Задание №1
Красота Кубика

ограничение по времени на тест: 15 *секунд*
ограничение по памяти на тест: 32 *мегабайт*

ввод: *стандартный ввод*
вывод: *стандартный вывод*

Внешне Кубик выглядит как трехмерный куб.

Кубик на каждой из его граней написано число. Красота Кубика равна наибольшему простому числу, написанному на одной из его граней, при условии, что оно равно сумме чисел не менее двух ей смежных граней. Если граней с таким условием нет, то сила Кубика, считается равной разности максимального и минимального чисел, написанных на его гранях. У Кубика имеется *N* различных натуральных чисел, которые он может написать на своих гранях. Определите максимальную красоту, которую может обрести Кубик.

Входные данные:
Первая строка содержит целое число *N* (*5< N <200*) – количество чисел последовательности.
Следующая строка содержит *N* натуральных чисел a_1, a_2, …, a_n ($1 \le a_i \le 10^{10}$).

Выходные данные:
Выведите искомое число – максимально возможную силу Кубика.

Пример
входные данные
7
1 2 3 4 5 6 7

выходные данные
7

Пример
входные данные
8
2 8 9 20 21 45 40 100

выходные данные
98

Примечание
В первом примере у Кубика может быть грань с числом 7, смежные грани которой 3 и 4

Во втором примере у Кубика могут быть грани с числами 100 и 2

Решение на языке C++

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

static bool simple(long long n)
{
   long long s = round(sqrt(n));
   if (s * s == n) return false;
   for (long long i = 2; i <= s; i++)
      if (n % i == 0) return false;
   return true;
}

int main()
{
   long long n;
   cin >> n;
   long long *v = new long long[n];
   for (long long i = 0; i < n; i++) {
      cin >> v[i]
   }
   sort(v[n-1], v.rend());
   long long m = v[0] - v[v.size() - 1];
   for (long long i = 0; i < v.size(); i++) {
      if (v[i] <= m) break;
      if (simple(v[i])) {
         bool p = false;
         for (long long i1 = i + 1; (i1 < v.size()) && !p; i1++) {
            for (long long i2 = i1 + 1; (i2 < v.size()) && !p; i2++) {
               p= (v[i] == (v[i1] + v[i2]));
               for (long long i3 = i2 + 1; (i3 < v.size()) && !p; i3++) {
                  p = (v[i] == (v[i1] + v[i2] + v[i3]));
                  for (long long i4 = i3 + 1; (i4 < v.size()) && !p; i4++) {
                     p = (v[i] == (v[i1] + v[i2] + v[i3]+v[i4]));
                  }
               }
            }
         }
         if (p) m = v[i];
      }
```

```
    }
    cout << m;
    return 0;
}
```

Задание 2
Собирая шарики

ограничение по времени на тест
10 секунд
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод

Кубик собирает шарики в кувшин. Кувшин может вместить все шарики, которые найдет Кубик, но не все шарики пролезают в горлышко кувшина. Каждый шарик характеризуется двумя значениями (d, m), где d – диаметр шарика, а m – вес шарика. Кубик собирает только шарики одного диаметра. В процессе прогулки Кубик нашел N шариков.

Определить максимальный вес шариков, которые Кубик может собрать в кувшин, горлышко которого имеет диаметр D (если диаметр шарика равен D, то он может быть помещен к кувшин).

Входные данные:

Первая строка содержит два целых числа D, N ($0<D<10^5$, $1<N<10^4$) – диаметр шарика и количество найденных шариков.

Следующие N строк содержат пару целых чисел (d_1, m_1), (d_2, m_2), …, (d_N, m_N) ($1 \leq d_i, m_i \leq 10^5$), где d_i, m_i — это диаметр и вес i-го шарика.

Выходные данные:

Выведите максимальный суммарный вес шариков, которые поместятся в кувшин.

Пример

входные данные

10 5
2 5
8 4
2 6
31 81
9 7

выходные данные

11

Решение на языке C++

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

int main() {
	long D, N;
	cin >> D >> N;
	long long *jug = new long long[D+1];
	for (long i = 0; i <= D; ++i)
		jug[i] = 0;
	for (long i = 0; i < N; ++i) {
		long long d, m;
		cin >> d >> m;
		if (d <= D) {
			jug[d] = jug[d] + m;
		}
	}
	long long  answer = 0;
	for (long i = 1; i <= D; ++i)
		answer = max(jug[i], answer);
	cout << answer;
	return 0;
}
```

Задание 3

Доставить депешу

ограничение по времени на тест: *10 секунд*
ограничение по памяти на тест: *640 мегабайта*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

В стране Кубикляндия города соединены дорогами и сетью телепортов. Дороги построены так, что из любого города можно попасть на любой другой города страны, двигаясь только по дорогам. А телепорты связывают только некоторые города. Передвижение по дорогам бесплатно, но требует времени. Перемещение между городами с помощью телепорта мгновенно, но требует денег.

Король послал курьера с депешей к своему генералу. Для ускорения доставки курьер получил сумму денег, достаточную только на одно перемещение. Курьер может двигаться по дороге и, возможно, один раз воспользоваться телепортом.

Помогите курьеру доставить депешу как можно скорее.

Входные данные:

Первая строка содержит два целых числа *n*, *m* ($0<n\leq 2*10^3$; $1\leq m\leq 3*10^4$) – количество городов и количество дорог.
Все города перенумерованы числами от *1* до *n*.
Вторая строка содержит два целых числа *VK*, *VG* – город короля и генерала соответственно.
В третьей строке через пробел перечислены города, в которых есть телепорты.
В следующих *m* строках записаны три числа *v1*, *v2*, *t* ($0<v1,v2\leq n$, $1\leq t\leq 3*10^4$)– номера городов, соединенных дорогой, и время движения по дороге.

Выходные данные

Выведите целое число — минимальное время, за которое курьер может доставить депешу.

Примеры

входные данные

7 9

1 6

2 3 7

1 3 8

1 7 3

1 4 14

2 3 6

7 4 10

2 4 9

2 5 15

4 5 4

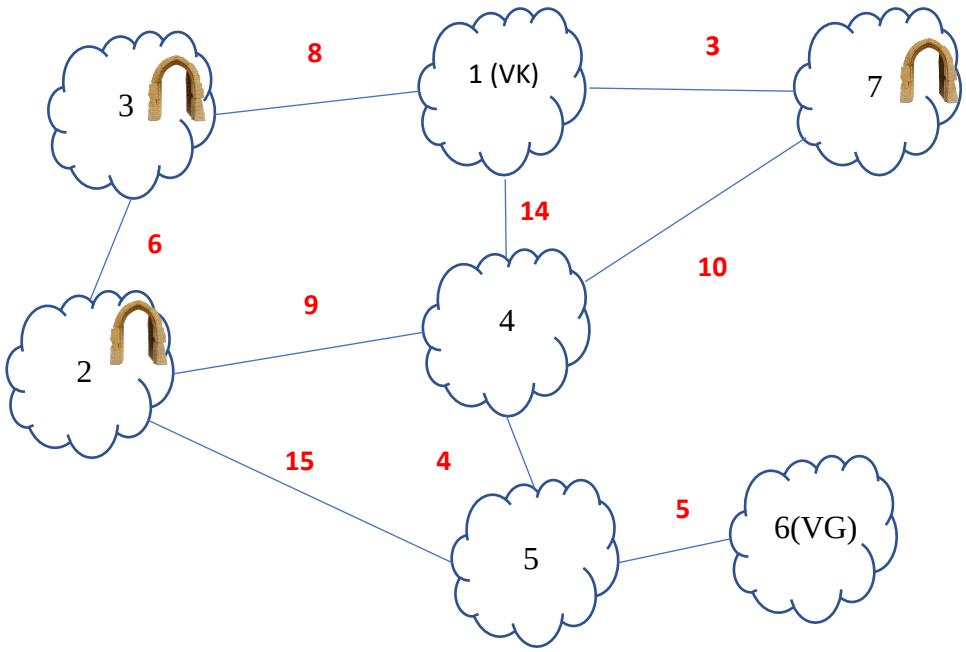
5 6 5

выходные данные

21

Примечание

Список городов, через которые пройдет курьер: 1,7, 2, 4, 5, 6



Решение на языке C++

```
#include <iostream>
#include <algorithm>
#include <string>

using namespace std;


long **d;
long answer;
short* teleport;
int n;

void FloydWarshall() {
    for (long k = 0; k < n; ++k)
        for (long i = 0; i < n; ++i)
            for (long j = 0; j < n; ++j)
                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
};

int main() {
    int m;
    cin >> n >> m;
    teleport = new short[n];
    d = new long* [n];
    for (long i = 0; i < n; ++i) {
        d[i] = new long[n];
        teleport[i] = 0;
        for (long j = 0; j < n; ++j)
            d[i][j] = 1000000000;
        d[i][i] = 0;
    }
    int VK, VG;
    cin >> VK >> VG;
    string s;
    getline(cin, s);
    getline(cin, s);
    s = s + ' ';
    int j = 0;
    for (int i = 0; i < s.length(); ++i) {
        if (s[i] != ' ') j = j * 10 + s[i] - '0';
        else {  teleport[j-1] = 1; j = 0;}

    }
    for (int i = 0; i < m; ++i){
```

```
        int v1, v2, t;
        cin >> v1 >> v2>>t;
        d[v1-1][v2-1] = d[v2-1][v1-1] = t;
    }
    FloydWarshall();
    answer = d[VK-1][VG-1];
    for (long i = 0; i < n; ++i) {
        if (teleport[i]) {
            for (long j = 0; j < n; ++j)
                if (teleport[j]) answer = min(answer, d[VK-1][i] + d[j][VG-1]);
        }
    }
    cout << answer;
    return 0;
}
```

Задание 4
Удачная серия
ограничение по времени на тест: *2 секунды*
ограничение по памяти на тест: *256 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубик передвигается прыжками. Кубик считает серию подряд выполненных прыжков удачной если любые два последовательных прыжка отличаются не более чем на M. Кубик записал все свои прыжки за несколько дней и захотел узнать количество прыжков в самой длинной удачной серии. Помогите Кубику найти это значение.

Входные данные:
В первой строке заданы два целых числа t, M ($1 \le t, M \le 100$) — количество дней и допустимое отличие между соседними прыжками удачной серии.
Далее следуют описания t наборов входных данных.
В первой строке каждого набора входных данных задано число n ($1 \le n \le 10^4$) — количество прыжков за день.
Во второй строке каждого набора заданы n целых чисел d_1, d_2, …, d_n — последовательность длин прыжков ($d_i < 10^4$, $i = 1..n$).

Выходные данные:
Найдите значение, являющееся решением задачи.

Пример
входные данные
2 2
5
3 1 20 4 2
8
1 2 4 3 10 2 3 6
выходные данные
4

Примечание
Самая удачная серия (1, 2, 4, 3) была выполнена во второй день.

Решение на языке C++
```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
```

```
	int  t, M;
	cin >> t >> M;
	int answer =0;
	for (int i = 0; i < t; ++i) {
		int n;
		cin >> n;
		int pred;
		cin >> pred;
		int l = 1;
		for (int j = 1; j < n; ++j) {
			int cur;
			cin >> cur;
			if (abs(cur - pred) <= M) ++l;
			else l = 1;
			pred = cur;
			answer = max(answer, l);
		}
	}
	cout << answer;
	return 0;
}
```

Задание 5

Самый высокий

ограничение по времени на тест

15 секунд

ограничение по памяти на тест

512 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

В очереди стоят N кубиков. Каждый кубик характеризуется высотой. Кубик хочет узнать некоторую информацию об очереди, с этой целью он задает вам вопросы. Каждый вопрос следующего вида:
Он называет вам два числа a и b ($1 \le a \le b \le n$), а вы должны сказать ему, чему равна высота вес самого высокого в очереди среди кубиков с номерами от a до b (нумерация от начала очереди).

Входные данные:

Первая строка содержит одно целое число n ($0<n<10^5$) – количество кубиков в очереди.
Вторая строка содержит n целых чисел $a_1,a_2,\ldots,a_n$ ($1 \le a_i \le 10^9$), где a_i — это высота i-го кубика.
В третьей строке записано целое число m ($1 \le m \le 10^5$) — количество вопросов Кубика. Затем следует m строк, в каждой строке записано два целых числа: a и b ($1 \le a \le b \le n$), описывающих текущий вопрос.
Вам нужно вывести ответ на вопрос.

Выходные данные

Выведите m строк. В каждой строке должно быть записано целое число — ответ на вопрос Кубика. Выводите ответы на вопросы в порядке их следования во входных данных.

Примеры

входные данные

```
6
6 4 2 7 2 9
2
3 4
1 6
```

выходные данные

```
7
9
```

Решение на языке C++

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

long long* a;
long long* tree;

//построение дерева отрезков
void build(long long v, long long l, long long r) {
	if (l == r)
		tree[v] = a[l];
	else {
		long long m = (l + r) / 2;
		build(v * 2, l, m);
		build(v * 2 + 1, m + 1, r);
		tree[v] = max(tree[v * 2], tree[v * 2 + 1]);
	}
}

// поиск по дереву отрезков
long long find(long long v, long long tl, long long tr, long long l, long long r) {
	if (l > r)
		return 0;
	if (l == tl && r == tr)
		return tree[v];
	long long tm = (tl + tr) / 2;
	return max(find(v * 2, tl, tm, l, min(r, tm)), find(v * 2 + 1, tm + 1, tr, max(l,
tm + 1), r));
}

int main() {
	long long n;
	cin >> n;
	a = new long long[n];
	tree = new long long[4 * n];
	for (long long i = 0; i < n; ++i)
		cin >> a[i];
	build(1, 0, n - 1);
	long long m;
	cin >> m;
	for (long long i = 0; i < m; ++i) {
		long long a, b;
		cin >> a >> b;
		cout << find(1, 0, n - 1, a - 1, b - 1) << endl;
```

```
        }
        return 0;
}
```

Задание №6
Особый Кубик

ограничение по времени на тест: *2 секунды*
ограничение по памяти на тест: *16 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Внешне Кубик выглядит как трехмерный куб.

На каждой из его граней Кубика написано натуральное число. Кубик считается особым, если сумма всех чисел на его гранях имеет ровно 7 делителей. Определите является ли Кубик особым.

Входные данные:

В первой строке задано целое число t ($1 \le t \le 100$) — количество наборов входных данных.

Следующие t строк содержат по *шесть* чисел a_1, a_2, a_3, a_4, a_5, a_6 ($0 < a_i < 10^9$)– значения на гранях Кубика.

Выходные данные:

Для каждого Кубика напишите “YES”, если он особый, и “NO” – в противном случае.

Пример

входные данные

```
3
1 2 3 5 6 7
10 21 3 5 11 12
6 8 10 2 4 6
```

выходные данные

```
NO
YES
NO
```

Решение на языке C++

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

bool f(long n) {
	long a = round(sqrt(n));
	if (a * a != n) return false;
	int k = 3;
```

```
        for (long i = 2; i < a; i++) {
                if (n % i == 0) k += 2;
                if (k > 7) return false;
        }
        return k == 7;
}

int main() {
        long long n;
    cin >> n;

        for (long long i = 0; i < n; ++i) {
                int a1, a2, a3, a4, a5, a6;
                cin >> a1>> a2>> a3>> a4>> a5>> a6;
                if (f(a1 + a2 + a3 + a4 + a5 + a6)) cout << "YES" << endl;
                else cout << "NO" << endl;
        }
        return 0;
}
```