Тураев Тимур, 504 (SE)
07.12.2013

13.1. Посчитать число подпалиндромов за квадрат.

1-й способ.

Заведем массив L[0..n] (n – длина строки), где L[i] означает число подпалиндромов, центр которых лежит в позиции i в исходной строке. Причем для подпалиндромов нечетной длины, центр определяется однозначно, а для «четных» центр определим как левый из двух центральных букв.

Пересчет: для каждой позиции в строке i будем двигаться двумя указателями налево и направо, пока есть куда двигаться и пока слово между двумя указателями еще палиндром (проверяется легко – пока соответствующие буквы s[left] и s[right] равны) и будем увеличивать число L[i] на единицу.

Ответ на задачу: сумма чисел в массиве L

Время: очевидно, O(n^2), так как для каждого индекса в худшем случае сделаем проход по всей строке.

2-й способ (он пригодится для решения второй задачи)

Заведем двумерный массив L, где L[i][j] будет означать начало j-го палиндрома, который заканчивается в позиции i в строке.

Пересчет: такой же как в первом способе, только обновление будет таким: если слово между left и right палиндром, то L[right].push_back(left).

Ответ на задачу: сумма «длин» массивов L[i] или просто количество чисел в массиве L.

Время: такое же.

13.2. Кратчайшее разбиение слова на подпалиндромы.

1-ый шаг: сначала найдем все подпалиндроме в слове за квадрат (это 2-й способ задачи 1), то есть заполним двумерный массив L, где L[i] – начала всех подпалиндромов, заканчивающихся в позиции i.

2-ой шаг:

Состояние динамики: dp[i] – число «слов» в кратчайшем разбиении i-го префикса нашего слова. То есть «ответ» на подстроке s[0:i] (длина подстроки равна i).

Пересчет динамики:

$$dp[i] = \min_{j \in L[i]} \{dp[L[i][j]]\} + 1$$

Понятно почему это так: когда мы находимся в позиции i в строке, мы знаем, что по этому индексу заканчивается какой-то подпалиндром (он всегда есть, хотя бы однобуквенный). Давайте переберем их все и выберем наилучший ответ.

Ответ на задачу: очевидно, ответ лежит в dp[n].

Время: Число подпалиндромов, заканчивающихся в позиции i не больше i, значит второй шаг работает (как и первый) за O(n^2).

Замечание: если нужно знать не только число подпалиндромов в разбиении, но и сами палиндромы, то вместе с динамикой можно хранить то самое начало палиндрома, который заканчивается в позиции i, которое минимизировало значение динамики – получится аналог массива предков, по котором можно определить из каких палиндромов состоит слово.

Замечание_2: если теперь найти все циклические сдвиги исходного слова (из ровно |s|=n), и для каждого слова решить эту задачу, а потом найти среди них минимум (уже работать будет за n*O(n^2) = O(n^3)), то получится задача D из домашнего задания, которую я уже сдал.

13.3. Кратчайшее представление.

Динамика по подотрезку (подстроке)

Состояние динамики: заведем такую динамику dp[i][j], которая нам скажет, какая длина сжатой подстроки s[i:j]. Ясно, что ответ будет лежать в dp[0][n].

Пересчет: переберем все подстроки в строке, начиная с самой короткой (то есть направление пересчета динамики: прямое в порядке увеличения длины рассматриваемой подстроки). Пусть сейчас рассматривается подстрока s[l:r], то есть определяется значение динамики dp[l][r]. Тогда возможны 3 случая:

- Кратчайшее представление подстроки s[l:r] и есть она сама, тогда dp[l][r]=len(s[l:r]).
- Строку можно представить в виде двух сжатых строк (важное замечание: сжатых строк <u>меньшей</u> длины, а так как направление пересчета в порядке увеличения длины подстроки, мы все уже про них знаем). Давайте их все переберем и выберем наилучший:
$$dp[i][j] = \min_{k\in[l+1,r)}\{dp[l][k] + dp[k][r]\}$$
- Строка состоит из нескольких (больше одного) одинаковых и их можно сжать. Найдем среди них наилучший, такой, что длина строки «(R)X» будем минимальной.

Значение dp[l][r] будем минимум по всем трем случаям.

Ответ: как уже говорилось, ответ лежит в dp[0][n].

Опять же, если нужно знать разбиение, то можно вместе с динамикой хранить и само представление сжатой строки.

13.4. Триангуляция с минимальным весом.

Идея такая же, как у предыдущей задачи: динамика по подмножеству.

Состояние динамики: пусть dp[i][s] означает минимальную триангуляцию многоугольника, начинающегося с вершины i и имеющий ровно s вершин, причем вершины в под-многоугольнике берутся подряд, в порядке обхода по часовой стрелке начиная от вершины i. Причем dp[i][k]=0 при k<4.

Пересчет: выберем какую-либо вершину k среди вершин i+1, i+2, …, i+s-3, i+s-2 (то есть все, кроме первой и последней) и проведем 2 диагонали в эту вершину из первой и последней. Получится 2 подзадачи меньшего размера. Будем считать динамику как в прошлой задаче, в порядке увеличения размера задачи (то есть размера многоугольника), это нам позволит решить бОльшую задачу. Таким образом, формула пересчета:
$$dp[i][s] = \min_{1\le k\le s-2}\{dp[i][k+1] + dp[i+k][s-k] + w(i,k) + w(i+k,i+s-1)\}$$
где $w(i,k)$ – вес ребра из первой (i) вершины в выбранную (k), а вторая w – вес ребра из последней вершины в выбранную; первые два слагаемых – решения меньших подзадач.

Ответ: минимум на последнем столбце таблицы dp[i][n]

Опять же, чтобы получить не только вес триангуляции, но и ее саму, нужно хранить то самое k, которое минимизирует динамику и с помощью этого восстановить все нужные диагонали.