Семинар к лекции 12.

#вшпи #аисд #дп

Автор конспекта: Гридчин Михаил

Задача 1

В дощечке в один ряд вбиты гвоздики (они отсортированы по координате). Любые два гвоздика можно соединить ниточкой. Требуется соединить некоторые пары гвоздиков ниточками так, чтобы к каждому гвоздику была привязана хотя бы одна ниточка, а суммарная длина всех ниточек была минимальна.

Решение: *TODO*. Это изи, и вроде даже гуглится по первой ссылке.

Задача 2

Дана матрица $n \times m$ целых чисел. Нужно пройти из левого верхнего в правый нижний угол,
ходить можно только вправо или вниз. Стоимость пути — это произведение всех чисел на пути. Найти минимальное количество нулей, на которое может заканчиваться стоимость пути.
Асимптотика $O(nm)$.

Решение:

Пусть $\mathrm{dp}[i][j]$ - количество двоек и пятёрок в разложении на простые множители самого оптимального пути в подматрице $a_{11} \dots a_{ij}$. При этом если посчитать минимальное количество двоек на пути во всей матрице и минимальное количество пятёрок на пути во всей матрице (возможно, другом), то ответом будет минимальное из этих чисел

$$\mathrm{dp}[i][j] = \begin{pmatrix} \min\{\mathrm{dp}[i-1][j][0], \mathrm{dp}[i][j-1][0]\} + \text{ количество 2 в разложении a[i][j]} \\ \min\{\mathrm{dp}[i-1][j][1], \mathrm{dp}[i][j-1][1]\} + \text{ количество 5 в разложении a[i][j]} \end{pmatrix}$$

Ответ - $\min\{\mathrm{dp}[n][m][0], \mathrm{dp}[n][m][1]\}$.

Задача 3

Рассмотрим задачу о кузнечике в следующей постановке. Кузнечик может прыгать на $1, 2, \dots, k$ клеток вперед. Надо добраться от $a_0 = 0$ до a_n, минимизировав сумму, где a_i — стоимость попадания на i-ю клетку. Время: $O(n)$, память: $O(k)$.

Решение:

Наивно:

$$\mathrm{dp}[i] = \min_{j=1}^{k}(\mathrm{a}[i-j]) + \mathrm{a}[i]$$

Асимптотика $O(nk)$.
Будем поддерживать очередь на минимум, чтобы за $O(1)$ добавлять очередной элемент в очередь ($\mathrm{dp}[i]$) и извлекать минимум за $O(1)$. Асимптотика $O(n)$ по времени и $O(k)$ по памяти. Это и требовалось построить.

Задача 4

На плоскости есть множество из N кругов. Любые два либо вложены, либо не пересекаются. Найдите наибольшую цепочку вложенных кругов за $O(N^2)$.

Решение:

Заметим, что круг с меньшим радиусом не может покрывать круг с большим радиусом. Круг A содержится в круге B тогда и только тогда, когда

$$\mathrm{dist}(center_A, center_B) + r_A \le r_B$$

Но поскольку все круги либо вложены, либо не пересекаются, то можно проверить проще:

$$|c_A - c_B| \le r_B - r_A \quad \&\& \quad r_A < r_B$$

Решение 1.
Отсортируем все круги по убыванию радиуса. Тогда $\mathrm{dp}[i]$ - это самая длинная цепочка вложенных кругов с концом в круге i. Тогда задача свелась почти к поиску наибольшей возрастающей подпоследовательности.

Решение 2.
Построим граф G на N вершинах, где вершины - это круги, а ориентированное ребро $(u, v), u, v \in V$ означает, что круг u вложен в круг v. Заметим, что полученный граф - лес, а ответ - длина максимального пути в этом лесе. Это легко сделать, например, запустив алгоритм dfs в каждой вершине. Асимптотика построения - $O(N^2)$, асимптотика решения - $O(N^2)$ наивно и $O(N)$ с сохранением уже посчитанных расстояний.

Упражнение. Покажите, что задачу *НЕВОЗМОЖНО* решить за $O(N \log N)$ в общем случае.
Решение. Отношение вложенности не является полным порядком - два круга могут быть несравнимы (каждый не внутри другого). Поэтому их нельзя просто отсортировать. В худшем случае придётся проверить и сравнить $\Omega(N^2)$ пар кругов.

Задача 5

Найдите наибольшую подпоследовательность-палиндром.

Решение:

Решение 1. Найдём НОП данной строки и развёрнутой строки.

Решение 2. Пусть $\mathrm{dp}[i][j]$ - длина наибольшей палиндромной подпоследовательности для подстроки $s[i..j]$:
База: $\mathrm{dp}[i][i] = 1$, $\mathrm{dp}[i][j] = 0, j < i$.
Переход:

$$\mathrm{dp}[i][j] = \begin{cases} 2 + \mathrm{dp}[i+1][j-1], & s[i] = s[j] \\ \max(\mathrm{dp}[i+1][j], \mathrm{dp}[i][j-1]), & s[i] \neq s[j] \end{cases}$$

Порядок вычисления: по увеличению $j - i$
Ответ: $\mathrm{dp}[0][n-1]$.

В обоих решениях память можно улучшить до $O(n)$, время работы обоих решений $O(n^2)$.

Упражнение. Докажите, что решить задачу за $O(n \log n)$ невозможно.

Задача 6

Решите задачу о рюкзаке в следующих постановках:

1. i-й предмет можно брать от 0 до cnt_i раз
2. каждый предмет можно брать неограниченное число раз

Решение:

Пусть для определённости w_i - вес предмета, c_i - стоимость, всего n предметов, $\sum w_j \leq W$, $\sum c_j \to \max$, где W - грузоподъёмность рюкзака. Классическую постановку задачи умеем решать за $O(nW)$ времени и памяти (с восстановлением ответа).

Задача 1.

Наивное решение: продублируем каждый предмет ровно cnt_i раз, получим решение за $O(W \sum \mathrm{cnt}_i)$.

Эффективное решение: Поскольку мы хотим, чтобы каждый предмет можно было взять $0, 1, \ldots, \mathrm{cnt}_i$ раз, то заметим, что если представить число cnt_i в битовой записи, то в ней будет $O(\log \mathrm{cnt}_i)$ битов. Давайте теперь представим эти биты, как веса этого самого предмета. Утверждается, что взяв каждый бит по 0 или по 1 разу, мы сможем набрать любое число от 0 до cnt_i. Действительно, это следует из двоичной записи числа. Теперь получаем, что каждый предмет дублируется уже не cnt_i раз, а $O(\log \mathrm{cnt}_i)$ раз. Таким образом, итоговая асимптотика равна $O(W \sum \log \mathrm{cnt}_i)$.

Note. Серьёзное замечание. Если cnt_i - не степень двойки, то наибольшую степень нужно брать, как cnt_i минус сумма всех меньших степеней, чтобы в какой-то момент не случился перебор и не пришлось задумываться об этом. Таким образом, мы уменьшили значение наибольшего слагаемого в разложении на степени двойки.

Утверждение (от Влада). Можно решить за $O(Wn)$. Автор не уверен, что это возможно.

Задача 2.

Жадное решение: отсортируем предметы по убыванию $\frac{c_i}{w_i}$ и будем жадно брать эти предметы, пока вмещаются в рюкзак.
Упражнение. Докажите, что жадное решение не работает.
Подсказка. $W = 12$.

c_i	10	8	4
w_i	9	8	4

Эффективное решение. Пусть $\mathrm{cnt}_i = \frac{W}{w_i}$. Тогда мы уже умеем решать эту задачу за $O(W \sum \log \frac{W}{w_i}) = O(W \log \frac{W^n}{w_1 \cdot \ldots \cdot w_n}) = O(Wn \log W)$. Если же мы умеем решать предыдущую задачу быстрее, то и здесь можем решить за $O(Wn)$.

Другое эффективное решение. Заметим, что если восстанавливать ответ необязательно, то написать одномерный рюкзак крайне просто. Существует два подхода сделать это:

1. Для каждого предмета i перебираем все веса *по возрастанию* и пытаемся добавить очередной предмет в рюкзак, стараясь улучшить ответ.
   $\mathrm{dp}[j] = \max(\mathrm{dp}[j], \mathrm{dp}[j - w[i]] + c[i])$.
2. Для каждого веса перебираем все предметы.

Оба подхода дают одинаковый результат (почему?))). Первый подход иногда проще для понимания.
Асимптотика решения $O(Wn)$.

Задача 7

Число является гладким, если модуль разности между любыми двумя соседними цифрами
лежит в отрезке $[l, r]$. Сколько существует различных гладких чисел длины n? Времени: $O(n)$.

Решение. Это классическая идея, называемая, как ДП по цифрам. Пусть $\mathrm{dp}[i][j]$ - количество гладких чисел длины i, заканчивающихся на цифру $j \in [0, \ldots, 9]$. Тогда пересчёт формулы будет таким:

$$\mathrm{dp}[i][j] = \sum_{k=0}^{9} \mathrm{dp}[i-1][k] \cdot (l \le |j-k| \le r)$$

Note (субъективный опыт). В таких задачах часто удобно писать ДП вперёд, а не ДП назад. Также бывает проще писать рекурсию с мемоизацией, чем итеративно. Также базу нужно вводить очень аккуратно (число не может начинаться на 0).

Задача 8

Дан массив целых чисел $a_1, \ldots, a_N$. Найти наибольшее подмножество S такое, что в любой паре $x, y \in A$ либо x делится на y, либо y делится на x.

Решение. Отсортируем все числа по убыванию. Теперь нужно найти наибольшую убывающую подпоследовательность, такую, что каждый элемент делится на все следующие в подпоследовательности. Тогда динамика ясна

$$\mathrm{dp}[i] = \max_{\substack{j<i \\ a_j | a_i}} \mathrm{dp}[j] + 1$$

Эту динамику легко посчитать за $O(N^2)$.

Задача 9

Дано множество кругов на плоскости, любые два либо вложены, либо не пересекаются. Най-
дите число подмножеств размера таких, что круги в нем можно представить как цепочку вложенных друг в друга кругов. Времени: $O(N^2)$, памяти $O(N)$.

Решение. Построим лес из 4 задачи, но для каждой вершины будем хранить всего одного предка (наименьший круг, покрывающий данный круг). Теперь запустим dfs от каждой вершины. Для каждой вершины будем хранить массив $\mathrm{dp\_v}$ размера , где $\mathrm{dp\_v}[i]$ - это количество цепочек для поддерева вершины v длины i. Тогда $\mathrm{dp\_v}[i]$ - это количество способов выбрать i вершин в поддереве вершины v, при условии, что v выбрана. Заметим, что чтобы добиться $O(N)$ дополнительной памяти, будем очищать массив $\mathrm{dp\_v}$ для тех вершин, которые нам больше не понадобятся для пересчёта динамики и ответа. Тогда сложность решения по времени будет $O(N^2 + N) = O(N^2)$.