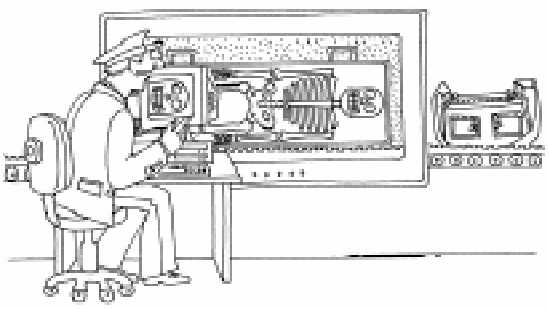
Маврин Павел Юрьевич

РАЗБОР ЗАДАЧ ВТОРОЙ ВСЕРОССИЙСКОЙ ОЛИМПИАДЫ ПО ИНФОРМАТИКЕ ЗАТО

ЗАДАЧА A. ТАМОЖНЯ



Идет 2163 год. Мишу, который работает в отделении таможни при космодроме города Нью-Питер, вызвал в кабинет шеф.

Как оказалось, недавно Министерство Налогов и Сборов выделило отделению определенную сумму денег на установку новых аппаратов для автоматического досмотра грузов. Естественно, средства были выделены с таким расчетом, чтобы грузы теперь находились на таможне ровно столько времени, сколько требуется непосредственно на их досмотр.

В руках шефа каким-то образом оказались сведения о надвигающейся ревизии – список из N грузов, которые будут контролироваться Министерством. Для каждого груза известны время его прибытия, отсчитываемое с некоторого момента, хранимого в большом секрете, и время, требуемое аппарату для обработки этого груза. Шеф дал Мише задание по этим данным определить, какое минимальное количество аппаратов необходимо заказать на заводе, чтобы все грузы Министерства начинали досматриваться сразу после прибытия. Необходимо учесть, что конструкция тех аппаратов, которые было решено установить, не позволяет обрабатывать два груза одновременно на одном аппарате. Напишите программу, которая поможет Мише справиться с его задачей.

Формат входных данных

На первой строке входного файла задано число N ($0 \le N \le 50000$). На следующих N строках находится по 2 целых положительных числа T_i и L_i – время прибытия соответствующего груза и время, требуемое для его обработки ($1 \le T_i \le 10^6$, $1 \le L_i \le 10^6$).

Формат входных данных

В выходной файл выведите одно число – наименьшее количество аппаратов, которое нужно установить, чтобы не вызвать подозрений у Министерства.

Пример

customs.in	customs.out
3 3 2 4 2 5 2	2
5 13 4 15 1 11 5 12 3 10 3	3

Решение

Необходимое количество аппаратов равно максимальному количеству грузов, проверяемых одновременно. Как найти момент времени, в который проверятся наибольшее количество грузов? Это количество установится после того, как на проверку придет один из грузов. Будем перебирать такие моменты в порядке их возникновения. При этом мы можем быстро пересчитывать количество обрабатываемых грузов при помощи структуры «куча» (также называется «пирамида», подробнее об этой структуре можно прочитать в [1]), которая позволяет находить и удалять минимум N хра-

нящихся в ней элементов за время $O(\log N)$. Поместим все грузы, находящиеся на обработке в данный момент в кучу по времени окончания обработки, теперь перед проверкой очередного момента прибытия будем удалять из кучи грузы, обработка которых завершится до того, как поступит этот груз. После этого добавим новый груз в кучу и сравним счетчик с максимумом.

Программа

Для сортировки грузов по времени прибытия используем быструю сортировку, подробное описание которой также можно найти в [1]:

```
procedure sort(ll, rr : integer);
var i, j, m, y : integer;
begin
  m := t[ll + random(rr - ll + 1)];
  i := ll;
  j := rr;
  while i <= j do
  begin
    while t[i] < m do inc(i);
    while t[j] > m do dec(j);
    if i <= j then
    begin
      y := t[i];
      t[i] := t[j];
      t[j] := y;
      y := l[i];
      l[i] := l[j];
      l[j] := y;
      inc(i);
      dec(j);
    end;
   end;
   if i < rr then sort(i, rr);
   if ll < j then sort(ll, j);
end;
```

Далее реализуем процедуры добавления элемента и удаления минимума для кучи:

```
procedure add(x : integer);
var i, y : integer;
begin
  inc(nh);
  heap[nh] := x;
  i := nh;
  while heap[i] < heap[i div 2] do
  begin
    y := heap[i];
    heap[i] := heap[i div 2];
    heap[i div 2] := y;
    i := i div 2;
  end;
end;

procedure extract_min;
var i, y : integer;
begin
  heap[1] := heap[nh];
  dec(nh);
  i := 1;
  while (i * 2 <= nh) do
  begin
    if i*2 = nh then
    begin
      if heap[i] > heap[i*2] then
      begin
        y := heap[i];
        heap[i] := heap[i*2];
        heap[i*2] := y;
      end;
      exit;
    end else
    begin
      if heap[i*2]<heap[i*2+1] then
      begin
       if heap[i] > heap[i*2] then
       begin
         y := heap[i];
         heap[i] := heap[i*2];
         heap[i*2] := y;
         i := i * 2;
       end else exit;
      end else
      begin
       if heap[i] > heap[i*2+1] then
       begin
         y := heap[i];
         heap[i] := heap[i*2+1];
         heap[i*2+1] := y;
         i := i * 2 + 1;
       end else exit;
      end;
    end;
  end;
end;
```

Теперь главная часть программы выглядит совсем просто:

Сортируем грузы:

```
sort(1, n);
```

Инициализируем кучу:

```
nh := 0;
h[0] := -1000000000;
```

Теперь для каждого груза от 1 до n делаем следующее:

Удаляем из кучи все грузы, проверка которых уже закончена:

```
while (nh > 0) and (heap[1] <= t[i])
do extract_min;
```

Теперь заносим в кучу новый груз в кучу:

```
add(t[i] + l[i]);
```

И сравниваем счетчик с максимумом:

```
if best < nh then best := nh;
```

ЗАДАЧА B. ДОМИНОШКИ

Дан прямоугольник $N \times M$. Требуется узнать, можно ли его разбить на доминошки (фигуры 1×2 или 2×1).

Формат входных данных

Входной файл содержит два натуральных числа N и M ($1 \le N, M \le 10000$).

Формат входных данных

В выходной файл выведите «YES» (без кавычек), если прямоугольник $N \times M$ можно разбить на доминошки, «NO» (без кавычек) в противном случае.

Пример

domino.in	domino.out
3 3	NO
2 2	YES

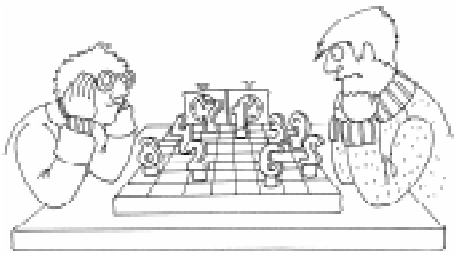
Решение

Задача решается совсем просто: если одно из чисел четно, то, направив все доминошки вдоль этой стороны прямоугольника, получим искомое замощение. Если же оба числа нечетные, то общее количество клеток NM тоже нечетное, следовательно, его нельзя покрыть доминошками из двух клеток.

Программа

```
read(n, m);
if (n mod 2 = 0) or (m mod 2 = 0) then
  writeln('YES')
else
  writeln('NO');
```

ЗАДАЧА C. МАТРИЦА



Дан прямоугольник $N \times M$, в каждой клетке которого стоит целое число. Затем с каждой строкой Петя делает следующее: сначала находит в ней максимум. Затем стирает все числа в этой строке, равные ему (в том числе и его самого). В результате этого получается новая строка, которая записывается обратно в прямоугольник. После этого, если в получившейся строке меньше чем i чисел (где i – номер строки), то строка полностью заполняется нулями, иначе начиная с i-ой позиции обратно выписываются стертые максимумы, а все остальное сдвигается вправо.

Например, если на первом шаге была строчка 1 3 2 3, то максимум в ней – 3. Он стирается. Получается строчка 1 2. Далее начиная с первой позиции вставляется 3 3. Получается 3 3 1 2. По данному прямоугольнику требуется вывести результат применения к нему данных операций.

Формат входных данных

Первая строка входного файла содержит два натуральных числа N и M ($1 \le N, M \le 100$) Далее идет N строк по M целых чисел в каждой. (Все числа по модулю не превосходят 10^6).

Формат входных данных

В выходной файл выведите результат применения этих действий.

Пример

matrix.in	matrix.out
3 4 1 3 3 3 4 4 4 4 1 2 3 4	3 3 3 1 0 0 0 0 1 4 2 3

Решение

В этой задаче не требуется придумывать сложных алгоритмов, нужно просто аккуратно сделать то, что написано в условии задачи.

Программа

Для каждой строки *i* делаем следующее:

Находим максмум:

```
mx := a[i, 1];
for j := 1 to m do
if a[i, j] > mx then mx := a[i, j];
```

Находим количество элементов, оставшихся после удаления максимумов:

```
k := 0;
for j := 1 to m do
if a[i, j] <> mx then inc(k);
```

Если их меньше *i*, то заполняем строку нулями:

```
if k < i then
  for j := 1 to m do a[i, j] := 0;
```

Иначе сдвигаем эти элементы к левому краю:

```
k := 0;
for j := 1 to m do
if a[i, j] <> mx then
begin
 inc(k);
 a[i, k] := a[i, j];
end;
```

Теперь освобождаем место для вставки максимумов, сдвигая нужное число элементов к правому краю:

```
q1 := k;
q2 := m;
while q1 >= i do
begin
 a[i, q2] := a[i, q1];
 dec(q1);
 dec(q2);
end;
```

А на освободившееся место вставляем максимумы:

```
for j := i to i + m - k - 1 do
                    a[i, j] := mx;
```

ЗАДАЧА D. ЕДИНИЦЫ

Найдите квадрат числа, десятичная запись которого состоит из *n* единиц.

Формат входных данных

Входной файл содержит единственное число n ($1 \le n \le 10^5$).

Формат входных данных

Выведите искомый квадрат.

Пример

ones.in	ones.out
2	121
9	12345678987654321

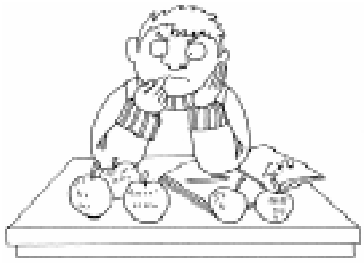
Решение

Число, десятичная запись которого состоит из *n* единиц очевидно равно $(10^n - 1)/9$, тогда его квадрат равен $(10^n - 1)^2 / 81$. Несложно убедиться, что число $(10^n - 1)^2$ имеет следующую структуру: 99...9800...1 (число девяток и нулей равно $(n - 1)$). Теперь, зная это число, можно воспользоваться стандартным алгоритмом деления числа «уголком» на 81 и получить искомое.

Программа

```
r := 0;
for i := 1 to 2*n do
begin
  r := r * 10;
  if i < n then r := r + 9 else
  if i = n then r := r + 8 else
  if i = 2*n then r := r +1;
  if i > 1 then
 write(chr(ord('0') + (r div 81)))
  r := r mod 81;
end;
```

ЗАДАЧА E. СУММА



Дано два натуральных числа, найдите их сумму.

Формат входных данных

Входной файл содержит два натуральных числа a и b ($1 \le a, b \le 10^{100}$).

Формат входных данных

Выведите в выходной файл искомую сумму без ведущих нулей.

Пример

sum.in	sum.out
361 239	600

Решение

Поскольку числа a и b не умещаются ни в один из стандартных типов, приходится действовать сложнее. Запишем эти числа в массивы, по одной цифре в ячейку, после чего используем обычный алгоритм сложения чисел «в столбик».

Программа

Прочитаем данные числа в строки, после чего запишем их по цифрам в массивы (цифры заносятся в массив, начиная с последней):

```
for i := 1 to length(s) do
a[i] := ord(s[length(s) - i + 1]) - ord('0');
j := 0;
```

Теперь сложим эти числа, занося результат в массив c:

```
j := 0;
for i := 1 to 101 do
begin
  j := j div 10 + a[i] + b[i];
  c[i] := j mod 10;
end;
```

Осталось вывести результат, пропустив ведущие нули:

```
j := 101;
while (c[j] = 0) and (j > 1) do
dec(j);
for i := j downto 1 do write(c[i]);
```

Замечание: Для всех задач ограничение по времени – 2 секунды, ограничение по памяти – 8 мегабайт.

Литература

1. *Вирт Н.* Алгоритмы и структуры данных. СПб.: «Невский диалект», 2001.

Маврин Павел Юрьевич,
студент 2 курса СПбГУ ИТМО.