зеленые двери.* Для компонент связности решаем независимо. Дополним компоненту до Эйлеровой. Найдем Эйлеров цикл. Ориентируем ребра в направлении цикла. Сейчас у каждой вершины входящих (оранжевых) и исходящих (зеленых) ребер одинаковое количество. Удалим лишние ребра, баланс у каждой вершины изменился максимум на один.
3. *Найти все вершины, обязательные при путешествии из a в b.* Версия про ребра проще – нашли все мосты, взяли произвольный путь и все мосты на этом пути. Теперь два решения версии про точки сочленения. Оба решения начинаются со слов "рассмотри произвольный простой путь из a в b". Решение `#1`: возьмем только те точки сочленения, при прохождении по которым меняется принадлежность ребра компоненте реберной двусвязности. Решение `#2`: построим дерево из вершин двух типов – точки сочленения и компоненты ребрной двусвязности. Путь будем искать на этом дереве. Возьмем ровно те точки сочленения, по которым пройдет данный путь.
4. *Дополнить граф до Эйлерова, добавив ребра минимального сумммарного веса.* Соединяем циклом нечетные вершины. Если в компоненте связности хотя бы две, и в какой-то компоненте нет нечетных вершин, возьмем вершину минимального веса и добавим ей свободную степень два.
5. *Разбить двудольный 2^k-регулярный граф на паросочетания.* В каждой компоненте связности есть Эйлеров цикл и, поскольку граф двудолен, цикл имеет четную длину. Возьмем отдельно четные ребра, это 2^{k-1}-регулярный граф. И нечетные ребра, еще один 2^{k-1}-регулярный граф. Так делаем с каждой компонентой. Время итерации $\mathcal{O}(E)$. Каждую 2^k-компоненту заменили на две 2^{k-1}-компоненты. Всего нужно сделать k итераций, время работы $\mathcal{O}(kE)$.
6. *У каждой вершины должен быть враг в другой доли.* Строим любое корневое остовное дерево. Цвет вершины – четность её глубины.

д4. *Цветные провода.* Сведение к `2-SAT`: $x_i = 0$, если i-й провод воткнут в первый подходящий разъём, и $x_i = 1$, если во второй. Противоречие будет, если $\text{color}[i] = \text{color}[j]$, и позиции $x_i = e, x_j = f$ соседние на круге. Запрещаем противоречие, пишем $\neg(x_i = e \wedge x_j = f) \Leftrightarrow (x_i = \overline{e} \vee x_j = \overline{f})$. Получили конъюнкуцию 2-дизюъюнктов, то есть `2-SAT` задачу. `2-SAT` на n переменных и m дизюъюнктах мы умеем решать за $\mathcal{O}(n+m)$.

2. Новые задачи

1. Дополнение связного графа до сильносвязного (старая задача)
   a) Перейдем к конденсации
   b) Заметим стоки и истоки, построим новый граф
   c) Угадаем ответ (количество стоков и истоков)
   d) Решение `#1`: попытаемся провести ребро, которое на 1 уменьшает число истоков и стоков.
   e) Решение `#2`: попробуем замкнуть в цикл любое максимальное по включению паросочетание.
   f) Решение `#3`: $\mathcal{O}(V+E)$ возьмем любой максимальный по включению непересекающийся по вершинам набор путей из истоков в стоки.
2. Задача про командный пункт: выбрать k из n данных точек на окружности, максимизировать площадь.
   a) Решение за $\mathcal{O}(n^3 k)$ (очевидное)
   b) Решение за $\mathcal{O}(n^3 \log k)$ (очевидное)
   c) Решение за $\mathcal{O}(n^3)$ (нужно доказать корректность)
   d) Решение за $\mathcal{O}(n^2 \log k)$ (нужно доказать корректность)
3. Динамика
   a) Разбить текст на строки ширины не более w, слова переносить нельзя.
   b) Выбрать подотрезок слов текста, который поместится на экран h на w без переносов, и суммарная длина слов в котором максимальна.
   c) Сколько существует таблиц Юнга (диаграмма Юнга, в ней расставлены числа от 1 до n, строки и столбцы возрастают)? Есть два решения. Динамикой и формула.
4. Реккурентные соотношения
   a) $T(n) = T(n-1)+T(n-5)$, $S(n) = S(n-2)+S(n-3)$, докажите, что $T(n) = \Theta(S(n))$
   b) $T(n,k) = \mathcal{O}(n) + 2T(n, \frac{k}{2})$
   c) $T(n,k) = \mathcal{O}(n) + T(n, \frac{k}{2})$
   d) $T(n) = 2T(\frac{n}{2}) + \log n$
   e) $T(n) = \max\limits_{0<x<n} [T(x) + T(n-x) + x(n-x)]$

3. Разобранное в классе

2. Простое решение: перебрали точку разреза, запустили динамику, считающую за $\mathcal{O}(n^2k)$ функцию $area[n,k] \to \max$. Состояние: сколько точек взяли, какая последняя. Переход: перебрать следующую точку. Общее время работы решения – $\mathcal{O}(n^3k)$. Вероятностное решение за $\mathcal{O}(n^3)$: перебираем не все n начальных точек, а $\frac{n}{k}$ случайных, с вероятностью e^{-1} мы попадем хотя бы в одну из k точек ответа. Улучшаем до $\mathcal{O}(\frac{n^3}{k})$: внутреннюю динамику можно посчитать за $\mathcal{O}(n^2)$, используя метод "двух указателей". $p[n,k]$ – позиция предыдущей точки, если последняя n, а всего выбрано k точек. $p[n,k-1] \le p[n,k] \le p[n+1,k]$. В таких пределах и будем перебирать $p[n,k]$ (считаем динамику $area[n,k]$ назад). Другой подход: в исходной задаче видим динамику на подотрезках. $area[l,r,k]$ – выбрать на отрезке круга $[l,r]$ такие k точек, что площадь максимальна. При переходе будем делить k на $\lceil \frac{k}{2} \rceil$ и $\lfloor \frac{k}{2} \rfloor$. Состояний всего $\mathcal{O}(n^2 \log k)$. При пересчете динамики мы перебираем точку $m[l,r,k]$. Используем монотонность m: $m[l,r-1] \le m[l,r] \le m[l+1,r]$, получаем, что всю динамику можно посчитать за $\mathcal{O}(n^2 \log k)$ времени и, если не нужно восстанавливать ответ, $\mathcal{O}(n^2)$ памяти.
3. a) Жадность. Берем в первую строку максимум того, что можем.
   b) Разбили текст на слова, длина i-го слова равна l_i. Двумя указателями за линейное время для каждого i посчитали такое максимальное f_i, что $\sum\limits_{j \in [i,f_i)} l_j \le w$. Насчитали на ссылках $i \to f_i$ двоичные подъемы. От каждого i сделали за $\mathcal{O}(\log h)$ прыжок на h строк вперед. Время работы решения – $\mathcal{O}(|t| \log h)$. Используемую память можно уменьшить да $\mathcal{O}(|t|)$.
   c) Динамика. Выписываем числа в порядке возрастания. Очередное число мы можем дописать лишь в конец одной из строк. При этом длины строк должны убывать. Состояние – разбиение числа n на слагаемые.
4. a) $T(n) = T(n-1) + T(n-5) = [T(n-2) + T(n-6)] + T(n-5)$
   $S(n) = S(n-2) + S(n-3) = S(n-2) + [S(n-5) + S(n-6)]$
   b) $T(n,k) = \mathcal{O}(nk)$ (можно обобщить задачу до $T(n,k) = T(n,x) + T(n,k-x)$).
   c) $T(n,k) = \mathcal{O}(n \log k)$
   d) Например, эта реккурентность получается, когда мы пытаемся построить декартово дерево: `Build(n) = Build(n/2) + Build(n/2) + Merge(n)`. Для простоты предположим, что $n = 2^k$. $T(n) = \log n + 2\log\frac{n}{2} + 4\log\frac{n}{4} + \dots + 2^k \log\frac{n}{2^k}$. Заметим, что $2^i \log\frac{n}{2^i} = 2^i(k-i)$. Поэтому $T(n) = \sum\limits_{i=k..0} \frac{n}{2^i}(k-(k-i)) = \mathcal{O}(n)$.

4. Домашнее задание

4.1. Обязательная часть

1. (2) $T(n) = T(n-1)+T(n-6)$, $S(n) = S(n-3)+S(n-3)$, докажите, что $S(n) = \mathcal{O}(T(n))$.
2. (2) $T(n,k) = T(n-1,k) + T(n,k-1) + \Theta(n+k)$, докажите, что $T(n,k) = \mathcal{O}(2^{n+k}\min(n,k))$.
3. (2) $T(n,k) = T(2^n, k-1), T(n,0) = n$. Напишите явную формулу.
4. (2) $T(n) = \max_{x=1..n-1}[T(n-x) + T(x) + 1], T(1) = 1$. Напишите явную формулу.
5. (2) Найти в произвольном взвешенном графе паросочетание, вес которого отличается от паросочетания максимального веса не более чем в два раза. За полином.
6. (2) Дан неор граф, найти в нем ромб $((1,2);(1,3);(4,2);(4,3);(2,3))$ за $\mathcal{O}(VE)$.
7. (3) Дано дерево. Выбрать на нем k вершин так, чтобы максимальное расстояние от любой вершины до ближайшей из выбранных вершин было минимально. $\mathcal{O}(n\log n)$.
8. (3) По кругу стоят n точек. Выбрать k из них так, чтобы минимальное расстояние между соседними было максимально. $\mathcal{O}(n \cdot poly(log))$.

4.2. Дополнительная часть

1. (3) Дано дерево. Выбрать на нем k вершин так, чтобы сумма расстояний до ближайшей из выбранных вершин была минимальна.
2. (3) Задача с лекции про ориентацию графа. Докажите оценку времени работы $\mathcal{O}(E^2)$. При желании можно модифицировать решение.
3. (4) Дан неор граф, найти в нем ромб $((1,2);(1,3);(4,2);(4,3);(2,3))$ за $\mathcal{O}(E\sqrt{E})$.
4. (4) $T(n,k) = \max\limits_{0<x<n}[T(x,k) + T(n-x,k) + \min(x,k)\min(n-x,k)]$.