

**Московский Авиационный Институт
(национальный исследовательский университет)**

**Институт информационных технологий и прикладной
математики**

Кафедра вычислительной математики и программирования

**Журнал по исследовательской практике (индивидуальный
план)**

Студенты	Группа
Тарпанов Даниил Александрович	М8О-307Б-19
Ежов Никита Павлович	М8О-307Б-19
Полюбин Арсений Игоревич	М8О-306Б-19
Команда:	МАИ #40 Ezhov, Polubin, Tarpanov

Москва, 2021

Сводная таблица за осень 2021

Дата	Название	Время	Место проведения	Решенные задачи	Дорешанные задачи
12.09.2021	Grand Prix of Dolgoprudny	11:00-16:00	Дистанционно	L, O	M
19.09.2021	Grand Prix of IMO	11:00-16:00	Дистанционно	M, O, P, Q	-
26.09.2021	Grand Prix of XiAn	11:00-16:00	Дистанционно	L, N, P	-
10.10.2021	XXII Открытая Всесибирская олимпиада	10:00-15:00	Дистанционно	A	-
24.10.2021	Grand Prix of Korea	11:00-16:00	Дистанционно	A, C, I, K	-
07.11.2021	Grand Prix of Siberia	11:00-16:00	Дистанционно	3, 8, 13, 14	-
14.11.2021	Grand Prix of EDG	11:00-16:00	Дистанционно	A, M, N	-
28.11.2021	Grand Prix of Southern Europe	11:00-16:00	Дистанционно	O, P	-
05.12.2021	Grand Prix of Poland	11:00-16:00	Дистанционно	H, N, O, P, R	-
12.12.2021	Grand Prix of Nanjing	11:00-16:00	Дистанционно	A, N, O, Q	-
19.12.2021	Moscow Regional Contest	11:00-16:00	Дистанционно	A, F, N	-

Явка на контесты

Дата	Название	Присутствующие
12.09.2021	Grand Prix of Dolgoprudny	Ежов, Полюбин, Тарпанов
19.09.2021	Grand Prix of IMO	Ежов, Полюбин, Тарпанов
26.09.2021	Grand Prix of XiAn	Ежов, Полюбин, Тарпанов
10.10.2021	XXII Открытая Всесибирская олимпиада	Ежов, Полюбин, Тарпанов
24.10.2021	Grand Prix of Korea	Ежов, Полюбин, Тарпанов
07.11.2021	Grand Prix of Siberia	Ежов, Полюбин, Тарпанов
14.11.2021	Grand Prix of EDG	Ежов, Полюбин, Тарпанов
28.11.2021	Grand Prix of Southern Europe	Ежов, Полюбин, Тарпанов
05.12.2021	Grand Prix of Poland	Ежов, Полюбин, Тарпанов
12.12.2021	Grand Prix of Nanjing	Ежов, Полюбин, Тарпанов
19.12.2021	Moscow Regional Contest	Ежов, Полюбин, Тарпанов

1 Stage 1-B: Grand Prix of Dolgoprudny, Division 2

L. Analysing Forests

Time limit	2 seconds
Memory limit	512Mb
Input	standard input or input.txt
Output	standard output or output.txt

Tree is a connected graph without loops (especially, the isolated vertex is the tree too). **Forest** is an union of some non-intersecting trees.
Given the forest of N vertices and M edges. Count the maximal number of the trees in that forest.

Input

First line of the input contains two integers N and M ($2 \leq M, N \leq 10^4$).


Output


Print one integer — the maximum number of the trees in the forest. If there is no forests for the given M and N , print 0 otherwise.

Examples

standard input	standard output
4 2	2
1201 2020	0

Sample 1


Input 


Output 

4 2

2

Sample 2

Input 

Output 

1201 2020

0

Идея

Если количество ребёр больше количества вершин, то количество деревьев в графе равно $m - n$, в ином же случае деревьев в графе нет, а значит их количество равно нулю. Сложность данного решения - $O(1)$.

Исходный код

```
1 | #include <iostream>
2 |
3 | int main() {
4 |     int m,n;
5 |     std::cin >> m >> n;
6 |     if( m > n){
7 |         std::cout << m - n;
8 |     }else{
9 |         std::cout << 0;
10 |    }
11 |
12 |    return 0;
13 | }
```

Положение команды

Я	29	MAI #40 Ezhov, Polubin, Tarpanov :	—	-1 02:32	—	—	—	—	—	+	01:02	—	-4 02:34	+	00:29	2	92
---	----	------------------------------------	---	-------------	---	---	---	---	---	---	-------	---	-------------	---	-------	---	----

2 Stage 2-B: Grand Prix of IMO, Division 2

M. Math

Time limit	2 seconds
Memory limit	512Mb
Input	standard input or input.txt
Output	standard output or output.txt

You are given an array a of n **distinct** positive integers. Find the number of pairs (i, j) with $1 \leq i, j \leq n$ for which the number $a_i^2 + a_j$ is a square of an integer.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 10^6$), the size of the array.
The second line of the input contains n distinct positive integers a_1, \dots, a_n ($1 \leq a_i \leq 10^6$).

Output

Output a single integer: the answer to the problem.

Example

standard input	standard output
5 1 2 3 4 5	2

Note

In the example, there are two such pairs, corresponding to $1^2 + 3 = 4 = 2^2$ and $2^2 + 5 = 9 = 3^2$.

Sample

Input 	Output 
5 1 2 3 4 5	2

Идея

Примем a_i как a , а a_j как b , в таком случае получим, что у нас $a^2 + b = c^2$. Если мы отсортируем массив исходных чисел, то сможем проверять, существует ли в нашем массиве такое b , которому удовлетворяет условию, что $b = c^2 + 2 \cdot a \cdot c$ (формула была выведена). Алгоритм заключается в следующем, после сортировки массива мы начинаем проходить его с начала, перебирая c начиная с 1, пока наш бинарный поиск не начнёт выходить за границы массива (в таком случае останавливаем `while`, и переходим к следующему элементу массива).

Сложность данного алгоритма $O(n \cdot k \cdot \log n)$, где k - максимальный квадрат, полученный в задаче.

Исходный код

```
1 | #include <vector>
2 | #include <algorithm>
3 | #include <cmath>
4 | #include <iostream>
5 | #include <map>
6 | #define ll long long
7 | using namespace std;
8 |
9 | ll square(ll toSquare){
10 |     return toSquare*toSquare;
11 | }
12 |
13 | int main(){
14 |     cin.tie(0);
15 |     cout.tie(0);
16 |     ios::sync_with_stdio(false);
17 |     int n;
18 |     cin >> n;
19 |     vector<ll> numbers(n);
20 |     for(int i = 0; i < n; ++i){
21 |         cin >> numbers[i];
22 |     }
23 |     sort(numbers.begin(), numbers.end());
24 |     ll k = 1;
25 |     bool stop = false;
26 |     auto end = numbers.end();
27 |     ll res = 0;
28 |     for(ll i = 0; i < n; ++i){
29 |         while(!stop){
30 |             auto it = lower_bound(numbers.begin(), numbers.end(), k*k + 2*numbers[i]*k)
31 |             ;
32 |             if(it == end){
33 |                 stop = true;
34 |                 continue;
35 |             }
36 |             else if(*it == k*k + 2*numbers[i]*k){
37 |                 ++res;
38 |                 ++k;
39 |                 continue;
40 |             }
41 |             else{
42 |                 ++k;
43 |             }
```

```

43     }
44     k = 1;
45     stop = false;
46 }
47 cout << res << "\n";
48 return 0;
49 }

```

Положение команды

19	MAI #40 Ezhov, Polubin, Tarpanov	—	—	—	—	—	—	—	—	+7 03:08	—	+3 03:44	+3 01:26
----	----------------------------------	---	---	---	---	---	---	---	---	-------------	---	-------------	-------------

3 Stage 3-B: Grand Prix of XiAn, Division 2

N. Easy Game

Time limit	1 second
Memory limit	512Mb
Input	standard input or input.txt
Output	standard output or output.txt

Here is a game for two players. The rule of the game is described below:

- In the beginning of the game, there are a lot of piles of beads.
- Players take turns to play. Each turn, player choose a pile i and remove some (at least one) beads from it. Then he could do nothing or split pile i into two piles with a beads and b beads. ($a, b > 0$ and $a + b$ equals to the number of beads of pile i after removing)
- If after a player's turn, there is no beads left, the player is the winner.

Suppose that the two players are all very clever and they will use optimal game strategies. Your job is to tell whether the player who plays first can win the game.

Input

There are multiple test cases. Please process till EOF.

For each test case, the first line contains a positive integer n ($n \leq 10^5$) means there are n piles of beads. The next line contains n positive integers, the i -th positive integer a_i ($a_i < 2^{31}$) means there are a_i beads in the i -th pile.

Output

For each test case, if the first player can win the game, output "Win" and if he can't, output "Lose".

Example

standard input	standard output
1	Win
1	Lose
2	Lose
1 1	
3	
1 2 3	

Sample

Input 	Output 
1	Win
1	Lose
2	Lose
1 1	
3	
1 2 3	

Идея

Так как в задаче описывается равноправная игра двух игроков, то она эквивалентна игре "НИМ" а это значит, что для каждого входных данных можно посчитать XOR сумму, если она отлична от нуля, то выигрывает второй игрок, а

если нет - первый.

Сложность данного алгоритма $O(n \cdot \log m)$ (чтобы посчитать XOR, нужно $\log_2 \max(n + m)$ итераций для чисел m и n).

Исходный код

```
1 | #include <iostream>
2 | #define ll long long
3 | using namespace std;
4 |
5 | int main(){
6 |     cin.tie(0);
7 |     cout.tie(0);
8 |     ios::sync_with_stdio(false);
9 |     int n;
10 |    ll input;
11 |    ll res = 0;
12 |    while(cin >> n){
13 |        res = 0;
14 |        for(int i = 0; i < n; ++i){
15 |            cin >> input;
16 |            res ^= input;
17 |        }
18 |        if(!res){
19 |            cout << "Lose\n";
20 |        }
21 |        else{
22 |            cout << "Win\n";
23 |        }
24 |    }
25 |    return 0;
26 | }
```

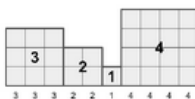
Положение команды

4 Stage 4-B: Grand Prix of Korea, Division 2

A. Histogram Sequence 3

Time limit	2 seconds
Memory limit	1Gb
Input	standard input or input.txt
Output	standard output or output.txt

Consider the histogram composed of n squares with side lengths a_1, a_2, \dots, a_n . Let's call the sequence (a_1, a_2, \dots, a_n) the histogram sequence of this histogram. Let's consider the height of each column in this histogram. The first a_1 columns will each have height a_1 , the following a_2 columns will each have height a_2 , ... and the last a_n columns will each have height a_n . Now, let us define the height sequence $(b_1, b_2, \dots, b_{a_1+a_2+\dots+a_n})$ where b_j ($1 \leq j \leq a_1 + a_2 + \dots + a_n$) is the height of the j -th column. For example, the histogram with $(3, 2, 1, 4)$ as its histogram sequence has $(3, 3, 3, 2, 2, 1, 4, 4, 4, 4)$ as its height sequence.



Write a program to find the histogram sequence given the height sequence.

Input

The first line contains a single integer m ($1 \leq m \leq 10^6$) representing the length of the height sequence $\{b_i\}$ is given.

The second line of the input contains m integers, the height sequence. Specifically, the i -th integer in the line is b_i ($1 \leq b_i \leq m$).

The input is designed such that the provided height sequence corresponds to a valid histogram sequence.

Output

Output n integers on a single line, a_1, a_2, \dots, a_n where (a_1, a_2, \dots, a_n) is the histogram sequence corresponding to the given height sequence. If there are multiple answers, any one of them will be accepted.

Examples

standard input	standard output
10 3 3 3 2 2 1 4 4 4 4	3 2 1 4
5 2 2 2 2 1	2 2 1

Sample 1

Input	Output
10 3 3 3 2 2 1 4 4 4 4	3 2 1 4

Sample 2

Input	Output
5 2 2 2 2 1	2 2 1

Идея

Считаем гистограмки....

Сложность данного алгоритма $O(n)$

Исходный код

```

1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  int main() {
7      int m;
8      cin >> m;
9      vector<int> b(m);
10     for (int i = 0; i < m; ++i) {
11         cin >> b[i];
12     }
13     vector<int> result;
14     int start;
15     for (int i = 0; i < m; ++i) {
16         start = b[i];
17         for (int j = 0; j < start - 1; ++j) {
18             ++i;
19         }
20         result.push_back(start);
21     }
22     for (auto i : result) {
23         cout << i << ' ';
24     }
25 }

```

Положение команды

19 MAI #40 Ezhov, Polubin, Tarpanov

+
00:40

+3
04:04

+12
03:37

+
03:19

4

5 Stage 5-B: Grand Prix of Siberia, Div 2

3. Кости

Ограничение времени	2 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Ходислав играет в настольную ролевую игру. Он наконец-то выбрал оружие, чтобы совладать с монстром, и наносит решающий удар. Для этого он бросает игральные кости, подсчитывает сумму чисел на гранях и называет её ведущему.

Бросок группы одинаковых костей характеризуется тремя числами n , f и m , где n — количество костей, f — количество граней каждой кости, m — модификатор. На гранях написаны все числа от 1 до f , каждая грань может выпасть, броски костей независимы. Например, если $n = 3$, $f = 8$, $m = 5$, то для определения суммы следует бросить три восьмигранные кости, сложить результаты и добавить пять; обычно это записывается в виде $3d8 + 5$.

Ведущий хочет проверить, мог ли Ходислав набрать названную им сумму после броска игровых костей.

Input

В первой строке входного файла записано единственное целое число B — количество ударов ($1 \leq B \leq 10^5$), нанесённых Ходиславом.

Далее в отдельных строках описаны удары. Первым следует целое число S — сумма, названная Ходиславом. Далее следует тройка целых чисел: n , f и m — описание группы игровых костей ($1 \leq S \leq 300$, $1 \leq n \leq 10$, $2 \leq f \leq 20$, $0 \leq m \leq 10$).

Output

Для каждого удара в порядке следования во входном файле выведите в отдельной строке YES, если сумма могла быть набрана, и NO иначе.

Examples

input.txt	output.txt
5	YES
3 1 6 0	NO
1 1 8 1	NO
16 1 12 3	NO
1 2 4 0	YES
42 3 20 1	

Идея

В цикле смотрим смог ли Ходислав выиграть бой, если сумма его очков удовлетворяет условию выигрыша, то выводим “YES”, В противном случае выводим “NO”.

Сложность данного алгоритма $O(B)$

Исходный код

```
1 | #include <iostream>
2 | using namespace std;
3 |
4 | int main(){
5 |     cin.tie(0);
6 |     cout.tie(0);
7 |     ios::sync_with_stdio(false);
8 |     int B;
9 |     cin >> B;
10 |    int n, f, m;
11 |    int subRes;
12 |    for(int i = 0; i < B; ++i){
13 |        cin >> subRes >> n >> f >> m;
14 |        if(subRes >= n + m && subRes <= n*f + m){
15 |            cout << "YES\n";
16 |        }
17 |        else{
18 |            cout << "NO\n";
19 |        }
20 |    }
21 |    return 0;
22 | }
```

Положение команды

28.	MAI #40 Ezhov, Polubin, Tarpanov	-1 1:11	-	+1 0:28	-	-	-	-2 2:17	-	-	+	2:43
-----	----------------------------------	------------	---	------------	---	---	---	------------	---	---	---	------

6 Stage 6-B: Grand Prix of EDG, Div 2

M. Pots

Time limit	1 second
Memory limit	256Mb
Input	standard input or input.txt
Output	stdandard output or output.txt

In a cooking pot shop sales decreased dramatically. Marketing managers of the shop did a research and found out that the reason was frying pans. People stopped buying pots as pans are both cheaper and more compact during storage. The Board of Directors made a decision to extent assortment and start selling also pans. The first batch is already ordered.

Warehouse logistics department was given a task to find a place for the new goods. Now there are N pots in the warehouse. Every pot has a diameter D_i . There is only one way to save space — it is possible to embed in any pot only one pot of a smaller diameter, in which others can be embedded.

Help the logistics specialist find a minimal number of pots in the warehouse in which all other pots can be embedded.

Input format

The first line contains a single number N ($1 \leq N \leq 1000$). The second line contains N integers D_i separated by spaces ($1 \leq D_i \leq 10\,000$).

Output format

Output the obtained number.

Sample

Input 	Output 
5	2
7 5 2 5 2	

Идея

Отсортируем все горшки по диаметру. Реализуем функцию, осуществляющую “помещение” наименьшего горшка в первый подходящий по размерам горшок. В цикле

будем вызывать эту функцию, до момента, пока схлопывание не перестанет происходить.

Сложность данного алгоритма $O(n \cdot \log n)$ из-за сортировки

Исходный код

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4
5 using namespace std;
6
7 bool put(vector<int> &vec, vector<bool> &used) {
8
9     for (int j = 0; j < vec.size(); ++j) {
10         if (vec[0] < vec[j] && used[j] == false) {
11             vec.erase(vec.begin());
12             used[j] = true;
13             used.erase(used.begin());
14             return true;
15         }
16     }
17     return false;
18 }
19
20 int main() {
21     int n;
22     cin >> n;
23     vector<int> vec(n);
24     vector<bool> used(n, false);
25     for (int i = 0; i < n; ++i) {
26         cin >> vec[i];
27     }
28     stable_sort(vec.begin(), vec.end());
29     while (put(vec, used));
30     cout << vec.size();
31 }
32 }
```

Положение команды

40 MAI #40 Ezhov, Polubin, Tarpanov

+1
00:16

-3
01:33

+2
00:00

7 Stage 7-B: Grand Prix of Southeastern Europe, Div 2

P. A-series

Time limit	1 second
Memory limit	256Mb
Input	standard input or input.txt
Output	standard output or output.txt

There are $N + 1$ different sizes of paper: $A_0, A_1, A_2, \dots, A_N$, where each size is twice larger than the next one.

You have a_0 pieces of paper of size A_0 , a_1 of size A_1 , ..., a_N pieces of size A_N . You want to obtain **at least** b_0 pieces of size A_0 , b_1 of size A_1 , ..., b_N pieces of size A_N . At any point you can fold and cut a paper in half, obtaining two pieces of smaller size (e.g. $A_4 \rightarrow A_5 \times 2$). What is the minimum number of cuts needed to obtain the required pieces?

Input

The first line contains a single integer N ($1 \leq N \leq 2 \cdot 10^5$).

The second line contains $N + 1$ integers a_0, a_1, \dots, a_N ($0 \leq a_i \leq 10^6$).

The third line contains $N + 1$ integers b_0, b_1, \dots, b_N ($0 \leq b_i \leq 10^6$).



Output

Output a single integer — the minimum number of cuts needed to obtain the required pieces, or -1 , if it's not possible to obtain them.

Examples

standard input	standard output
1 10 0 0 19	10
1 10 0 0 21	-1
3 2021 11 21 10 10 21 11 2021	1758

Sample 1

Input 	Output 
1 10 0 0 19	10

Sample 2

Input 	Output 
1 10 0 0 21	-1

Sample 3

Input 	Output 
3 2021 11 21 10 10 21 11 2021	1758

Идея

Будем итерироваться по двум массивам начиная с конца. На каждой итерации будем вычитать $a[i]$ из $b[i]$. После вычитания, перенесем количество листов из $b[i]$ в $b[i - 1]$ по следующему принципу: если $b[i] \% 2 == 0$, то перенесем в $b[i - 1]$ ровно $b[i]/2$ листов и прибавим к ответу $b[i]/2$. В противном случае, перенесем $(b[i] + 1)/2$ листов и прибавим столько же к ответу. В конечном счете, получим новый массив b , в первой ячейке которого будет лежать необходимое количество листов размера $A0$. Если $a[0] > b[0]$, то выводится посчитанный ранее ответ, в противном случае невозможно порезать листы так, чтобы их хватило.

Сложность данного алгоритма $O(n)$

Исходный код

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  #define int long long
7
8  int32_t main() {
9      int n;
10     cin >> n;
11     n++;
12     vector<int> a(n);
13     for (int i = 0; i < n; ++i) {
14         cin >> a[i];
15     }
16     vector<int> b(n);
17     for (int i = 0; i < n; ++i) {
18         cin >> b[i];
19     }
20     int ans = 0;
21     for (int i = n - 1; i > 0; --i) {
22         if (b[i] > a[i]) {
23             b[i] -= a[i];
24             a[i] = 0;
25         } else {
26             a[i] -= b[i];
27             b[i] = 0;
28         }
29         if (b[i] % 2 == 0) {
30             b[i - 1] += b[i] / 2;
31             ans += b[i] / 2;
32         } else {
33             b[i - 1] += (b[i] + 1) / 2;
```

```

34         ans += (b[i] + 1) / 2;
35     }
36     b[i] = 0;
37 }
38 if (a[0] - b[0] < 0) {
39     cout << -1;
40 }
41 else cout << ans << '\n';
42 }

```

Положение команды

32

MAI #40 Ezhov, Polubin, Tarpanov

-6
04:35

—

—

—

—

—

—

8 Stage 8-B: Grand Prix of Poland, Div 2

R. Unused Digits

Time limit	1 second
Memory limit	256Mb
Input	standard input or input.txt
Output	standard output or output.txt

Given a string, consisting of digits from 0 to 9. Find out all digits, which are not used in this string.



Input format

First line of the input file contains one integer T — number of the test cases ($1 \leq T \leq 150$). Each of next T lines contains one test case — non-empty string, consisting of digits from 0 to 9. Length of the given string does not exceed 30.

Output format

For each test case print in accending order all digits, which did not used in this string. If all ten digits are used, print “full” instead.

Sample

Input 	Output 
2	013579
2468	full
0123456789	

Идея

Заведем массив `used` с 10 элементами типа `bool`, изначально инициализированных как `false`. Потом пройдемся по строке `s` и будем присваивать `used[s[i] - '0']` значение `true`. После этого на каждое `i`-ое вхождение `true` будем добавлять в результирующий массив `i`. Если результирующий массив окажется пустым, выведем “full”, иначе выведем

все его элементы.

Сложность данного алгоритма $O(n)$

Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <string>
4 |
5 | using namespace std;
6 |
7 | int main() {
8 |     int n;
9 |     cin >> n;
10 |
11 |     for (int i = 0; i < n; ++i) {
12 |         string s;
13 |         cin >> s;
14 |         vector<bool> used(10, false);
15 |         for (int j = 0; j < s.size(); ++j) {
16 |             used[(s[j] - '0')] = true;
17 |         }
18 |         vector<int> ans(0);
19 |         for (int j = 0; j < used.size(); ++j) {
20 |             if (!used[j]) {
21 |                 ans.push_back(j);
22 |             }
23 |         }
24 |         if (ans.empty()) {
25 |             cout << "full\n";
26 |         }
27 |         else {
28 |             for (auto &i : ans) {
29 |                 cout << i;
30 |             }
31 |         }
32 |         cout << '\n';
33 |     }
34 |
35 | }
```

Положение команды

30

MAI #40 Ezhov, Polubin, Tarpanov

+
00:29

9 Stage 9-B: Grand Prix of Nanjing, Div 2

Q. Emptycoin

Time limit	2 seconds
Memory limit	256Mb
Input	standard input or input.txt
Output	standard output or output.txt

To estimate the stability of new cryptocurrency *Emptycoin*, the analysts are using the data on the hourly change of the price d_i of the Emptycoin. There are n observations. Consider as the *stability distance* the length of the maximum continuous segment, such as sum of d_i on that segment is zero, i.e the maximum integer k such as exists the index j : $\sum_{i=0}^{k-1} d_{j+i} = 0$. If the sum is non-zero for all k and j , then the stability distance is considered to be equal to zero.

Given the data, calculate the stability distance.

Input format

First line of the input contains one integer n — the number of the observations ($1 \leq n \leq 2 \cdot 10^5$). i -th of the following n lines contains one integer d_i ($-10^9 \leq d_i \leq 10^9$) — i -th change of the price.

Output format

Print one integer — the stability distance.

Sample 1

Input	Output
5 28 -11 11 0 0	4

Sample 2

Input	Output
4 1 2 3 4	0

Идея

Создаём вспомогательный массив, где будем хранить сумму первых n наблюдений. Также создаём массив для хранения всех встретившихся до этого сумм. Идея работы алгоритма такова - мы считаем i -ю сумму первых элементов, прибавляя введённое наблюдение к $i - 1$ результату. После вычисления текущего результата, смотрим, по-

являлась ли у нас такая сумма ранее. Если да - считаем разницу между позициями текущей и полученной равной суммы, и обновляем ответ, если только что посчитанное расстояние больше текущего.

Сложность данного алгоритма $O(n \cdot \log n)$ (из-за карты).

Исходный код

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <set>
5  #include <map>
6  using namespace std;
7  #define ll long long
8  int main(){
9      map<ll, int> poses;
10
11     int n;
12     cin >> n;
13     vector <ll> helpVector(n);
14     vector <ll> numbers(n);
15     cin >> numbers[0];
16     helpVector[0] = numbers[0];
17     poses[numbers[0]] = 0;
18     int maxLen = 0;
19     if(helpVector[0] == 0){
20         maxLen = 1;
21     }
22     for(int i = 1; i < n; ++i){
23         cin >> numbers[i];
24         helpVector[i] = helpVector[i-1] + numbers[i];
25         if(helpVector[i] == 0){
26             maxLen = max(maxLen, i+1);
27         }
28         else if(poses.find(helpVector[i]) == poses.end()){
29             poses[helpVector[i]] = i;
30         }
31         else{
32             maxLen = max(maxLen, i - poses[helpVector[i]]);
33         }
34     }
35     cout << maxLen << "\n";
36     return 0;
37 }
```

Положение команды

27	MAI #40 Ezhov, Polubin, Tarpanov :	<div><div>+</div><div>01:21</div></div>	<div><div>-1</div><div>01:20</div></div>	—	—	—	—	—	—	—	<div><div>+</div><div>00:22</div></div>
----	------------------------------------	---	--	---	---	---	---	---	---	---	---

10 Отбор на всесибирскую олимпиаду

Задача 1. Драма

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

После прочтения античной драмы и произведений Шекспира Густав решил, что многие драматические произведения можно разделить на пять частей: экспозицию, развитие действия, кульминацию, развязку и постпозицию.

При таком делении *уровень конфликта* в произведении во время экспозиции постоянен, во время развития действия строго возрастает и достигает пика в кульминации, во время развязки строго убывает и, наконец, во время постпозиции остаётся постоянным.

Экспозиция и постпозиция могут происходить мгновенно, то есть занимать один момент времени. Развитие действия и развязка занимают минимум два момента времени, кульминация всегда одномоментна. Смотрите рисунки на следующей странице.

Для нескольких произведений Густав выписал уровни конфликта в хронологическом порядке и просит вас определить, соответствуют ли они уровням конфликта при вышеописанном делении. Если соответствуют, он просит вас сообщить моменты времени, когда

1. экспозиция переходит в развитие действия — t_1 ;
2. уровень конфликта достигает кульминации — t_2 ;
3. развязка переходит в постпозицию — t_3 .

Формат входных данных

В первой строке входного файла записано одно целое положительное число D — количество произведений, проанализированных Густавом. Далее в отдельных строках следует описание этих произведений.

Первым в строке расположено целое число T_d — количество моментов времени, для которых выписаны уровни конфликта в произведении, за которым следует T_d целых положительных чисел c_t , соответствующих уровню конфликта в момент времени t ($3 \leq T_d \leq 10^5$, $1 \leq d \leq D$, $1 \leq c_t \leq 2 \cdot 10^9$, $1 \leq t \leq T_d$).

Гарантируется, что сумма всех T_d не превосходит 10^5 .

Формат выходных данных

Для каждого произведения в порядке следования во входном файле выведите в отдельной строке выходного файла: **Freytag**, если уровни конфликта в произведении изменяются в соответствии с делением Густава, и **Nein** иначе. В случае ответа **Freytag** дополнительно выведите три целых положительных числа t_1 , t_2 , t_3 — вышеописанные моменты времени ($1 \leq t_i \leq T_d$, $1 \leq i \leq 3$, $t_1 < t_2 < t_3$).

Примеры

input.txt	output.txt
6	Freytag 3 8 11
12 1 1 1 2 3 5 8 13 9 5 4 4	Nein
12 1 1 1 2 3 5 8 13 13 9 5 4	Freytag 1 2 3
3 2 3 1	Nein
3 5 5 5	Nein
12 1 1 1 2 2 5 8 13 9 5 4 4	Nein
12 1 1 1 2 3 5 8 13 9 5 6 4	

Идея

Создаем функцию solve для решения нашей задачи. В ней для каждой части акта мы проверяем нарушается ли условие Густава. Для каждого акта проходим и считаем количество сцен. Если условие нарушается, то выводим “НЕТ”, в противном случае выводим “ДА”.

Исходный код

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  void solve(vector<int> &vec) {
7      int t1 = -1, t2 = -1, t3 = -1, j = 0;
8      while (j < vec.size() - 1 && vec[j] == vec[j + 1]) {
9          j++;
10     }
11     t1 = j + 1;
12     while (j < vec.size() - 1 && vec[j] < vec[j + 1]) {
13         j++;
14     }
15     t2 = j + 1;
16     while (j < vec.size() - 1 && vec[j] > vec[j+1]) {
17         ++j;
18     }
19     t3 = j + 1;
20     while (j < vec.size() - 1 && vec[j] == vec[j + 1]) {
21         j++;
22     }
23     if (j != vec.size() - 1 || t1 == t2 || t2 == t3 || t1 == t3) {
24         cout << "Nein\n";
25     } else {
26         cout << "Freytag " << t1 << ' ' << t2 << ' ' << t3 << '\n';
27     }
28 }
29
30 int main() {
31     int d, q;
32
33     cin >> d;
34     for (int i = 0; i < d; ++i) {
35         cin >> q;
36         vector<int> vec(q);
37         for (int j = 0; j < q; ++j) {
38             cin >> vec[j];
```

```

39 |         }
40 |         solve(vec);
41 |     }
42 | }

```

Положение команды

176	MAI #40 Ezhov, Polubin, Tarpanov	99:03	239:24
-----	----------------------------------	-------	--------

11 Moscow Regional Contest

F. Build the Non-Cactus

Time limit	1 second
Memory limit	512Mb
Input	standard input or input.txt
Output	standard output or output.txt

In graph theory, a *cactus* is a connected undirected graph in which any two simple cycles have at most one vertex in common.

Given an integer n , build a **connected** graph with n vertices that is **not** a cactus. Note that your graph can't have self-loops or multiple edges. The number of edges in your graph should be minimum possible.

Input

The input consists of a single integer n ($2 \leq n \leq 1000$).

Output



If there is no connected graph of n vertices without self-loops and multiple edges that is not a cactus, print -1 in the only line of the output. Otherwise, first print positive integer m — the number of edges in your graph. Then print m lines, each containing two integers — edges of the resulting graph. Use consecutive integers $1, 2, \dots, n$ to enumerate the vertices of the graph.

If there are more than solutions with minimum number of edges, print any of them.

Example

standard input	standard output
3	-1

Sample

Input		Output	
3		-1	

Идея

Если дано более трех вершин, будем строить “круглый” граф, то есть граф, с ребрами вида $[i, i+1]$, $[i+1, i+2] \dots [i+n, i]$. Такое построение гарантирует минимально возможное количество ребер. После этого, чтобы граф перестал быть кактусом, нужно провести в нем хорду — ребро между двумя несмежными вершинами. Для простоты, в каждом графе соединим вершины 1 и 3.

Сложность данного алгоритма $O(n)$

Исходный код

```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main() {
7     int n;
8     cin >> n;
9     if (n <= 3) {
10         cout << "-1\n";
11         exit(0);
12     }
13     vector<pair<int,int>> edges;
14     for (int i = 1; i < n; i++) {
15         edges.push_back(make_pair(i, i + 1));
16     }
17     edges.push_back(make_pair(1,n));
18     edges.push_back(make_pair(1,3));
19     cout << edges.size();
20     for (auto &i: edges) {
21         cout << i.first << " " << i.second << "\n";
22     }
23 }
```

Положение команды

235 Moscow AI: MAI #40 (Nikita
Ezhov, Arseniy Polyubin, Daniil
Tarpanov)

+
00:06

—

—

—

—

+13
02:47

—

-1
02:58