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 представимо в виде произведения ровно трех различных простых чисел (Например, число *255* – счастливое, так оно делится на 3, 5 и 17). 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
255
выходные данные
YES

РЕШЕНИЕ на языке Python:

```
def simple(n):
    if n==1:
       return False
    for i in range (2,int(n**0.5)+1):
       if n%i==0:
           return False
    return True
def f2(n,k):
    n=n//k
    for i in range (k+1, int(n**0.5)+1):
        if n%i==0 and n//i!=i and n//i>k and simple(i) and simple(n//i):
            return True
    return False
def f1(n):
    for i in range (2, int(n**0.5)+1):
        if n%i==0 and simple(i) and f2(n,i):
            return True
    return False
n=int(input())
if f1(n):
    print('YES')
else:
    print('NO')
```

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 q. Кубик размещает не более чем по три 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 8
1 2 3 4 5
выходные данные
2

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

РЕШЕНИЕ на языке Python:
Для решения задачи необходимо построить отсортированный массив весов грузов.

```
n,q=map(int,input().split())
w=[int(x) for x in input().split()]
w.sort()
k=0
while len(w)>0:
    k+=1
    if w[0]+w[-1]>q:
        w=w[:-1]
    else:
        d=0
        for i in range(1,len(w)-1):
            if w[0]+w[i]+w[-1]<=q:
                d=i
```

```
            else:
                break
        if d>0:
            w=w[1:d]+w[d+1:-1]
        else:
            w = w[1:-1]
print(k)
```

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 два 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 может начать строить новую пирамидку или отбросить этот круг. 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
8 5 6 4 3 2 2 1 8 5 2
выходные данные
4
Примечание
Максимальная пирамидка, которую сможет построить Кубик: (8, 6, 4, 2).

РЕШЕНИЕ на языке Python:

```
n=int(input())
r=[int(x) for x in input().split()]
k={}
for i in range(len(r)):
    if k.get(r[i]+2)==None:
        if k.get(r[i])==None:
            k[r[i]]=1
    else:
        if k.get(r[i]) == None:
            k[r[i]] = k[r[i]+2]+1
        else:
            k[r[i]] = max(k[r[i]],k[r[i] + 2] + 1)
m=0
for i in k.keys():
```

```
    m=max(m,k[i])
print(m)
```

Задача 4

Поесть блины

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

Кубик любит кушать блины. У Кубика есть n блинов. Каждый блин имеет свой вес. Кубик может скушать в i-й день только один блин, вес которого не менее чем i. Определите максимальное количество дней, в течении которых Кубик может кушать блины.

Входные данные:
Первая строка содержит одно целое число n ($0<n\leq 2*10^5$) – количество блинов.
Вторая строка содержит n целых чисел $a_1,a_2,\ldots,a_n$ ($1\leq a_i\leq 10^9$), где a_i — это вес i-го блина.

Выходные данные:
Выведите одно число — максимальное количество дней когда Кубик может кушать блины.

Пример
входные данные
4
3 1 4 1
выходные данные
3

РЕШЕНИЕ на языке Python:

```
n=int(input())
a=[int(x) for x in input().split()]
a.sort()
k=0
for i in range(len(a)):
    if a[i]>k:
        k+=1
print(k)
```

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 простые числа, поэтому она считает пирамидку красивой, если в ее составе есть хотя бы два круга, радиус которых являются простым числом. 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 9, 7, 5.

РЕШЕНИЕ на языке Python:

```
def simple(n):
    if n==1:
       return False
    for i in range (2,int(n**0.5)+1):
       if n%i==0:
           return False
    return True
n,r=map(int,input().split())
k=0
for i in range(n):
    if simple(r+i*2):
        k+=1
if k>1:
    print('YES')
else:
    print('NO')
```

Задача 6

Великое расселение!

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

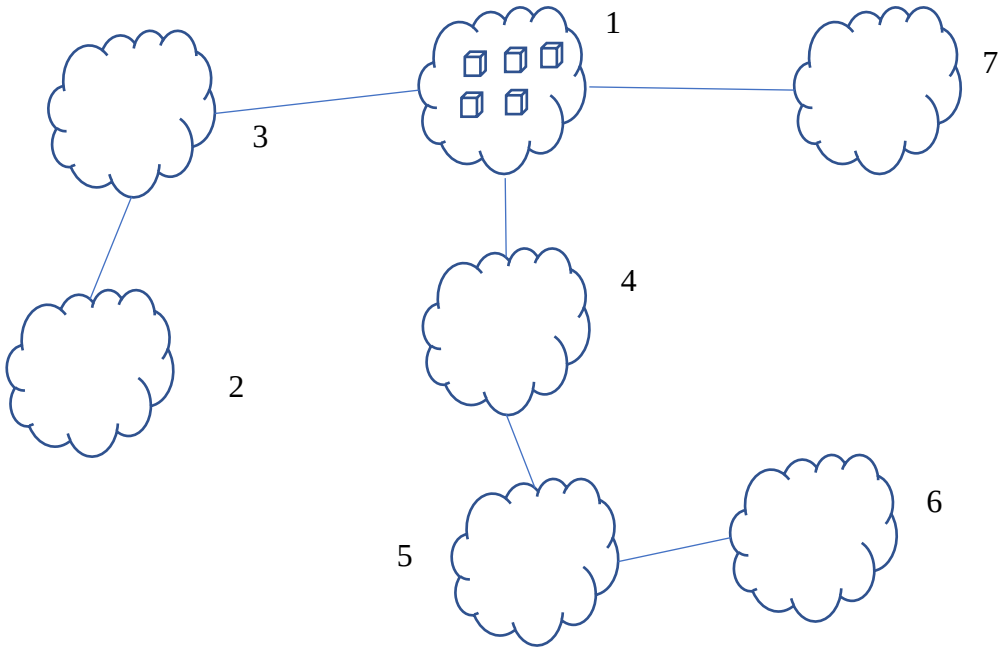
Группа кубиков живет на острове, расположенном на в дельте реки. Некоторые пары островов связаны мостами (одна пара не более чем одним мостом). Кубики решили расширить свой ареал обитания и расселиться по как можно больше островам. Остров считается заселенных, если на нем находится хотя бы четыре кубика. Помогите кубикам определить наибольшее количество островов, которые они могут заселить.

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
1

Примечание
Для данного примера кубикам могут принадлежать только один остров.



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 / 4);
        return 0;
}
```