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:	MAI #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 ($2 \le M, N \le 10^4$).

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

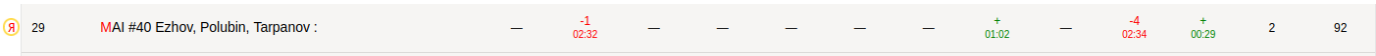
Идея

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

```cpp
#include <iostream>

int main() {
    int m,n;
    std::cin >> m >> n;
    if( m > n){
        std::cout << m - n;
    }else{
        std::cout << 0;
    }

    return 0;
}
```

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 $1 \le i, j \le n$ 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 ($1 \le n \le 10^6$), the size of the array.
The second line of the input contains n distinct positive integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^6$).

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, которою 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

```
#include <vector>
#include <algorithm>
#include <cmath>
#include <iostream>
#include <map>
#define ll long long
using namespace std;

ll square(ll toSquare){
    return toSquare*toSquare;
}

int main(){
    cin.tie(0);
    cout.tie(0);
    ios::sync_with_stdio(false);
    int n;
    cin >> n;
    vector<ll> numbers(n);
    for(int i = 0; i < n; ++i){
        cin >> numbers[i];
    }
    sort(numbers.begin(), numbers.end());
    ll k = 1;
    bool stop = false;
    auto end = numbers.end();
    ll res = 0;
    for(ll i = 0; i < n; ++i){
        while(!stop){
            auto it = lower_bound(numbers.begin(), numbers.end(), k*k + 2*numbers[i]*k)
                ;
            if(it == end){
                stop = true;
                continue;
            }
            else if(*it == k*k + 2*numbers[i]*k){
                ++res;
                ++k;
                continue;
            }
            else{
                ++k;
            }
```

```
        }
        k = 1;
        stop = false;
    }
    cout << res << "\n";
    return 0;
}
```

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 postive integer $n(n \leq 10^5)$ means there are n piles of beads. The next line contains n postive integers, the i-th postive 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, ouput "Win" and if he can't, ouput "Lose".

Example

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

Sample

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

Идея

Так как в задаче описывается равноправная игра двух игроков, то она эквивалентна игре "НИМ" а это значит, что для каждых 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

```cpp
#include <iostream>
#define ll long long
using namespace std;

int main(){
    cin.tie(0);
    cout.tie(0);
    ios::sync_with_stdio(false);
    int n;
    ll input;
    ll res = 0;
    while(cin >> n){
        res = 0;
        for(int i = 0; i < n; ++i){
            cin >> input;
            res ^= input;
        }
        if(!res){
            cout << "Lose\n";
        }
        else{
            cout << "Win\n";
        }
    }
    return 0;
}
```

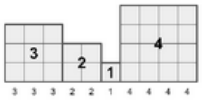
Положение команды

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 $a_1, a_2, \cdots, a_n$. Let's call the sequence $(a_1, a_2, \cdots, a_n)$ 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 $(b_1, b_2, \cdots, b_{a_1+a_2+\cdots+a_n})$ where b_j $(1 \le j \le a_1 + a_2 + \cdots + a_n)$ 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 $(1 \le m \le 10^6)$ 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 $(1 \le b_i \le m)$.
The input is designed such that the provided height sequence corresponds to a valid histogram sequence.

Output

Output n integers on a single line, $a_1, a_2, \cdots, a_n$ where $(a_1, a_2, \cdots, a_n)$ 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

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    int m;
    cin >> m;
    vector<int> b(m);
    for (int i = 0; i < m; ++i) {
        cin >> b[i];
    }
    vector<int> result;
    int start;
    for (int i = 0; i < m; ++i) {
        start = b[i];
        for (int j = 0; j < start - 1; ++j) {
            ++i;
        }
        result.push_back(start);
    }
    for (auto i : result) {
        cout << i << ' ';
    }
}
```

Положение команды

5 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 ($1 \le N \le 1000$). The second line contains N integers D_i separated by spaces ($1 \le D_i \le 10\,000$).

Output format

Output the obtained number.

Sample

Input	Output
5 7 5 2 5 2	2

Идея

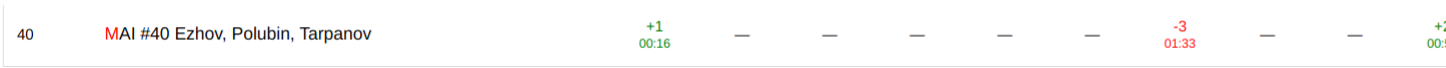
Отсортируем все горшки по диаметру. Реализуем функцию, осуществляющую “помещение” наименьшего горшка в первый подходящий по размерам горшок. В цикле

будем вызывать эту функцию, до момента, пока схлопывание не перестанет происходить.
Сложность данного алгоритма $O(n \cdot logn)$ из-за сортировки

Исходный код

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

using namespace std;

bool put(vector<int> &vec, vector<bool> &used) {

    for (int j = 0; j < vec.size(); ++j) {
        if (vec[0] < vec[j] && used[j] == false) {
            vec.erase(vec.begin());
            used[j] = true;
            used.erase(used.begin());
            return true;
        }
    }
    return false;
}

int main() {
    int n;
    cin >> n;
    vector<int> vec(n);
    vector<bool> used(n, false);
    for (int i = 0; i < n; ++i) {
        cin >> vec[i];
    }
    stable_sort(vec.begin(), vec.end());
    while (put(vec, used));
    cout << vec.size();

}
```

Положение команды



6 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: $A0$, $A1$, $A2$, ..., AN, where each size is twice larger than the next one.
You have a_0 pieces of paper of size $A0$, a_1 of size $A1$, ..., a_N pieces of size AN. You want to obtain **at least** b_0 pieces of size $A0$, b_1 of size $A1$, ..., b_N pieces of size AN. At any point you can fold and cut a paper in half, obtaining two pieces of smaller size (e.g. $A4 \to A5 \times 2$). What is the minimum number of cuts needed to obtain the required pieces?

Input

The first line contains a single integer N ($1 \le N \le 2 \cdot 10^5$).
The second line contains $N+1$ integers $a_0, a_1, \ldots, a_N$ ($0 \le a_i \le 10^9$).
The third line contains $N+1$ integers $b_0, b_1, \ldots, b_N$ ($0 \le b_i \le 10^9$).

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 перенсем в $b[i-1]$ ровно $b/2$ листов и прибавим к ответу $b[i]/2$. В противном случае, перенсем $(b[i]+1)/2$ 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

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

using namespace std;

#define int long long

int32_t main() {
    int n;
    cin >> n;
    n++;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }
    vector<int> b(n);
    for (int i = 0; i < n; ++i) {
        cin >> b[i];
    }
    int ans = 0;
    for (int i = n - 1; i > 0; --i) {
        if (b[i] > a[i]) {
            b[i] -= a[i];
            a[i] = 0;
        } else {
            a[i] -= b[i];
            b[i] = 0;
        }
        if (b[i] % 2 == 0) {
            b[i - 1] += b[i] / 2;
            ans += b[i] / 2;
        } else {
            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	—	—	—	—	—	—

7 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 ($1 \le T \le 150$). 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 2468 0123456789	013579 full

Идея

Заведем массив used с 10 элементами типа bool, изначально инциализированных 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

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

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

    for (int i = 0; i < n; ++i) {
        string s;
        cin >> s;
        vector<bool> used(10, false);
        for (int j = 0; j < s.size(); ++j) {
            used[(s[j] - '0')] = true;
        }
        vector<int> ans(0);
        for (int j = 0; j < used.size(); ++j) {
            if (!used[j]) {
                ans.push_back(j);
            }
        }
        if (ans.empty()) {
            cout << "full\n";
        }
        else {
            for (auto &i : ans) {
                cout << i;
            }
        }
        cout << '\n';
    }

}
```

Положение команды

30	MAI #40 Ezhov, Polubin, Tarpanov	—	—	—	+ 00:29	—	—

8 1/4 ICPC

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 ($2 \le n \le 1000$).

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 $1, 2, \ldots, n$ 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]...[i+n, i]. Такое построение грантирует 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

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    int n;
    cin >> n;
    if (n <= 3) {
        cout << "-1\n";
        exit(0);
    }
    vector<pair<int,int>> edges;
    for (int i = 1; i < n; i++) {
        edges.push_back(make_pair(i, i + 1));
    }
    edges.push_back(make_pair(1,n));
    edges.push_back(make_pair(1,3));
    cout << edges.size();
    for (auto &i: edges) {
        cout << i.first << " " << i.second << "\n";
    }
}
```

Положение команды