

# Ряд чисел

## Решение .

Динамическим программированием отчётливо пахнет...

В самом деле, как разрезать полосу на  $K$  частей оптимальным образом? Очень просто: как-нибудь отрезать последний кусок полосы, а затем разделить оставшуюся часть полосы на  $(K-1)$  частей оптимальным образом. А как отрезать последний кусок полосы, в смысле, какого размера он должен быть? Тоже очень просто определить: перебрать все возможные варианты и выбрать из них наилучший.

Основная идея совершенно прозрачна. Давайте займёмся более формальным описанием:

- назовём *стоимостью отрезка полосы* сумму чисел, находящихся в этом отрезке;
- назовём *стоимостью разрезания полосы* стоимость самого дорогого отрезка полосы, образующегося в этом разрезании;
- назовем разрезание полосы на  $M$  частей *оптимальным*, если стоимость этого разрезания не превосходит стоимости любого другого разрезания полосы на  $M$  частей;
- обозначим  $t_M(i)$  стоимость оптимального разрезания на  $M$  частей полосы из первых  $i$  данных чисел;
- обозначим  $s(i)$  сумму первых  $i$  данных чисел:  $s(i) = a_1 + a_2 + \dots + a_i$ ,  $i=1, 2, \dots, N$ .

Теперь основную идею решения можно записать поточнее. Решаем такую задачу: найти  $t_M(i)$  – стоимость оптимального разрезания полосы из первых  $i$  чисел на  $M$  частей. Пусть последняя часть этого разрезания начинается с  $j$ -го числа. Тогда стоимость последней части равна  $a_j + a_{j+1} + \dots + a_i = s(i) - s(j-1)$ , стоимость оптимального разрезания оставшейся части полосы на  $(M-1)$  частей равна  $t_{M-1}(j-1)$ , и, в соответствии с вышесказанным,  $t_M(i) = \min_{M \leq j \leq i} [ \max ( t_{M-1}(j-1) , s(i) - s(j-1) ) ]$ . Заметим, что мы перебираем здесь значения  $j$ , начиная с  $M$ , потому, что для разрезания полосы на  $(M-1)$  частей требуется по крайней мере  $(M-1)$  чисел.

Соответствующий алгоритм решения выглядит так:

1. Инициализация.  $t_1(i) = s(i)$ ,  $i=1, 2, \dots, N$ .
2. Счёт. Для  $M=2, 3, \dots, K$  вычисляем последовательно  $t_M(i) = \min_{M \leq j \leq i} [ \max ( t_{M-1}(j-1) , s(i) - s(j-1) ) ]$ ,  $i= M, M+1, M+2, \dots, N$ .
3. Результат.  $t_K(N)$  – это ответ на вопрос задачи.

Оценим сложность полученного алгоритма. Легко видеть, что основная вычислительная работа происходит на шаге 2. Для каждого  $M$  ( $2 \leq M \leq K$ ) для вычисления  $t_M(i)$  ( $i= M, M+1, M+2, \dots, N$ ) требуется  $O((N-M)^2)$  операций. Для простоты можем считать, что  $K$  много меньше  $N$ , т.е. для каждого  $M$  счёт требует  $O(N^2)$  операций. Тогда весь алгоритм требует  $O(KN^2)$  операций.

Этот алгоритм реализован в программе `row0.pas`

Как обычно, займёмся улучшениями полученного алгоритма. Основное время занимает шаг 2 – мы перебираем значения  $j$  от  $M$  до  $i$  в поисках наименьшего значения

величины  $\max(t_{M-1}(j-1), s(i)-s(j-1))$ . Легко видеть, что с ростом  $j$  величина  $s(j)$  растёт, т.е. разность  $s(i)-s(j-1)$  убывает с ростом  $j$ . Понятно также (из чисто “физических” соображений), что величина  $t_{M-1}(j-1)$  растёт, точнее, не убывает с ростом  $j$ . Это наблюдение сразу позволяет включить бинарный поиск для поиска наименьшего значения  $\max(t_{M-1}(j-1), s(i)-s(j-1))$ ,  $M \leq j \leq i$ .

В самом деле, пусть  $j_0$  – последнее (наибольшее) значение  $j$ ,  $M \leq j \leq i$ , при котором  $t_{M-1}(j-1) \leq s(i)-s(j-1)$ . Тогда  $t_{M-1}(j_0) \geq s(i)-s(j_0)$ , и  $t_M(i) = \min(s(i) - s(j_0-1), t_{M-1}(j_0))$ , а  $j_0$  находим с помощью бинарного поиска.

В результате сложность шага 2 уменьшилась до  $O(N \cdot \log N)$ , и, соответственно, сложность алгоритма – до  $O(KN \cdot \log N)$ . Существенное улучшение. Его реализация приведена в файле `row1.pas`

Но на этом приключения не заканчиваются. Динамическое программирование – мощный метод, что и продемонстрировано в очередной раз в данной задаче. Но рядом с динамическим программированием очень часто просвечиваются *жадные алгоритмы*. Это не оскорбление, это общепринятая терминология. Жадные алгоритмы – это подход к решению задач оптимизации. Смысл этого подхода, в общем-то, понятен из названия, более точно, можно определить его примерно так:

1. разбиваем задачу на подзадачи, на шаги;
2. при решении каждой подзадачи стремимся получить максимальное приращение (или уменьшение, если мы хотим что-то минимизировать) целевой функции, причём руководствуемся только локальными критериями – не обращая внимания на предыдущие и последующие подзадачи (шаги);
3. решив подзадачу, больше к ней не возвращаемся.

Жадный подход далеко не всегда дает оптимальное решение, но во многих случаях получаемое решение оказывается достаточно близким к оптимальному, а для некоторых задач жадные алгоритмы дают оптимальное решение.

Подробнее о жадных алгоритмах можно прочитать, например, в уже упоминавшейся книге

*Т.Кормен, Ч.Лейзерзон, Р.Ривест, К.Штайн. Алгоритмы: построение и анализ, 2-е издание: М., Издательский дом “Вильямс”, 2005.*

Соответствующий фрагмент лежит в учебных материалах.

И, сразу уж, парочка ссылок:

<http://www.tspu.tula.ru/ivt/umr/ealg/docs/doc07/doc07.htm> - краткое описание жадного подхода и сравнение его с динамическим программированием

<http://rain.ifmo.ru/cat/view.php/theory/algorithm-analysis/greedy-2004> - уже знакомый участникам сайт: страничка, посвященная жадным алгоритмам

Но вернёмся к нашей задаче. Сразу не совсем понятно, что в ней можно сделать жадно. А жадно в ней можно ответить на вспомогательный вопрос: на какое минимальное количество частей можно разрезать полоску, при условии, что стоимость каждой части не превосходит  $C$ ? Легко видеть, что жадный подход здесь срабатывает –

просто отрезаем каждый раз от края полоски максимальный (по длине) кусок, стоимость которого не превосходит  $C$ . Полученное число частей и есть ответ на вспомогательный вопрос. В этом месте я советую остановиться ненадолго и ответить себе, почему же так получается. В данном случае так быстрее и проще, чем читать чьи-то объяснения...

А дальше что? А дальше заметим, что если существует разрезание полоски со стоимостью не больше  $C$  на сколько-то частей, то существует разрезание полоски и на любое большее количество частей с тем же условием – лишь бы чисел на полоске хватило. И вот теперь-то воспользуемся бинарным поиском!

“Вездесущий бинарный поиск!” - воскликнем мы вслед за Дж.Бентли и снова обратимся к книге

*Бентли Дж. Жемчужины программирования, 2-е изд. СПб.: Питер, 2002.*

Главы, посвященные бинарному поиску, выложены в учебных материалах к задаче 6.

С помощью бинарного поиска находим наименьшее число  $C$ , при котором полосу можно разрезать не больше, чем на  $K$  частей, так, чтобы стоимость каждой части не превосходила  $C$ . Это и есть ответ нашей задачи.

Одна проверка жадным алгоритмом требует  $O(N)$  операций, соответственно, сложность полученного алгоритма равна  $O(N \cdot \log A)$ , где  $A$  – сумма всех данных чисел. Это явно лучше, чем в предыдущем решении. Реализация приведена в файле `row2.pas`