Субпалиндромы

Решение

Казалось бы, при чём здесь динамическое программирование? Вроде бы, здесь не надо искать ничего минимального/максимального/наилучшего/и т.д. Надо просто что-то подсчитать. И тем не менее, что-то общее есть. Ну, например, то, что перебирать все варианты построения субпалиндромов - дело заведомо безнадёжное, уж очень их может быть много ☺.

Но динамическое программирование подразумевает разбиение задачи на шаги. Как здесь шагать? Легко: будем добавлять по одной букве. И если мы, зная всё, что нам необходимо знать о строке длины N символов, сумеем вычислить всё необходимое для строки из (N+1) одного символа, то всё у нас получится!

Отвлечёмся немного от нашей задачи, и, для примера, рассмотрим совсем другую, совсем простую задачу. Вот такую: в траве сидел кузнечик с поломанной лапкой ☹; из-за травмы кузнечик может прыгать только вправо и только прыжками длиной либо 1, либо 3, либо 4 метра; на расстоянии M метров справа от кузнечика расположено его гнездо. Спрашивается, сколько имеется у кузнечика разных способов попасть в гнездо? Например, если M=6, то имеется всего 9 способов: 6 = 1+1+4 = 1+4+1 = 4+1+1 = = 3+3 = 3+1+1+1 = 1+3+1+1 = 1+1+3+1 = 1+1+1+3 = 1+1+1+1+1+1.

Ключ к решению в этой задаче именно такой, как описано выше. Обозначим f(N) количество способов кузнечика попасть в точку, отстоящую от его начальной позиции на N. Допустим, мы всё знаем о движениях кузнечика на рассстояние, не превосходящее N от начальной точки. А как найти f(N+1)?

Последний прыжок кузнечика перед попаданием в точку N+1 мог иметь длину 1, 3 или 4 метра. Понятно, что имеется f(N) таких способов попасть в точку N+1, которые заканчиваются прыжком в 1 метр, f(N-2) таких способов попасть в точку N+1, которые заканчиваются прыжком в 3 метра и f(N-3) таких способов попасть в точку N+1, которые заканчиваются прыжком в 4 метра, а других способов попасть в точку N+1 нет. Значит, f(N+1)=f(N)+f(N-2)+f(N-3) для всех N>3. Остаётся только вычислить f(1), f(2) и f(3) и затем пчследовательно вычислять все осталные значения f вплоть до f(M). Можно даже не вычислять f(1), f(2) и f(3), а просто положить (вполне естественно!) f(0)=1, f(-1)=f(-2)=0 и вычислять f(1), f(2), ..., f(M) по полученной формуле.

В решении задачи о кузнечике хорошо видны мотивы из динамического программирования: вычислив величину f(N) мы забываем все многочисленные варианты попадания в точку N, мы помним только необходимую нам величину f(N). И неважно, что это не наибольшая/наименьшая/наилучшая в каком-то смысле величина, а просто количество чего-то (в случае с кузнечиком – способов), важно то, что получив некоторую промежуточную величину, мы можем забыть, как именно она была получена.

Но вернёмся к задаче о субпалиндромах.

Допустим, мы знаем всё, что только захотим узнать про субпалиндромы строки из первых N символов. А теперь мы хотим узнать, сколько субпалиндромов содержит строка из первых (N+1) символов. Обозначим g(N) количество субпалиндромов строки из первых N символов. Понятно, что g(N+1) = g(N) + количество субпалиндромов, заканчивающихся на (N+1)-ом символе строки. В самом деле, g(N) – это количество субпалиндромов, заканчивающихся левее (N+1)-го символа, и надо добавить

субпалиндромы, заканчивающиеся на (N+1)-ом символе. А как узнать эту величину? Обозначим её h(N+1). Обозначим также символ, стоящий на (N+1) месте в строке через s(i). Чтобы вычислить h(N+1) будем двигаться вдоль строки слева направо в поисках символа, совпадающего с s(N+1). Допустим для определённости, что такой символ встретился на k-ом месте. Легко видеть, что количество субпалиндромов, начинающихся на k-ом символе строки и заканчивающихся на (N+1)-ом равно g(k+1,N), где g(k+1,N) – количество субпалиндромов, содержащихся в строке от (k+1)-го символа данной строки до N-го символа данной строки. И нам надо вычислить сумму g(k+1,N) для всех таких k<=N, что s(k) совпадает с s(N+1). Вот оно! Мы пришли к необходимости подсчитывать субпалиндромы не только всей строки от первого до N-го символа, а любой строки от L-го символа до R-го. Двигаясь в этом направлении, мы быстро поймём, что для подсчёта g(L,R) понадобятся знать величины $g(L_1,R_1)$ для некоторых $L< L_1<R_1<R$ и так далее, в общем, ситуация запутывается...

Ничего страшного – для распутывания таких ситуаций имеется стандартный приём, и состоит он в том, что мы вычисляем g(L,R) не в порядке возрастания R, а в порядке возрастания длины обрабатываемого отрезка, т.е. сначала вычисляем все g(L,R) для всех отрезков длины 1 (R=L), затем – все g(L,R) для отрезков длины 2 (R=L+1), затем – длины 3 (R=L+2) и т.д. вплоть до отрезка (единственного) длины Len, где Len – это длина всей данной строки. Кстати, g(1,Len) даст ответ на вопрос исходной задачи.

Приём этот срабатывает в подобных ситуациях очень часто. В учебных материалах кратко изложены решения нескольких задач, в которых так и происходит.

Но, всё-таки, сначала закончим с задачей о субпалиндромах.

Итак, нам необходимо вычислить g(L,R). Длина этого отрезка строки равна R-L+1, а мы уже знаем $g(L_1,R_1)$ для всех строк меньшей длины. Рассмотрим два случая.

1 случай. Символы s(L) и s(R) различаются. В этом случае в строке от L-го символа до R-го нет субпалиндромов, содержащих и s(L), и s(R). Количество субпалиндромов, не содержащих символа s(L), равно g(L+1,R); количество субпалиндромов, не содержащих символа s(R), равно g(L,R-1), всего получаем g(L+1,R)+g(L,R-1) субпалиндромов. Однако в этой сумме дважды подсчитаны все субпалиндромы, расположенные на отрезке от (L+1)-го до (R-1)-го символа. Уберём лишнее. Итого, в данном случае

$$g(L,R) = g(L,R-1) + g(L+1,R) - g(L+1,R-1).$$

2 случай. Символы s(L) и s(R) одинаковые. Имеется g(L+1,R-1) субпалиндромов, которые начинаются с s(L), заканчиваются в s(R) и содержат ещё какие-то символы между ними, ещё один субпалиндром, состоящий из двух символов - s(L) и s(R), а также все субпалиндромы, которые не содержат одновременно и s(L) и s(R) – количество таких субпалиндромов мы уже подсчитали в первом случае. Итого, в данном случае g(L,R) = g(L+1,R-1) + 1 + g(L,R-1) + g(L+1,R) - g(L+1,R-1), т.е.

$$g(L,R) = 1 + g(L,R-1) + g(L+1,R).$$

Остаётся только заметить, что строки длины 0 не содержат субпалиндромов, а каждая строка длины 1 содержит один субпалиндром. Этого достаточно – ведь для вычисления g(L,R) достаточно знать только значения $g(L_1,R_1)$ для всех строк, длины которых меньше на 1 или на 2.

Это же соображение позволяет нам заметить, что для последоватльного вычисления всех g(L,R) не нужно заводить двумерный массив – вполне достаточно трёх одномерных массивов длины Len. В приведённом Паскаль-тексте они названы r0, r1 и r2. Программа простая, прозрачная и комментариев не требует.