Задание 1

Счастливое число Кубика

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

По мнению Кубика, число считается счастливым, если оно имеет ровно три делителя (Например, число *289* – счастливое, так оно делится на 1, 17 и 289). Кубик хочет узнать является ли имеющееся у него число счастливым. Помогите Кубику решить эту задачу.

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

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

Пример
входные данные
289
выходные данные
YES

РЕШЕНИЕ
Необходимо проверить является ли число квадратом простого числа
Пример решения на PascalABC

```
function simple(n :int64):boolean;
var
  i: int64;
begin
  result:=true;
  if n=1 then
  begin
    result:=false;
    exit;
  end;
  for i:=2 to round(sqrt(n)) do
    if n mod i=0 then
    begin
      result:=false;
      exit;
    end;
end;
var
  a,n : int64;
begin
  readln(n);
  a:=round(sqrt(n));
  if (a*a=n) and simple(a) then
    writeln('YES')
  else
```

```
    writeln('NO');
end.
```

Задание 2
Отправка

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

Кубику необходимо распределить n грузов по контейнерам. Каждый груз характеризуется весом w, а каждый контейнер — грузоподъемностью g. Кубик размещает не более чем по два груза в один контейнер там, чтобы их суммарный вес не превышал q. Помогите Кубику определить минимальное количество контейнеров, которые необходимо для распределения всех грузов.

Входные данные:
В первой строке два целых числа n, q ($1 \le n \le 5*10^5$, $1 \le q \le 5*10^5$) — количество грузов и грузоподъемность контейнеров соответственно.
Во второй строке заданы n целых чисел w_1, w_2, …, w_n — веса грузов ($0 < w_i \le q$, $i = 1..n$).

Выходные данные:
Выведите число – ответ на задачу.

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

Примечание
Возможно следующее решение: в первый контейнер необходимо поместить грузы с весами 1 и 4, во второй — 2 и 3, в третий — 5.

РЕШЕНИЕ
Для решения задачи необходимо построить отсортированный массив весов грузов. Затем последовательно ввести два указателя для минимального и максимального весов, помещаемых в контейнер. Если оба груза возможно поместить в контейнер, то оба указателя изменяются, иначе изменяется только указатель тяжелого груза. В обоих случаях счетчик контейнеров увеличивается на 1.

Пример решения на PascalABC

```
var
  i,j, answer,q, n : integer;
  w : array of integer;
begin
  readln(n,q);
  w := new integer[n];
  for i:=0 to n-1 do
    read(w[i]);
```

```
  sort(w);
  answer:=0;
  i:=0;
  j:=n-1;
  while i<=j do
  begin
    if w[i]+w[j]<=q then
      inc(i);
    dec(j);
    inc(answer);
  end;
  writeln(answer);
end.
```

Задание 3

Самая высокая пирамидка

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, радиус которого ровно на единицу больше. 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 количество кругов в самой высокой пирамидке, которую сможет построить Кубику.

Входные данные:
В первой строке задано целое число n ($1 \le n \le 10^{10}$) — количество кругов на прилавке.
Во второй строке заданы n целых чисел $r_1, r_2, \ldots, r_n$ — радиусы кругов ($0 < r_i < 10^5$, $i = 1..n$).

Выходные данные:
Выведите искомое количество, являющееся решением задачи.

Пример
входные данные
11
5 6 4 3 2 2 1 8 5 4 2
выходные данные
3
Примечание
Максимальная пирамидка, которую сможет построить Кубик: (4, 3, 2).

РЕШЕНИЕ
Для решения задачи достаточно сравнить диаметр текущего круга и предыдущего.

Пример решения на PascalABC

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

```
      else
        current:=1;
      answer:=max(answer,current);
      a:=b;
    end;
    writeln(answer);
end.
```

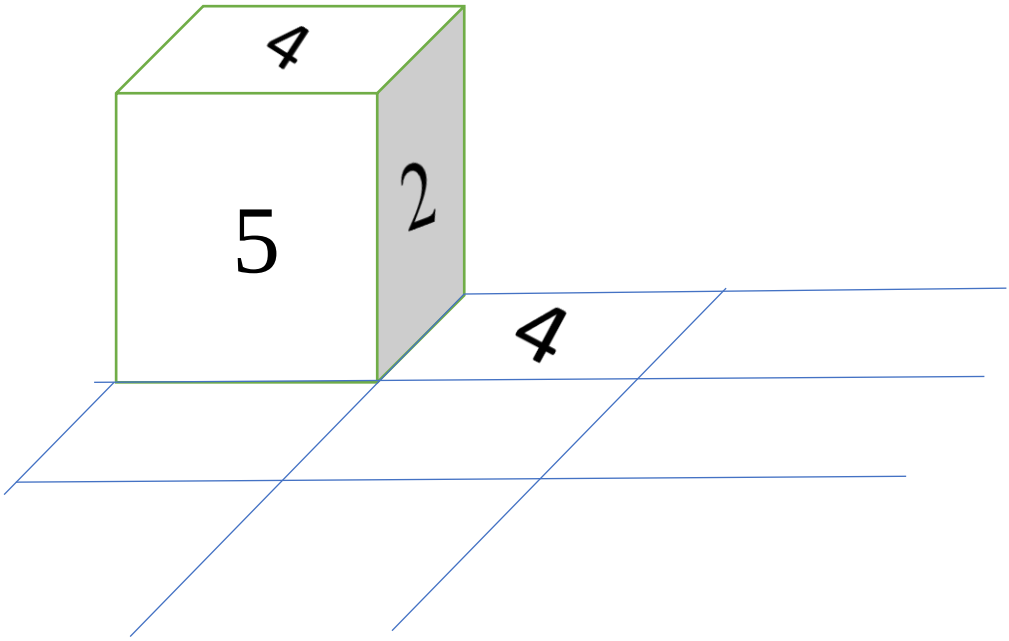
Задача 4

Собрать число

ограничение по времени на тест: 10 *секунд*
ограничение по памяти на тест: *2048 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Внешне Кубик выглядит как трехмерный куб. Кубик на своих гранях имеет числа. Кубик находится на бесконечной прямоугольной площадке, разделенной на квадраты. Каждый квадрат площадки размерами совпадает с боковой гранью Кубика. Перемещаться Кубик может только с нижней грани на одну из боковых, перекатываясь через их общее ребро. Если Кубик оказался на клетке площадки, то в этой клетке записывается число равное числу нижней грани Кубика. Кубик может перекатиться на соседнюю клетку если в ней не записано число.

Например, в ниже приведенной ситуации Кубик может перекатиться на грань со значением 5, а на грань со значением 2 – не может.



Определите максимальную сумму чисел (включая исходную), записанных в клетках после не более *n* перемещений Кубика.

Входные данные:
Первая строка содержит 6 чисел через пробел – значения на нижней, верхней, правой, левой, передней и задней гранях Кубика.
Вторая строка содержит целое число *n* (*0< n <10*) – количество перемещений.

Выходные данные:

Выведите искомое число, являющееся решением задачи.

Пример
входные данные
1 2 3 4 5 6
3

выходные данные
16

Примечание
Кубику необходимо перекатиться вперед и два раза налево.

РЕШЕНИЕ
Для решения задачи достаточно перебрать все возможные движения кубика.

Пример решения на PascalABC

```
const
 areaSize=9;
type
 areaType = array[-areaSize..areaSize,-areaSize..areaSize] of integer;

function MoveFront (a:array of integer):array of integer;
begin
  result:=new integer[6];
  result[0]:=a[4];
  result[5]:=a[0];
  result[4]:=a[1];
  result[1]:=a[5];
  result[2]:=a[2];
  result[3]:=a[3];
end;
function MoveBack (a:array of integer):array of integer;
begin
  result:=new integer[6];
  result[0]:=a[5];
  result[5]:=a[1];
  result[1]:=a[4];
  result[4]:=a[0];
  result[2]:=a[2];
  result[3]:=a[3];
end;
function MoveLeft(a:array of integer):array of integer;
begin

  result:=new integer[6];
  result[4]:=a[0];
  result[0]:=a[3];
  result[3]:=a[1];
  result[1]:=a[2];
  result[4]:=a[4];
  result[5]:=a[5];
end;
function MoveRight(a:array of integer):array of integer;
begin
  result:=new integer[6];
```

```
  result[3]:=a[0];
  result[0]:=a[2];
  result[2]:=a[1];
  result[1]:=a[3];
  result[4]:=a[4];
  result[5]:=a[5];
end;
function SumArea(area:AreaType):integer;
begin
 result:=0;
 for var i:=-areaSize to areaSize do
  for var j:=-areaSize to areaSize do
    result:=result+area[i,j];
end;
procedure CopyArea(var FromArea, ToArea:AreaType);
begin
 for var i:=-areaSize to areaSize do
  for var j:=-areaSize to areaSize do
    ToArea[i,j]:=FromArea[i,j];
end;

function FillArea(k,n,y,x : integer; cube:array of integer; area:AreaType):integer;
begin
  result:=0;
  if k=n then
    begin
     result:=SumArea(area);
     exit;
    end;
  if area[y+1,x]=0 then
  begin
    var NewArea : AreaType;
    CopyArea(area, NewArea);
    var NewCube:=MoveFront(cube);
    NewArea[y+1,x]:=NewCube[0];
    result:=max(result, FillArea(k+1,n,y+1,x,NewCube,NewArea))
  end;
  if area[y-1,x]=0 then
  begin
    var NewArea :AreaType;
    CopyArea(area,NewArea);
    var NewCube:=MoveBack(cube);
    NewArea[y-1,x]:=NewCube[0];
    result:=max(result, FillArea(k+1,n,y-1,x,NewCube,NewArea))
  end;
  if area[y,x+1]=0 then
  begin
    var NewArea :AreaType;
    CopyArea(area,NewArea);
    var NewCube:=MoveRight(cube);
    NewArea[y,x+1]:=NewCube[0];
    result:=max(result, FillArea(k+1,n,y,x+1,NewCube,NewArea))
  end;
  if area[y,x-1]=0 then
  begin
    var NewArea :AreaType;
    CopyArea(area,NewArea);
    var NewCube:=MoveLeft(cube);
    NewArea[y,x-1]:=NewCube[0];
    result:=max(result, FillArea(k+1,n,y,x-1,NewCube,NewArea))
  end;
end;
```

```
var
  n :integer;
  cube:array of integer;
  area:AreaType;
begin
 cube := new integer[6];
 readln(cube[0],cube[1],cube[2],cube[3],cube[4],cube[5]);
 readln(n);
 for var i:=-areaSize to areaSize do
  for var j:=-areaSize to areaSize do
    area[i,j]:=0;
 area[0,0]:=cube[0];
 writeln(FillArea(0,n,0,0,cube,area));
end.
```

Задача 5

Красивые пирамидки

ограничение по времени на тест: *1 секунда*
ограничение по памяти на тест: *2 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубичка (подруга Кубика) смотрела как Кубик складывает пирамидки из кругов (задача №3). Кубичка любит совершенные числа, поэтому она считает пирамидку красивой, если в ее составе есть хотя бы один круг, радиус которого является совершенным числом (Число является совершенным, если оно равно сумме всех своих делителей, за исключением самого числа. Например, 6 — совершенное число 1+2+3). Определите является ли красивой, по мнению Кубички, пирамидка, построенная Кубиком.

Входные данные:
В строке заданы целые числа n, r: n ($1 \leq n, r \leq 10^{10}$) – количество кругов в пирамидке, r – радиус верхнего круга пирамидки

Выходные данные:
Выведите YES если пирамидка красивая и NO в противном случае.

Пример
входные данные
3 5

выходные данные
YES

Примечание
Пирамидка состоит из кругов радиуса 7, 6, 5.

РЕШЕНИЕ
Необходимо проверить является ли хотя бы один радиусов совершенным числом.

Пример решения на PascalABC

```
function perfect(n: int64): boolean;
var
  i,sum: int64;
begin
  sum:=0;
  for i:=1 to n div 2 do
    if n mod i=0 then
     sum:=sum+i;
  result:=sum=n;
end;

var
  i,n,r : int64;
begin
```

```
  readln(n,r);
  for i:=0 to n-1 do
    begin
     r:=r+1;
     if perfect(r) then
     begin
       writeln('YES');
       exit;
     end;
    end;
  writeln('NO');
end.
```

Задача 6

Захватить острова!

ограничение по времени на тест: *5 секунд*
ограничение по памяти на тест: *512 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

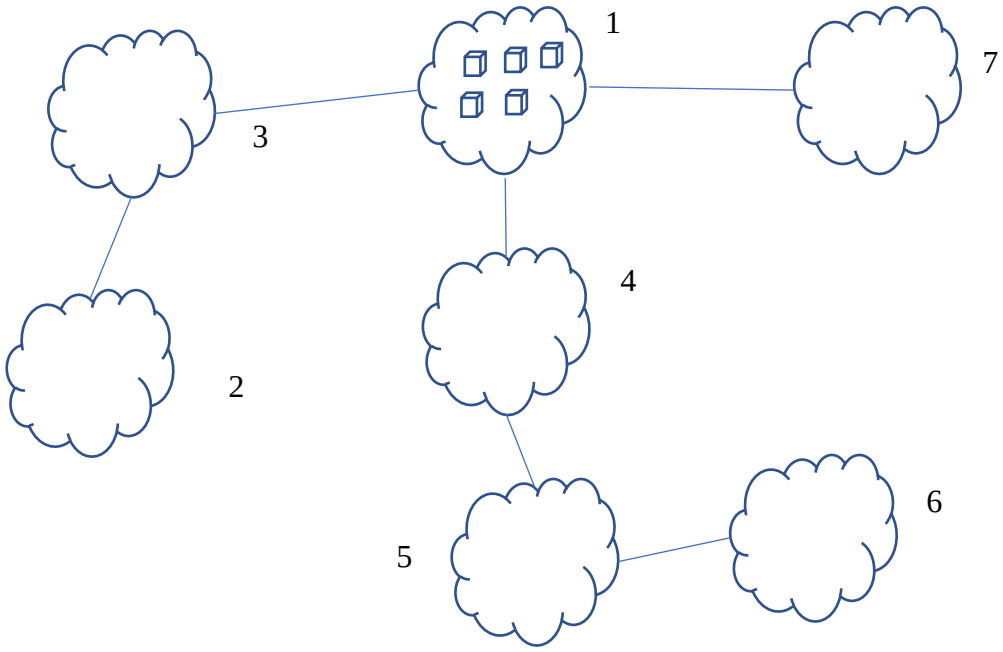
Группа кубиков живет на острове, расположенном на болоте. Некоторые пары островов связаны мостами (одна пара не более чем одним мостом). Кубики решили расширить свой ареал обитания и захватить как можно больше островов. Остров считается захваченным, если на нем находится хотя бы два кубика. Помогите кубикам определить наибольшее количество островов, которые они могут захватить.

Входные данные:
Первая строка содержит три целых числа *n*, *m*, *k* ($0 \le n \le 2*10^4$; $1 \le m,k \le 3*10^4$) – количество кубиков, количество островов и количество мостов.
Считается, что кубики изначально обитают на острове с номером *1*.
В следующих *k* строках записаны пара чисел – номера островов, соединенных мостом.

Выходные данные
Выведите целое число — найденное количество.

Примеры
входные данные
```
5 7 6
1 3
1 4
1 7
2 3
4 5
5 6
```
выходные данные
```
2
```

Примечание
Для данного примера кубикам могут принадлежать только два острова.



РЕШЕНИЕ

Необходимо найти количество вершин графа, достижимых из первой вершины.

Пример решения на C++

```
#include <iostream>
#include <vector>
using namespace std;
struct vertex {
	int visited;
	vector<int> neighbors;
};

vector <vertex> graph;

void dfs(int root) {
	graph[root].visited = 1;
	for (int i = 0; i < graph[root].neighbors.size(); ++i) {
		if (graph[graph[root].neighbors[i]].visited == 0) {
			dfs(graph[root].neighbors[i]);
		}
	}
}
int main() {
	int n, m, k;
	cin >> n >> m >> k;
	graph.resize(m);
	for (int i = 0; i < k; i++) {
		int a, b;
		cin >> a >> b;
		graph[a - 1].neighbors.push_back(b - 1);
		graph[b - 1].neighbors.push_back(a - 1);
	}
	for (int i = 0; i < graph.size(); i++)
		graph[i].visited = 0;
	dfs(0);
```

```
	int counter = 0;
	for (int i = 0; i < graph.size(); i++)
		counter+=graph[i].visited;
	cout << min(counter, n / 2);
	return 0;
}
```