Задание №1

Некрасивые числа

ограничение по времени на тест :*1 секунда*
ограничение по памяти на тест: *32 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубик любит красивые числа. По мнению Кубика, число не считается красивым, если оно делится на 5 или на 7 (Например, число *151263* – некрасивое по мнению Кубика, так оно делится на 7). Кубик хочет узнать является ли имеющееся у него число некрасивым. Помогите Кубику решить эту задачу.

Входные данные:
Целое число a ($1 \le a \le 5*10^{18}$).

Выходные данные:
Выведите YES, если число некрасивое, и NO – иначе.

Пример
входные данные
151263
выходные данные
YES

РЕШЕНИЕ
Необходимо проверить введенное число на делимость на 5 или 7
Пример решения на Pascal

```
var
  a : int64;
begin
  readln(a);
  if (a mod 5=0)or (a mod 7=0) then
    writeln('YES')
  else
    writeln('NO');
end.
```

Задача 2

Шарик в ящик

ограничение по времени на тест: *1 секунда*
ограничение по памяти на тест: *256 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

У Кубика есть мешок n шариков и m ящиков. Каждый шарик характеризуется радиусом r, а каждый ящик длиной a, шириной b, высотой c внутреннего пространства ящика. Шарик можно положить в ящик, если он полностью помещается в нем. Кубик кладет по одному шарику в ящик. Помогите Кубику определить минимальное количество шариков, которые он не сможет положить в ящики.

Входные данные:
В первой строке два целых числа n, m ($1 \le n \le 5*10^5$, $1 \le m \le 5*10^5$) — количество шариков и ящиков соответственно.
Во второй строке заданы n целых чисел r_1, r_2, …, r_n — радиусы шариков ($r_i < 10^5$, $i = 1..n$).
Следующие m строк содержит 3 целых числа a, b, c ($1 \le a < b \le 5*10^{15}$) — размеры ящиков .

Выходные данные:
Выведите число – ответ на задачу.

Пример
входные данные
5 6
1 2 3 4 5
1 2 2
2 3 2
2 3 4
5 5 5
6 6 6
7 7 7
выходные данные
2

Примечание
Шарики с радиусами 4 и 5 не поместятся в ящики

РЕШЕНИЕ
Для решения задачи необходимо построить два отсортированных массива: массива радиусов шаров и минимальных размеров ящиков. Затем последовательно ввести два указателя, по одному на каждый массив, и для каждого шарика определить может ли он быть положен в текущий ящик. Если шарик вмешается в ящик, то счетчик свободных

ящиков уменьшается на единицу и указатель текущего шарика увеличивается на единицу. Указатель текущего ящика увеличивается на единицу.

Пример решения на Pascal

```
var
  balls :array[1..500000] of integer;
  boxes :array[1..500000] of int64;
  n,m,j,i, answer : integer;
  a,b,c : int64;
begin
  readln(n,m);
  for i:=1 to n do
    read(balls[i]);
  for i:=1 to m do
    begin
      readln(a,b,c);
      boxes[i]:=min(a,min(b,c));
    end;

  for i:=n-1 downto 1 do
    for j:=1 to i do
      if (balls[j]>balls[j+1]) then swap(balls[j+1],balls[j]);
  for i:=m-1 downto 1 do
    for j:=1 to i do
      if (boxes[j]>boxes[j+1]) then swap(boxes[j+1],boxes[j]);

  answer:=n;
  i:=1;j:=1;
  while (i<=n) and (j<=m) do
    begin
      if (2*balls[i]<=boxes[j]) then
        begin
          i:=i+1;
          answer:=answer-1;
        end;
      j:=j+1;
    end;
  writeln(answer);
end.
```

Задача 3

Пирамидка из кругов

ограничение по времени на тест: *2 секунды*
ограничение по памяти на тест: *256 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубик складывает пирамидку из кругов. При построении пирамидки Кубик последовательно кладет один круг на другой большего радиуса. Все круги лежат на длинном прилавке в ряд. Кубик последовательно берет круги ряда и строит пирамидку. Если очередной круг не может быть размещен на текущей пирамидке, то Кубик начинает строить новую пирамидку. Определите сколько пирамидок, в которых больше h кругов, сможет построить Кубику.

Входные данные:
В первой строке заданы два целых числа n ($1 \le n \le 10^{10}$) и h ($1 \le h \le 10^5$) — количество кругов на прилавке и высота пирамидки в кругах соответственно. Во второй строке каждого набора заданы n целых чисел $d_1, d_2, \ldots, d_n$ — диаметры кругов ($d_i < 10^5$, $i = 1..n$).

Выходные данные:
Выведите искомое количество, являющегося решением задачи.

Пример
входные данные
11 3
5 6 4 3 2 2 1 8 5 4 2
выходные данные
2
Примечание
Кубик построит четыре пирамидки: (5), (6, 4, 3,2), (2, 1), (8, 5, 4, 2). Только две из них состоят из более чем трех кругов

РЕШЕНИЕ
Для решения задачи достаточно сравнить диаметр текущего круга и предыдущего. Если высота достигла h, то увеличивать количество пирамидок.

Пример решения на Pascal

```
var
  a,b,
  currentH,answer,h, n,i : int64;
begin
  readln(n,h);
  read(a);
  currentH:=1;
  answer:=0;
  if h=currentH then answer:=answer+1;
  for i:=2 to n do
    begin
      read(b);
```

```
      if b<a then
        currentH:=currentH+1
      else
        currentH:=1;
      if h=currentH then answer:=answer+1;
      a:=b;
    end;
    writeln(answer);
end.
```

Задача 4

Оптимизация застройки

ограничение по времени на тест: *2 секунды*
ограничение по памяти на тест: *256 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубик, являясь мэром города Кубиктаун, издал указ об оптимизации высоты домов. По мнению мэра высота домов оптимальна если максимальная разность между этажами домов одного квартала не превышает этажности самого низкого дома этого же квартала. Для проведения оптимизации этажности города можно как достроить дополнительные этажи к дому, так и снести уже имеющиеся. Стоимость достройки одного этажа n кублей (кубль – денежная единица в Кубиктауне), а снос одного этажа – m кублей. Кубик, как рачительный мэр, желает уменьшить затраты по оптимизации города. Помогите Кубику найти наименьшую сумму затрат на проведение оптимизации.

Входные данные:
В первой строке заданы целые числа n ($1 \le n \le 1000$) — стоимость достройки одного этажа, m ($1 \le m \le 100$) — стоимость сноса одного этажа, t ($1 \le t \le 10^5$) — количество кварталов в городе.
Далее следуют t строк — описания кварталов города. Первое число k ($1 \le k \le 10^5$) – количество домов квартала, а затем числа $h_1, h_2, \ldots h_k$ ($1 \le h_i \le 100$)– высота домов квартала

Выходные данные:
Значение минимальной суммы затрат на оптимизацию.

Пример
входные данные
2 8 3
5 2 2 6 2 2
3 1 2 4
6 3 3 3 3 7 3
выходные данные
18

Примечание
В первом квартале 5 домов, необходимо достроить двухэтажным домам по одному этажу (стоимость 8 кублей)
Во втором квартале 3 дома, необходимо достроить один этаж к одноэтажному дому (стоимость 2 кублей)
В третьем квартале 6 домов, необходимо снести один этаж у семиэтажного дома (стоимость 8 кублей)

РЕШЕНИЕ

Для решения необходимо для каждого квартала произвести итерационный подсчет. На каждой итерации необходимо сравнивать суммарную стоимость поднятия на один этаж всех домов с минимальной высотой и стоимости уменьшения на один этаж всех домов с максимальной высотой. При этом необходимо учитывать, что снос уменьшает разность на 1, а достройка - на 2. После выполнения итерации необходимо изменить количество соответствующих домов квартала

Пример решения на Pascal

```
var
  answer :int64;
  k,t,n,m,i,j,hi, hmax, hmin : integer;
  h : array[1..100] of integer;
begin
  readln(n,m,t);
  answer:=0;
  for i:=1 to t do
    begin
      read(k);
      for j:=1 to 100 do h[j]:=0;
      hmax:=-1;
      hmin:=101;
      for j:=1 to k do
        begin
          read(hi);
          hmax:=max(hmax, hi);
          hmin:=min(hmin, hi);
          h[hi]:=h[hi]+1;
        end;
      while (2*hmin<hmax) do
        begin
          if (hmax-2*hmin=1) then
            begin
              if (n*h[hmin]<=m*h[hmax]) then
                begin
                  answer:=answer+n*h[hmin];
                  h[hmin+1]:=h[hmin+1]+h[hmin];
                  hmin:=hmin+1;
                end
              else
                begin
                  answer:=answer+m*h[hmax];
                  h[hmax-1]:=h[hmax-1]+h[hmax];
                  hmax:=hmax-1;
                end;
            end
          else
            begin
              if (n*h[hmin]<=m*(2*h[hmax]+h[hmax-1])) then
                begin
                  answer:=answer+n*h[hmin];
                  h[hmin+1]:=h[hmin+1]+h[hmin];
                  hmin:=hmin+1;
                end
              else
                begin
                  answer:=answer+m*(2*h[hmax]+h[hmax-1]);
                  h[hmax-2]:=h[hmax-1]+h[hmax];
                  hmax:=hmax-2;
                end;
            end;
        end;
    end;
```

```
    writeln(answer);
end.
```

Задача 5

Простые пирамидки

ограничение по времени на тест: *1 секунда*
ограничение по памяти на тест: *256 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубичка (подруга Кубика) смотрела как Кубик складывает пирамидки из кругов (задача №3). Кубичка любит простые числа, поэтому она считает пирамидку красивой, если сумма радиусов кругов, его образующих, является простым числом. Определите сколько красивых, по мнению Кубички, пирамидок построил Кубику.

Входные данные:
В первой строке задано целое число n ($1 \le n \le 10^{10}$) — количество пирамидок. Далее следуют n строк — описания пирамидок. Первое число k ($1 \le k \le 100$) – количество кругов в пирамидке, а затем числа $r_1, r_2, \ldots r_k$ ($1 \le r_i \le 1000$)– радиусы кругов, составляющих пирамидку

Выходные данные:
Выведите искомое количество, являющегося решением задачи.

Пример
входные данные
3
3 1 2 3
4 1 2 3 5
5 2 3 4 5 9
выходные данные
2
Примечание
Красивыми являются вторая (1+2+3+5=11 – простое число) и третья пирамидки (2+3+4+5+9=23 – простое число)

РЕШЕНИЕ
Необходимо проверить на простоту сумму радиусов.

Пример решения на Pascal

```
var
  n,i,answer :int64;
  k,r,sum,j : integer;
function isSimple(a: integer):boolean;
var
  j,k : integer;
begin
  result:=true;
  if a=1 then result:=false
  else
    begin
      k:=round(sqrt(a));
      for j:=2 to k do
        if a mod j=0 then
```

```
        begin
          result:=false;
          break;
        end;
    end;
end;

begin
  readln(n);
  answer:=0;
  for i:=1 to n do
    begin
      read(k);
      sum:=0;
      for j:=1 to k do
        begin
          read(r);
          sum:=sum+r;
        end;
      if isSimple(sum) then answer:=answer+1;
    end;
  writeln(answer);
end.
```

Задача 6

Снять блокаду!

ограничение по времени на тест: *3 секунд*
ограничение по памяти на тест: *256 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубики живут в городах. Города нумеруются последовательным рядом натуральных чисел начиная с 1. Некоторые пары городов связаны дорогами. Считается, что два города связаны сетью дорог, если из одного города можно попасть в другой двигаясь по дорогам, возможно проезжая через другие города. Случилась беда, и на некоторых дорогах поселились шарики, которые не пропускают кубиков. Известно, что на одной дороге обитает не более одного шарика. Чтобы освободить проезд необходимо убить шарика. Кубики народ мирный и любят природу, поэтому они стараются не убивать шариков без необходимости. Но для выживания всех кубиков необходимо, чтобы между любой парой городов существовала возможность проезда. Помогите кубикам минимизировать ущерб популяции шариков при условии выживания кубиков. Гарантируется, что изначально в любой город можно было попасть из любого города.

Входные данные:

Первая строка содержит три целых числа n, m, u ($0 \le n \le 2*10^4$; $1 \le m \le 3*10^4$; $1 \le u \le 10^6$) – количество городов, количество шариков и количество дорог.

В следующих u строках записаны пара чисел – номера городов, непосредственно соединенных дорогой.

В следующих m строках записаны пара чисел – номера городов, соединенных прямой дорогой, на которой поселился шарик.

Выходные данные

Найдите минимальное количество шариков, которое придётся уничтожить для выживания кубиков.

Примеры

входные данные

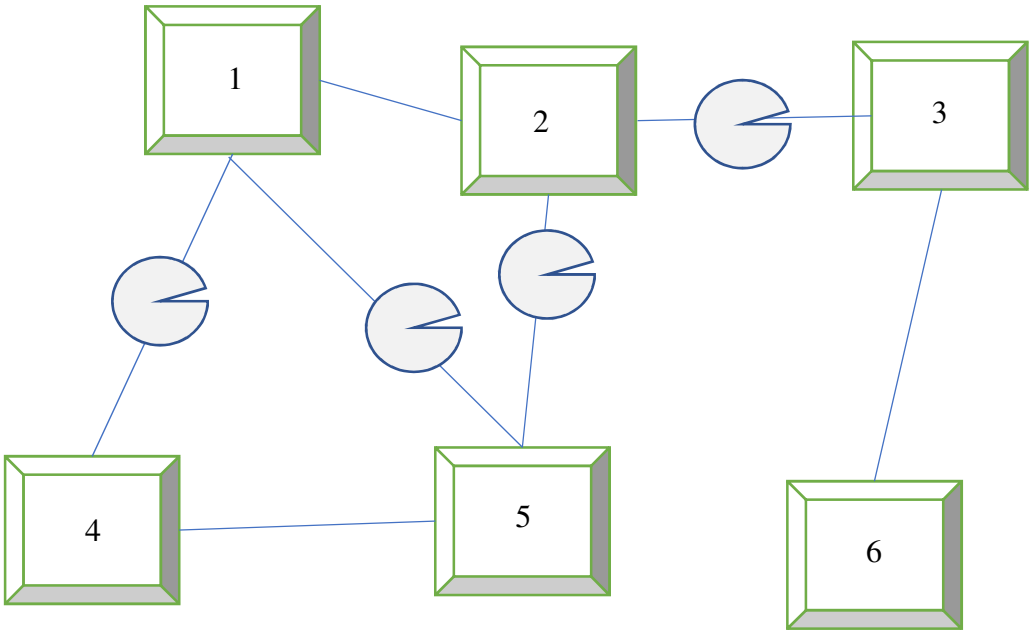
```
6 4 7
1 2
1 4
1 5
2 3
2 5
3 6
4 5
1 4
1 5
2 5
2 3
```

выходные данные

2

Примечание

Для данного примера кубикам достаточно уничтожить шариков на дорогах между городами 2 и 5, а также между городами 2 и 3.



РЕШЕНИЕ

Так как изначально все города были связаны, то достаточно удалить из графа ребра, на которых поселились шарики. Ответом будет число компонент связанности полученного графа уменьшенное на 1.

Пример решения на C++

```
#include <iostream>
#include <vector>
using namespace std;

vector<vector<long>> graph;
vector<int> color;

long long n, m, u;

void del(int a, int b) {
	for (int i = 0; i < graph[a].size(); i++)
		if (graph[a][i] == b) {
			graph[a].erase(graph[a].begin() + i);
			break;
		}
}

void dfs(int a, int c) {
	color[a] = c;
	for (int i = 0; i < graph[a].size(); i++) {
		if (color[graph[a][i]] == 0) dfs(graph[a][i], c);
	}
```

```
}

int main() {

	cin >> n >> m >> u;
	graph.resize(n);
	color.resize(n);

	for (int i = 0; i < u; i++) {
		int a, b;
		cin >> a >> b;
		a--; b--;
		graph[a].push_back(b);
		graph[b].push_back(a);
	}
	for (int i = 0; i < m; i++) {
		int a, b;
		cin >> a >> b;
		a--; b--;
		del(a, b);
		del(b, a);
	}
	int c = 0;
	for (int i = 0; i < n; ++i) {
		if (color[i] == 0) dfs(i, ++c);
	}
	cout << c-1;
	return 0;
}
```