Михаил Густокашин. Бинарный поиск.

1 Бинарный поиск в упорядоченных массивах

Под упорядоченным массивом будем понимать массив, упорядоченный по неубыванию, т.е. $a[1] \leq a[2] \leq \ldots \leq a[N]$.

У нас имеется заданная своими границами область поиска. Мы выбираем ее середину, и, если искомый элемент меньше, чем средний, то поиск осуществляется в левой части, иначе — в правой. Действительно, если искомый элемент меньше среднего, то и меньше всех элементов, которые находятся правее среднего, а значит, их сразу можно исключить из рассмотрения. Аналогично для случая, когда искомый элемент больше среднего.

Код, осуществляющий бинарный поиск в упорядоченном массиве выглядит так:

```
while (l<r) {
  m=(l+r)/2;
  if (a[m]<k) l=m+1;
  else r=m;
}
if (a[r]==k) printf("%d", r);
else printf("-1");
```

Перед выполнением этого кода следует присвоить переменным l и r значения 0 и $n-1$ соответственно. В случае если элемент не найдем, эта программа выводит -1.

Сложность алгоритма бинарного поиска составляет $O(\log N)$, где N — количество элементов в массиве.

2 Бинарный поиск для монотонных функций

Бинарный поиск может использоваться не только для поиска элементов в массиве, но и для поиска корней уравнений и значений монотонных (возрастающих или убывающих) функций. Напомним, что функция называется возрастающей, если $\forall x_1, x_2 : x_1 > x_2 \Rightarrow f(x_1) > f(x_2)$ (для любых x_1 и x_2, если $x_1 > x_2$, то $f(x_1)$ также больше $f(x_2)$).

Действительно, так же как и в массиве, мы можем исключить из рассмотрения половину текущей области, если нам заведомо известно, что там не существует решения. В случае же, если функция не монотонна, то воспользоваться бинарным поиском нельзя, т.к. он может выдавать неправильный ответ, либо находить не все ответы.

Для примера рассмотрим задачу поиска кубического корня. Кубическим корнем из числа x (обозначается $\sqrt[3]{x}$) называется такое число y, что $y^3 = x$.

Сформулируем задачу так: для данного вещественного числа x ($x \geq 1$) найти кубический корень с точностью не менее 5 знаков после точки.

Функция при $x \geq 1$, ограничена сверху числом x, а снизу — единицей. Таким образом, за нижнюю границу мы выбираем 1, за верхнюю — само число x. После этого делим текущий отрезок пополам, возводим середину в куб и если куб больше x, то заменяем верхнюю грань, иначе — нижнюю.

Код будет выглядеть следующим образом:

```
r = x;
l = 1;
while (fabs(l-r)>eps) {
  m=(l+r)/2;
  if (m*m*m<x) l=m;
  else r=m;
}
```

Для того чтобы пользоваться функцией `fabs`, необходимо подключить библиотеку `math.h`.

3 Бинарный поиск по ответу

Во многих задачах в качестве ответа необходимо вывести какое-либо число. При этом достаточно легко сказать, больше ли это число, чем нужно, или меньше, несмотря на то, что вычисление самого ответа может быть довольно трудоемкой операцией. В таком случае мы можем выбрать число заведомо меньшее ответа и число заведомо большее ответа, а правильное решение искать бинарным поиском.

Для примера рассмотрим решение задач нескольких прошедших олимпиад.

Очень легкая задача

Московская олимпиада по информатике 2006-2007

Сегодня утром жюри решило добавить в вариант олимпиады еще одну, Очень Легкую Задачу. Ответственный секретарь Оргкомитета напечатал ее условие в одном экземпляре, и теперь ему нужно до начала олимпиады успеть сделать еще N копий. В его распоряжении имеются два ксерокса, один из которых копирует лист за секунд, а другой — за y. (Разрешается использовать как один ксерокс, так и оба одновременно. Можно копировать не только с оригинала, но и с копии.) Помогите ему выяснить, какое минимальное время для этого потребуется.

Формат входных данных

Во входном файле записаны три натуральных числа N, x и y, разделенные пробелом ($1 \le N \le 2 \times 10^8$, $1 \le x, y \le 10$).

Формат выходных данных

Выведите одно число — минимальное время в секундах, необходимое для получения N копий.

Примеры

Входные данные	Выходные данные
4 1 1	3
5 1 2	4

Существует конструктивное решение этой задачи (формула), которую можно вывести и, при желании, доказать. Однако очень легко реализовать решение этой задачи с помощью бинарного поиска.

Первую страницу мы копируем за $min(x, y)$ секунд и, затем, рассматриваем решение уже для $N-1$ страницы.

Пусть l - минимальное время, r - максимальное. Минимум нам необходимо потратить 0 секунд, максимум, например $(N-1) \times x$ секунд (страницы делаются полностью на одном ксероксе). Считаем среднее значение и смотрим, сколько полных страниц можно напечатать за это время, используя оба ксерокса. Если количество страниц меньше $N-1$, то мы меняем нижнюю границу, иначе — верхнюю.

```
int main(void)
{
  int n, x, y, i, j, l, r, now;
  double speed;
  scanf("%d%d%d", &n, &i, &j);
  x=i<j?i:j;
  y=i>j?i:j;
  l=0;
  r = (n-1)*y;
  while (l != r) {
    now = (l+r)/2;
    j = now / x + now / y;
    if (j < n-1) l = now+1;
    else r = now;
  }
  printf("%d", r+x);
  return 0;
}
```

Автобус

13 Украинская олимпиада

Служебный автобус совершает один рейс по установленному маршруту и в случае наличия свободных мест подбирает рабочих, которые ожидают на остановках, и отвозит их на завод. Автобус также может ждать на остановке рабочих, которые еще не пришли. Известно время прихода каждого рабочего на свою остановку и время проезда автобуса от каждой остановки до следующей. Автобус приходит на первую остановку в нулевой момент времени. Продолжительность посадки рабочих в автобус считается нулевой.

Задание: Написать программу, которая определит минимальное время, за которое автобус привезет максимально возможное количество рабочих.

Формат входных данных

Входной текстовый файл в первой строке содержит количество остановок N и количество мест в автобусе M. Каждая i-я строчка из последующих N строчек содержит целое число — время движения от остановки i к остановке $i+1$ ($N+1$-я остановка — завод), количество рабочих K, которые придут на i-ю остановку, и время прихода каждого рабочего на эту остановку в порядке прихода ($1 \le M \le 2000, 1 \le N, K \le 200000$).

Формат выходных данных

Единственная строка выходного текстового файла должен содержать минимальное время, необходимое для перевозки максимального количества рабочих.

Примеры

Входные данные	Выходные данные
3 5 1 2 0 1 1 1 2 1 4 0 2 3 4	4

Сначала определим, что такое максимально возможное количество рабочих. Если общее количество рабочих больше вместимости автобуса, то это — объем автобуса, если же рабочих меньше чем вместимость автобуса — то это количество всех рабочих (в этом случае вместимости автобуса уместно присвоить значение, равное количеству людей).

Когда мы считываем данные, следует определить время прихода последнего человека (т.е. то время, когда уже все люди будут на остановках) — это будет максимум в бинарном поиске. Минимум будет равен нулю. Если автобус должен задержаться перед остановкой, то он должен сделать это перед первой остановкой (действительно, если он подъедет к первой остановке, заберет людей, а потом будет ждать у второй остановки, то в это время на первую могут придти еще люди, а если ждать перед первой, то люди со второй никуда не денутся). Минимум и максимум у нас есть. Теперь берем задержку, равную $x = (min + max)/2$. С помощью процедуры, которая будет описана ниже, вычисляем, сколько людей успеет придти до момента x на первую остановку, для второй остановки будет задержка, равная $x + a[1]$, где $a[1]$ — время следования от первой остановки до второй, для третьей задержка — $x+a[1]+a[2]$ и т.д. Если количество севших в автобус на всех остановках больше либо равно вместимости автобуса, то надо заменить x на $(min + x)/2$, если остались места в автобусе то $x = (x + max)/2$. Условие выхода будет такое: если при некоторой задержке x автобус заполнен, а при задержке $(x - 1)$ автобус не полон, то ответ x.

Теперь второй бинарный поиск. Тот самый, который определяет, сколько людей успеет придти на определенную остановку до определенного момента. Здесь максимумом дихотомии будет количество людей на остановке, а минимумом — ноль. Выбираем среднего человека — если его время прихода меньше, чем задержка, то $x = (x + max)/2$, если он не успеет придти, то $x = (min + x)/2$. Здесь условие выхода такое: если человек успевает придти на остановку, а следующий за ним нет — то ответом будет номер человека. Отдельно нужно обрабатывать случай, если на автобус сядут все люди с остановки.