

Рассмотрим задачу о подсчёте субпалиндромов в очень общей формулировке, а именно: имеется некоторый массив данных; требуется вычислить некоторую характеристику этого массива (да уж, более общо сформулировать трудно ☺). Под массивом здесь подразумевается множество данных, каким-то образом расставленных, некоторым образом упорядоченных по месту их расположения. При решении этой задачи мы столкнулись с необходимостью вычислять некоторые характеристики всевозможных подмассивов массива данных. Скажем, для задачи о подсчёте субпалиндромов массив – это заданный ряд букв, а подмассивы – это отрезки этого ряда. В примерах, которые мы разберём ниже, появятся и другие варианты.

Для вычисления необходимых характеристик всех подмассивов мы применили такой **приём: рассматриваем все подмассивы в порядке увеличения их размеров**. Снова проецируя эту формулировку на задачу о подсчёте субпалиндромов, увидим, что размер подмассива в ней – это, естественно, длина подмассива. Приём этот срабатывает довольно часто и весьма эффективно. Примерам применения этого приёма и посвящены заметки ниже.

Пример 1. Начнём с «вечнозелёного» примера – умножение цепочки матриц. Разбор этой задачи встречается чуть ли не в любом тексте, посвящённом динамическому программированию. Не будем ломать традицию, хотя и взглянем на задачу со своих позиций. Что такое умножение матриц для этой задачи не существенно, поэтому я сформулирую задачу без её обычной «сказки». Итак...

Имеется массив чисел $a_1, a_2, a_3, \dots, a_{N-1}, a_N$. За один ход разрешается вычеркнуть любое число в ряду, кроме двух крайних. Оставшиеся числа сдвигаются, чтобы в ряду не оставалось пустого места. За вычеркивание числа следует заплатить сумму, равную произведению трёх чисел: вычеркиваемого числа и двух его соседей. В результате после $(N-2)$ вычёркиваний останется ряд из двух чисел: a_1 и a_N . Требуется определить наименьшую сумму, которую нужно заплатить, чтобы вычеркнуть все числа (кроме двух крайних, конечно).

Допустим, последним мы вычёркиваем число a_K . Тогда исходный ряд распадается на две части: от a_1 до a_K и от a_K до a_N . Понятно, что вычёркивание чисел из первой части (от a_2 до a_{K-1}) не влияет на вычёркивание чисел из второй (от a_{K+1} до a_{N-1}) и наоборот. Значит, нам надо знать наименьшую стоимость вычёркивания чисел в каждой части. Если мы это знаем для любого K , то переберём все значения K от 2 до $(N-1)$ и выберем наименьший результат. Иначе говоря, искомая наименьшая сумма равна $\min_{2 \leq K \leq N-1} (f(1, K) + f(K, N) + a_1 \cdot a_K \cdot a_N)$, где $f(1, K)$ и

$f(K, N)$ – те самые наименьшие стоимости вычёркивания на соответствующих отрезках. А чтобы найти $f(1, K)$ нам потребуется знать $f(1, M)$ и $f(M, K)$ для всевозможных M . Обратите внимание: знать $f(M, K)$ для всевозможных M при произвольном K , то есть нам надо знать $f(M, K)$ для произвольного отрезка исходного массива, а не только для отрезков, прижатых к какому-нибудь краю, как могло показаться на первый взгляд. Мы пришли к необходимости вычислять $f(L, R)$ – наименьшую стоимость вычёркивания всех чисел от $(L+1)$ -го до $(R-1)$ -го.

Получаем следующее решение.

$$f(L, R) = \min_{L+1 \leq K \leq R-1} (f(L, K) + f(K, R) + a_L \cdot a_K \cdot a_R), \text{ если } L < R-2$$

$$f(L, R) = 0, \text{ если } L = R-1$$

Реализация получается мгновенно: вычисляем последовательно $f(L,R)$ сначала для всех отрезков из двух чисел, т.е. $L=R-1$ (впрочем, вычисляем – это громко сказано, ведь все эти величины равны 0), затем – для отрезков из трёх чисел ($L=R-2$), из четырёх чисел, пяти чисел и т.д. вплоть до единственного отрезка из N чисел. Величина $f(1,N)$ содержит ответ.

Пример 2. Имеется прямоугольный лист клетчатой бумаги размером $N \times M$ клеток. Одним ходом разрешается разрезать его на две части. При этом разрез должен быть прямолинейным и проходить от одного края листа до противоположного, причём по границам клеток. С каждым из полученных кусков бумаги разрешается делать то же самое. Цель – добиться того, чтобы все получившиеся куски бумаги были квадратными. Требуется определить, какое наименьшее количество разрезов для этого надо сделать (или, что то же самое, наименьшее количество получаемых квадратов; действительно, количество полученных частей в любой момент времени на 1 больше количества сделанных к этому моменту разрезов).

Начало точно такое же: перебираем все возможные разрезы и, зная, какое наименьшее количество разрезов надо сделать для каждой из двух получающихся частей, находим наилучший разрез. Здесь нам понадобится вычислять величину $f(a,b)$ – наименьшее количество разрезов, необходимое для разрезания прямоугольника размером $a \times b$ клеток на квадраты. Соотношения для вычисления $f(a,b)$ получить несложно: перебираем все возможные разрезы (первые разрезы!) листа – и вертикальные, и горизонтальные; для каждого разреза вычисляем сумму минимального числа разрезов для обеих получающихся в результате частей, и выбираем разрез с наименьшей такой суммой. Не забыть только добавить потом 1 – сам первый разрез. Получаем такие формулы:

если $a \neq b$, то $f(a,b) = \min \{F_1, F_2\} + 1$, где

$$F_1 = \min_{1 \leq k \leq a-1} (f(k,b) + f(a-k,b)) ,$$

$$F_2 = \min_{1 \leq j \leq b-1} (f(a,j) + f(a,b-j))$$

если $a=b$, то $f(a,b) = 0$.

Конечно, здесь можно внести ряд усовершенствований, повышающих эффективность, например, заметив, что $f(a,b) = f(b,a)$, или, что для вычисления F_1 и F_2 достаточно перебирать только половину соответствующего размера:

$$F_1 = \min_{1 \leq k \leq a \div 2} (f(k,b) + f(a-k,b)) , \quad F_2 = \min_{1 \leq j \leq b \div 2} (f(a,j) + f(a,b-j))$$

А в каком порядке вычислять $f(a,b)$, т.е. заполнять массив значений f ? И ответов здесь много: вычисление $f(a,b)$ опирается на значения f для прямоугольников меньшей площади – можем заполнять в порядке роста площадей; вычисление $f(a,b)$ опирается на значения f для прямоугольников меньшего периметра – можем заполнять в порядке роста периметров; вычисление $f(a,b)$ опирается на значения f для прямоугольников, в которых один размер меньше соответствующего размера прямоугольника $a \times b$ – можем заполнять, добавляя поочередно по единичке к каждому размеру. Правда, с учётом $f(a,b) = f(b,a)$, это выглядит немножко смешно – достаточно увеличивать один размер, а увеличение второго получается автоматически. Но давайте усложним задачу: закрепим лист (запретим его поворачивать) и будем оценивать горизонтальные разрезы в 2 штрафные единицы, а вертикальные – в одну, имея целью, разумеется, минимизировать штраф. Тогда

соотношение $f(a,b) = f(b,a)$ не выполняется, и такой способ заполнения – поочерёдно увеличивая ширину и высоту – оказывается вполне естественным.

Итого, возвращаясь к формулировкам, приведённым в начале, в данной задаче подмассив – это прямоугольник, а размер подмассива – это уж как мы выберем – площадь прямоугольника, либо его периметр, либо пара длин сторон, причём увеличиваем каждую из двух длин по очереди. Можно, конечно, и ещё что-нибудь придумать, но вряд ли это будет естественным. Главное – задача решена применением всё того же общего приёма.

Пример 3. Имеется бревно длины L метров. На нём нанесены $(N-1)$ отметок, расположенных на расстоянии $a_1 < a_2 < a_3 < \dots < a_{N-1}$ от левого края бревна. Требуется распилить бревно на N частей, сделав распилы в отмеченных местах. При этом распилы делают на специальной машине, которая может за одно включение сделать два распила (но может и один). За каждое включение машины следует уплатить столько денег, какова длина распиливаемого в данный момент отрезка бревна. Пилить одновременно два отрезка бревна нельзя. Например, если $L=10$, $a_1=2$, $a_2=3$, $a_3=6$, и $a_4=7$, то мы можем поступить так: распилить бревно в точках a_1 и a_4 (это стоит 10 единиц), а затем полученный пятиметровый отрезок распилить в точках a_2 и a_3 . Общая стоимость – 15. Если же мы сначала распилим целое бревно в точках a_1 и a_2 , а затем полученный семиметровый отрезок распилим в точках a_3 и a_4 , то общая стоимость составит 17 единиц.

Дополним ряд меток двумя метками: $a_0=0$ и $a_N=L$. Обозначим через $f(L,R)$ наименьшую стоимость распиливания отрезка бревна от метки a_L до метки a_R . Нетрудно видеть, что

$$f(L,R) = \begin{cases} 0, & \text{если } L = R \text{ или } L = R - 1 \\ \min_{L < k \leq j < R} (f(L,k) + f(k,j) + f(j,R) + a_R - a_L), & \text{если } L < R - 1 \end{cases}$$

Нестрогое неравенство в цепочке $L < k \leq j < R$ сделано для того, чтобы учесть случай одного распила при включении.

И опять, как и в первом примере, нас интересует $f(0,N)$, и снова мы можем вычислять значения $f(L,R)$ в порядке возрастания их длины. Под длиной в данном случае подразумевается $R-L$ (а не длина отрезка бревна $a_R - a_L$ ☺).

Пример 4. А вот и вовсе задача с IOI. Правда, с IOI'96, давно это было, но тем не менее... Итак, задача.

Рассмотрим такую игру для двух игроков. Игровое поле – это ряд из чётного количества положительных чисел. Ход состоит в том, что игрок выбирает число, стоящее с левого или с правого края имеющегося ряда. Выбранное число удаляется с поля, а его величина прибавляется к сумме игрока. Игра заканчивается, когда закончатся все числа. Оба игрока играют наилучшим образом. Первый игрок выигрывает, если его сумма окажется не меньше суммы второго игрока. Известно, что для любого набора чисел у первого игрока имеется выигрышная стратегия.

Я не буду приводить дальнейшее условие детально, потому что мы его немножко изменим (вот только в сторону упрощения или усложнения? ☺). На олимпиаде от участников требовалось написать программу Library Reactive типа, т.е. программа должна была отвечать на ходы противника, которые она узнавала обращением к некоторой библиотечной функции. Мы же рассмотрим несколько

иную задачу: определить максимальную сумму, которую может собрать первый игрок при условии, что второй игрок играет наилучшим образом. Кажется, эта задача проще, но

1. переделать решение нашей задачи в решение задачи с олимпиады не представляет никаких проблем;
2. на олимпиаде не требовалось набирать максимальную сумму (или, иначе говоря, добиваться наибольшей разности сумм первого и второго игроков). А у такой задачи имеется очень красивое и очень легко реализуемое решение. Но об этом ниже.

Понятно, что задачи «набрать наибольшую сумму» и «достичь наибольшей разницы набранных игроками сумм» эквивалентны, поскольку вместе они набирают столько очков, какова сумма всех чисел в данном ряду, т.е. зная одно из двух – «набранная сумма» и «разность набранных сумм», второе вычисляется однозначно.

Обозначим данные числа $a_1, a_2, a_3, \dots, a_{N-1}, a_N$. Обозначим через $f(L, R)$ наибольшую разницу, которой может достичь первый игрок, если игра начинается с ряда чисел $a_L, a_{L+1}, a_{L+2}, \dots, a_{R-1}, a_R$. Если первый игрок возьмёт левое число, то он получает a_L очков, и перед вторым игроком открывается ряд чисел $a_{L+1}, a_{L+2}, \dots, a_{R-1}, a_R$, причём он ходит теперь первым. Понятно, что, поскольку второй игрок тоже играет наилучшим образом, он сократит разрыв в счёте на $f(L+1, R)$ очков, т.е. первый игрок в этом случае (ход слева) увеличит разность на $a_L - f(L+1, R)$ очков. Аналогично, если первый игрок возьмёт число справа, то он увеличит разность на $a_R - f(L, R-1)$ очков. Конечно же, первый игрок выберет лучший вариант, т.е.

$$f(L, R) = \max(a_L - f(L+1, R), a_R - f(L, R-1)), \text{ если } L < R$$

$$f(L, R) = a_L, \text{ если } L = R$$

Понятно, что если нас интересует, какой ход надо сделать, то следует выбирать тот из двух вариантов, на котором достигается максимум.

И опять срабатывает наш приём: сначала вычисляем $f(L, R)$ для отрезков из одного числа ($L=R$), затем для всех отрезков из двух чисел ($L=R-1$) и т.д. вплоть до отрезка из N чисел. $f(1, N)$ и есть ответ на вопрос нашей задачи. Заметим, кстати, что в решении никак не используется чётность числа N , т.е. решение годится и для нечётных.

Пожалуй, и всё на это раз. Осталось только привести обещанное красивое решение последней задачи. Вот оно.

Раскрасим наши числа в черный и белый цвета поочередно, т.е. 1-е, 3-е, 5-е ... числа будут чёрными, а 2-е, 4-е, 6-е ... числа – белыми. В начальной позиции слева находится чёрное число, справа – белое. Допустим, для определённости, что сумма всех чёрных чисел не больше суммы всех белых чисел. Тогда стратегия первого игрока – всегда брать белое число. Первым ходом он может это сделать. В результате на обоих краях оставшегося ряда стоят чёрные числа, второй игрок вынужден взять чёрное число, открывая тем самым доступ к соседнему с ним белому числу, которое возьмёт первый игрок и т.д. В конце все белые числа окажутся у первого игрока, все чёрные – у второго, т.е. сумма первого окажется не меньше суммы второго, что, собственно, и требовалось. Если сумма всех чёрных чисел больше суммы всех белых, то первый игрок должен всегда брать чёрные числа.

Понятно, что реализовать такое решение – проще простого.