Домашняя работа №6

Шибаев Александр Б05-222

Март 2023

1 Первая задача

Очевидно, что соединять два несоседних гвоздика никогда нет смысла. Отсортируем гвоздики и сделаем $dp[i][0]$ и $dp[i][1]$, где $dp[i][0]$ – минимальная суммарная длина верёвок для первых i гвоздиков, если i-ый гвоздик ни с кем не соединён, $dp[i][1]$ – аналогично, но i-ый соединён с предыдущим. Тогда пересчёт будет выглядеть так: $dp[i][0]$ можно обновить только из $dp[i$ - $1][1]$, или $i = 0$. $dp[i][1]$ можно обновить и из
$dp[i$ - $1][0]$, и $dp[i$ - $1][1]$, выбрав минимум.

2 Вторая задача

Пусть dp[i][j] - минимальное количество перевозок, которое необходимо сделать кораблю, чтобы перевести все объекты с номерами a_1 , ..., a_i и a_j , ..., a_n. Вместе с этим будем также хранить, сколько свободного места ($rest[i][j]$) останется на корабле при последней перевозке, если мы возьмем столько объектов. Тогда очевидно, что либо объект с номером i, либо объект с номером j мы берем последним
т.е. $dp[i][j] = min(dp[i-1][j] + delta1$, dp[i][j 1] $+ delta2)$, где соответствующие $delta$ равны 0, если можно уместить текущий объект (т. е. $rest[i][j] < a[i/j]$), или 1, если нельзя. Остаток пересчитывается также тривиально.

3 Третья задача

Обычный рюкзак работает за $O(Wn)$, давайте его оптимизируем. Обычно, когда мы хотим уменьшить асимптотику в n раз, используется очень простая идея - давайте разделим n на n блоков и посчитаем ответ для каждого блока, потом за n сольем из них ответ для всего множества. Не трудно понять, что здесь тоже работает такое рассуждение. Разделим все n предметов на n блоков и для каждого блока посчитаем ответ за $O(Wn)$ времени, используя $O(Wn)$ памяти (будем использовать одну и ту же таблицу, при этом для каждого блока сохранять только последнюю строчку). Тогда мы получим ответ для каждого блока за необходимую асимптотику. Теперь восставновим ответ - нужно просто пройтись с конца и, зная ответ, восстановить его в каждом блоке.
Асимптотика $O(Wn)$ памяти и $O(2Wn)$ времени.

4 Пятая задача

Написанный код - это обычное SOS (Sum Over Subset) dp. Давайте разберем его подробнее. Очевидно, что сумму по всем подмаскам длины n можно решить за $O((2^n)^2)$ просто перебором по всем подмаскам. Однако с помощью дп можно ускорить время работы данного алгоритма до $O((2^n) * \log 2^n) = O(2^n * n)$. Как мы можем видеть из кода, предоставленная функция работает именно за такую асимптотику. Давайте разберем идеи, заложенные при написании этой функции. Обозначим $dp[k][mask]$ - сумма по всем подмаскам маски $mask$, причем каждая из подмасок должна гарантированно совпадать в первых k битах. База здесь очевидна: $dp[n][mask] = a[mask]$. Поймем переход: пусть у нас фиксированы первые i - 1 бит и какая-то маска. Будем пересчитывать состояния для маски, в которой отличается первый незафиксированный бит (действительно - это логично, поскольку у нас есть измерение динамики по этому параметру, поэтому мы можем изменять его только на 1). Если первый незафиксированный бит этой маски равен 0, то мы не можем получить новую подмаску, а если 1, то тогда подмаска получается тривиально:

$$submask = mask(1 << i)$$

Именно такой переход индукции мы можем увидеть в коде (разница только в том, что мы разобрали динамику назад, а в коде реализована динамика вперед).
Почему это работает? Потому что мы фактически делаем "перебор"с уже предподсчитанными значениями т.е. для каждой позиции мы суммируем все подмаски. Почему именно все? Потому что для данной маски и i зафискированных бит мы можем получить ограниченное количество переходов (а именно - два), которые мы рассмотрели.

5 Шестая задача

Очевидно, что решением задачи является модифицированный рюкзак. Вместо обычной таблицы будем хранить таблицу размера n * W/ω, причем в каждой ячейке будем хранить битовую маску из ω символов. Каждый n-ый бит в маске на i-ой строке в j-ом столбце будет равен 1, если с помощью первых i предметов можно набрать вес W/ω * $j + n$ и ноль иначе. Очевидно, что если мы могли набрать такой вес с i предметами, то можем и с $i + 1$ (просто не взяв последний), т.е. $dp[i][j] = dp[i-1][j]$. Также понятно, что мы можем взять вес, равный весу самого предмета т. е. соответствующий бит надо установить в единицу. При этом также понятно, что к уже существующим весам мы можем прибавить текущий вес т.е. также верно, что $dp[i][j] = dp[i][j]|(dp[i][j-1] >> (a[i]\%\omega))$. Как же восстановить ответ? Очень просто - на каждом шаге мы можем просто поддерживать значение, какой максимальный по номеру бит у нас является единичкой. Тогда сразу после прохождения таблицы мы можем получить ответ.
P.S. решение задачи может показаться довольно странным, но по сути это просто обычное ДП с таблицей, только таблицу мы уменьшили в ω раз, чтобы уложиться в асмиптотику.

6 Седьмая задача

Понятно, что любому мультимножеству, написанному на кубиках, соответствует мультимножество из 6 чисел. Более того, если у нас есть мультимножество из n чисел и 3 кубика, то методом сжатия координат мы можем уменьшить это число до 18 различных чисел т.е. для любого n мы можем построить биекцию и решать задачу на числах от 1 до 18. Поскольку мы поняли, что для любого n можно решить задачу на маленьких (до 19) множествах, то давайте просто предпосчитаем все заранее необходимые ответы. Можно это сделать втупую - просто перебрать 18^{18} комбинаций, для каждой из них посчитать вероятность выпадения большего числа и оценить ответ. Таким образом для каждого n получим асимптотику $O(18^{18} * 63) = O(1)$.

7 Восьмая задача

Пусть $dp[i][j][k]$ - ответ для строки длины i $(1 \leq i \leq l)$, причем суффиксом этой строки являются k символов из строки j. Почему надо рассматривать именно такую динамику? Это следует из примитивного наблюдения, что сама строка должна состоять из частей слов s_i , так что наполнять какими-то рандомными символами смысла нет, поэтому будем делить нашу строку на части. Тогда понятно, что можно пересчитать динамику из всех строк меньшей длины:

$$dp[i][j][k] = min(dp[i-1][j][k])$$

по всем j из $1..n$ и k из $1..|s_j|$. Тогда очевидно, что ответ мы получим за $O(S * l * n)$. Давайте улучшим время - очевидно, что каждый раз мы делаем однотипные переходы в динамике, поэтому здесь мы можем воспользоваться возведением матрицы размера $nS *$ nS в степень i. т.е. мы можем решать задачу за
$O((nS)^{2.8} * \log l)$. Теперь давайте заметим, что когда мы переходим от i-ой длины слова к $i+1$, у нас имеется всего лишь S состояний динамики (и они все также однотипные, поэтому возможность использования матриц сохраняется), поскольку оно равно $\sum\limits_{i=1}^{n}|a_i| =$ S. Тогда все, что нам остается - создать матрицу S на S и возвести её в степень l, что, очевидно, выполняется за необходимую асимптотику.