Роман Едемский, родился 19 июля 1992 года в городе Киеве. Учился в “Британской Международной Школе” г. Киева с 1999–2003 г., в Техническом Лицее при НТУУ КПИ 2003–2005 г., в лицее № 171 “Лидер” с 2005–2009 г. В 2013 году получил диплом бакалавра по специальности “Прикладная математика” на факультете кибернетики Киевского национального университета имени Тараса Шевченка. С 2014 года учится в магистратуре факультета кибернетики.

Летом 2011 года стажировался в Google (Mountain View, California), а летом 2012 года — в Facebook (Menlo Park, California).

Работает в киевском офисе Yandex.

Основные достижения:

- Участник отборов на IOI 2009.
- 1 место на SEERC 2012 и 2 место на SEERC 2013.
- Серебряная медаль на ACM ICPC World Finals 2013 в составе команды Киевского национального университета (г. Санкт-Петербург, Россия).
- 3 место на Onsite раунде открытого кубка им. Е.В. Панкратьева в составе команды BZFlags.
- Автор задач для Всеукраинской олимпиады школьников по информатике.

Теоретический материал. Оптимизация динамического программирования за счет использования свойств линейных функций и алгоритма Грэхема

Постановка задачи 1

Рассмотрим следующую задачу. Есть N городов, расположенных на одной прямой. Город с номером i имеет координату x_i (координаты считаются начиная от какой-то точки на прямой и измеряются в километрах), причем координаты всех городов различны. Все города пронумерованы в порядке увеличения их координат. Столица имеет номер 1. В каждом городе находится посланник, который в случае необходимости может доставить письмо в столицу. Каждый посланник характеризуется двумя числами: S_i и V_i. Здесь S_i — это время, необходимое посланнику для сборов в дорогу, а V_i — это время в минутах, за которое посланник i проходит один километр пути. Процесс доставки письма из города i ($i > 0$) в столицу выглядит следующим образом: посланник из города i собирается в течении S_i минут, после чего он двигается в город с номером $i - 1$. Затем, он либо продолжает движение без остановок, либо передает письмо посланнику, находящемуся в городе $i - 1$, после чего тот доставляет письмо по аналогичной схеме, тратя время на сборы и т.д.. В общем случае, в процессе доставки письма в столицу может принимать участие сколько угодно посланников. Требуется для каждого города вычислить наименьшее время, за которое письмо из него может быть доставлено в столицу.

Решение задачи 1

Будем решать задачу при помощи динамического программирования. Обозначим через $f(i)$ наименьшее время, необходимое для доставки письма из города i в столицу. Очевидно, $f(1) = 0$. Запишем формулу для вычисления $f(i)$ через уже вычисленные $f(j)$ для всех $j < i$.

$$f(i) = \min_{j<i}\{(x_i - x_j) \cdot V_i + S_i + f(j)\}$$

Приведенная рекуррентность уже позволяет решить задачу за $O(N^2)$. Для дальнейшей оптимизации раскроем скобки и вынесем из под минимума переменные, не зависящие от j:

$$f(i) = \min_{j<i}\{f(j) - x_j \cdot V_i\} + x_i \cdot V_i + S_i$$

Основная сложность заключается в том, чтобы вычислить минимум по j быстрее, чем за $O(N)$. Для этого дадим некую геометрическую интерпретацию этой формуле. Вспомним, что любую не вертикальную прямую можно задать уравнением вида $y = k \cdot x + l$, где k и l — это произвольные параметры, x — независимая переменная, а y — зависимая переменная. Число k еще называют угловым коэффициентом прямой. Итак, поставим в соответствие каждому городу j ($j < i$) прямую с $k = -x_j$ и $l = f(j)$. Мы можем такое сделать, поскольку для всех таких городов мы уже знаем значение $f(j)$. Тогда задача поиска минимума по j значения функции $f(j) - x_j \cdot V_i$ сводится к такой: есть набор прямых, заданных коэффициентами k и l, и требуется найти наименьшую y-координату пересечения прямых из этого набора с вертикальной прямой $x = V_i$. Для каждого x' выберем самую нижнюю точку пересечения вертикальной прямой $x = x'$ и прямых из нашего набора. Назовем получившееся множество точек нижним огибающим множеством для заданного набора прямых.

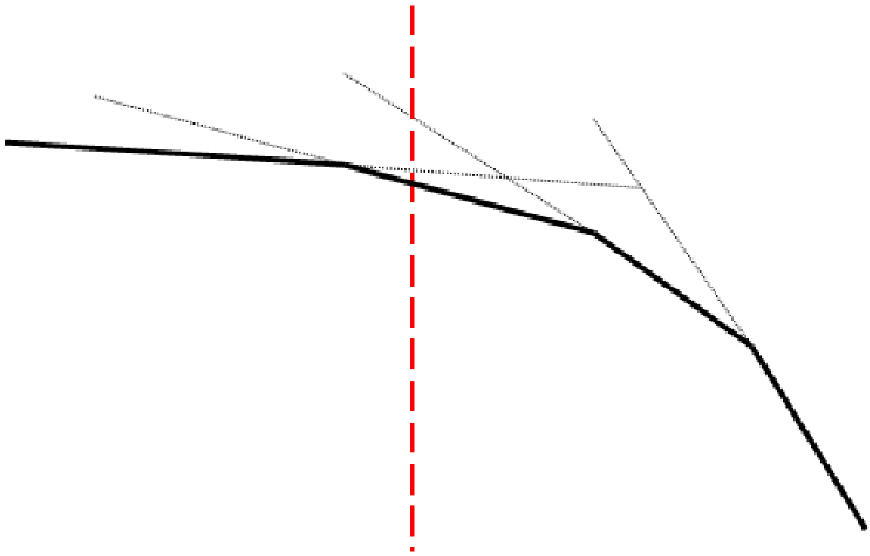

Рисунок 1.1 — Нижнее огибающее множество выделено черным

Что нам это дает? Как минимум то, что теперь нам уже не нужно пересекать вертикальную прямую со всеми прямыми из набора и выбирать самую нижнюю точку, а достаточно лишь пересечь вертикальную прямую $x = V_i$ с нижним огибающим множеством, после чего сказать, что ордината точки пересечения и есть искомым значением минимума по j.

Итак, если мы научимся поддерживать нижнее огибающее множество для набора прямых, а также быстро пересекать его с вертикальными прямыми, то мы сможем быстро вычислять $f(i)$ через $f(j)$ для $j < i$ и, как следствие, решать задачу быстрее, чем за $O(N^2)$. Для начала разберемся с тем, как такое множество поддерживать, а именно, как нужно перестроить нижнее огибающее множество после добавления в наш набор новой прямой (после того, как мы уже вычислили $f(i)$). Во-первых, заметим, что в силу специфики задачи, угловые коэффициенты всех прямых строго убывают при увеличении номера, поскольку координаты городов строго возрастают, а угловой коэффициент $k_i = -x_i$. Следовательно, при добавлении в набор новой прямой, ее часть точно войдет в нижнее огибающее множество, поскольку начиная с некоторого x именно эта новая прямая будет иметь наименьший y среди всего набора. В то же время, некоторые прямые уже не будут ничего вкладывать в нижнее огибающее множество и про них в дальнейшем можно забыть.

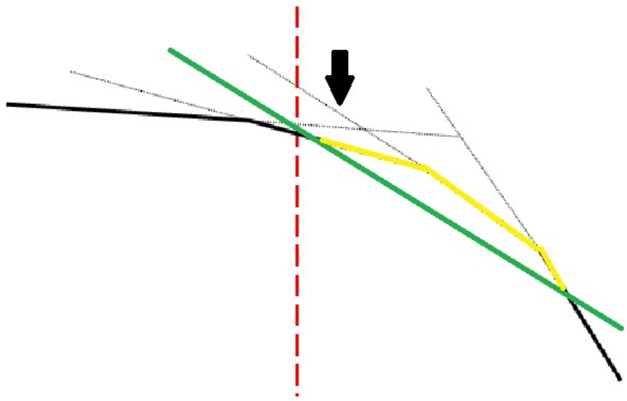

Рисунок 1.2 — Зеленым обозначена добавляемая прямая, желтым — та часть старого нижнего огибающего множества, которая больше ему не принадлежит, а стрелкой — прямая, которая теперь вообще не имеет пересечения с нижним огибающим множеством

Заметим, что нижнее огибающее множество состоит из отрезков и лучей, являющихся частями прямых из набора, причем угловые коэффициенты этих прямых убывают слева направо, а также каждая прямая вносит не более одного отрезка (или луча). Следовательно, для того, чтобы полностью задать нижнее огибающее множество, достаточно хранить номера прямых, части которых в него входят. Новая прямая всегда будет добавляться в конец этого массива, поскольку ее угловой коэффициент строго меньше, чем угловые коэффициенты прямых из нижнего огибающего множества. Тогда для того, чтобы добавить новую прямую в нижнее огибающее множество, выполним следующий алгоритм.

AddLine(L, S){

```
  /* L  --- добавляемая прямая, S  --- массив
  прямых, входящих в нижнее огибающее множество */
  Пока в массиве больше одной прямой :
    Взять последнюю прямую (L_1) из S
    Взять предпоследнюю прямую (L_2) из S
    Найти X точки пересечения x_1 прямых L_1 и L_2
    Найти X точки пересечения x_2 прямых L_1 и L
    Если x_1 < x_2 :
      Остановить выполнение цикла
    иначе
      Удалить прямую L_1 из конца массива

  Добавить прямую L в конец списка
}
```

Таким образом, мы последовательно будем удалять из конца массива те прямые, части которых больше не будут входить в нижнее огибающее множество, после чего добавим новую прямую в конец массива. Ясно, что каждая прямая будет удаляться из списка не более одного раза, поэтому в сумме мы потратим $O(N)$ операций на поддержание нижнего огибающего множества.

Итак, поддерживать нижнее огибающее множество мы научились, теперь осталось быстро находить его пересечение с вертикальной прямой. Это можно сделать при помощи бинарного поиска. Заметим, что точки пересечения соседних прямых в порядке их хранения в нашем списке отсортированы по возрастанию X координаты, поэтому мы можем делать бинарный поиск по ним для того, чтобы найти две точки соседние пересечения прямых из нижнего огибающего множества, которые лежат, соответственно, слева и справа от нашей вертикальной прямой. Теперь дело осталось за малым — подставить X координату вертикальной прямой в ту прямую, которая проходит через найденные 2 точки пересечения и найти Y координату пересечения вертикальной прямой и нижнего огибающего множества. Сложность нахождения этого пересечения — $O(\log N)$ на запрос. Следовательно, всю задачу мы теперь умеем решать за $O(N \log N)$.

Постановка задачи 2

Дан массив, состоящий из N целых положительных чисел. Также задана квадратичная функция $g(X) = a \cdot x^2 + b \cdot x + c$ своими коэффициентами a, b и c, причем $a < 0$. Требуется разбить массив на последовательные отрезки так, чтобы сумма значений g по всем отрезкам была максимальной, где под значением функции от отрезка подразумевается значение функции от суммы чисел этого отрезка.

Решение задачи 2

Будем решать задачу при помощи динамического программирования. Обозначим через f_i наибольшую сумму значений функции g, если мы будем рассматривать (и разбивать на отрезки) только первые i элементов нашего массива. Будем считать, что $f(0) = 0$. Тогда можем записать формулу:

$$f(i) = \max_{j \le i}\{a \cdot sum(j,i)^2 + b \cdot sum(j,i) + c + f(j-1)\}$$

Под $sum(j,i)$ подразумевается сумма всех чисел на отрезке от j до i. Понятно, что подсчет такой динамики требует $O(N^2)$ операций. Для того, чтобы его ускорить, перепишем немного формулу, стоящую под минимумом, обозначив $sum(1,j)$ через $\sigma(j)$ ($\sigma(0) = 0$):

$$\begin{aligned} &f(j-1) + a \cdot (\sigma(i) - \sigma(j-1))^2 + b \cdot (\sigma(i) - \sigma(j)) + c = \\ &= f(j-1) + a \cdot (\sigma(i)^2 - 2\sigma(i)\sigma(j-1) + \sigma(j-1)^2) + b \cdot (\sigma(i) - \sigma(j-1)) + c = \\ &= (a\sigma(i)^2 + b\sigma(i) + c) + f(j-1) - 2a\sigma(i)\sigma(j-1) + a\sigma(j-1)^2 - b\sigma(j-1) \end{aligned}$$

Теперь запишем, чему равно $f(i)$:

$$f(i) = \max_{j \le i}\{(a\sigma(i)^2 + b\sigma(i) + c) + f(j-1) - 2a\sigma(i)\sigma(j-1) + a\sigma(j-1)^2 - b\sigma(j-1)\}$$

Введем следующие обозначения: $t = (a\sigma(i)^2 + b\sigma(i) + c)$, $k = -2a\sigma(j-1)$, $x = \sigma(i)$, $l = f(j-1) + a\sigma(j-1)^2 - b\sigma(j-1)$. Тогда $f(i)$ выражается следующим образом:

$$f(i) = \max_{j \le i}\{kx + l\} + t$$

Теперь у нас задача свелась к такой, которую мы уже умеем решать (см. предыдущую задачу из лекции). Единственное различие в том, что здесь нам нужно хранить верхнее огибающее множество, а не нижнее. То есть, мы можем решить эту задачу за $O(N)$.

Можно еще заметить, что поскольку $\sigma(i)$ монотонно возрастает при увеличении i, то абсцисса точки пересечения вертикальной прямой $x = \sigma(i)$ и верхнего огибающего множества будет двигаться только вправо. Следовательно, мы можем хранить указатель на текущий отрезок из верхнего огибающего множества, который пересекается с $x = \sigma(i)$ и двигать его при пересчете f для больших i. Таким образом, мы можем избавиться от бинарного поиска и добиться итоговой асимптотики решения $O(N)$.