

## А Квадрат

Нужно просто возвести целое число в квадрат.

### Решение C++

```
#include<iostream>
using namespace std;
int main()
{
    long int n;
    cin>>n;
    cout<<n*n;
}
```

### Решение Python

```
print(int(input())**2)
```

## В Потепление

Нужно найти в массиве подотрезок из положительных чисел максимальной длины.

### Решение C++

```
#include <iostream>

using namespace std;

int main() {
    int n;
    cin >> n;
    int res = 0, cur = 0;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        if (x > 0)
            cur++;
        else cur = 0;
        res = max(res, cur);
    }
    cout << res << endl;
}
```

### Решение Python

```
n=input()
*k,=map(int,input().split())
w=0
m=0
for i in k:
    if i>0:
        m+=1
        w=max(m,w)
    else:
        m=0
print(w)
```

## [С Пароль](#)

Можно вручную найти каждый ответ.

### Решение C++

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    int a[] = {0, 8, 98, 986, 9876, 98764, 987654, 9876542, 98765432, 987654312};
    cout << a[n];
    return 0;
}
```

### Решение Python

```
print([8, 98, 986, 9876, 98764, 987654, 9876542, 98765432, 987654312][int(input()) - 1])
```

## [D Пополнение гардероба](#)

Ограничения задачи позволяют решать ее за  $O(N^2)$  Мы можем для каждого элемента массива вложенным циклом проверить, встречалось ли данное число ранее.

Правильнее данную задачу решать с использованием структуры данных set или map. Таким образом асимптотика алгоритма будет  $O(N \cdot \log N)$

### Решение C++

```
#include<iostream>
#include<set>
#include<vector>
using namespace std;
int main()
{
    set < int > s;
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        int a;
        cin >> a;
        if (s.find(a) == s.end())
        {
            s.insert(a);
            cout << "1 ";
        }
        else cout << "0 ";
    }

    return 0;
}
```

## Решение Python

```
n=int(input())
ts=set()
for i in map(int,input().split()):
    if i not in ts:
        print(1,end=" ")
        ts.add(i)
    else:
        print(0,end=" ")
```

## Е Кастрюли

Если все размеры кастрюль различны, то мы можем вложить их все в одну «матрешку».

Сложности возникают, когда есть одинаковые. В принципе не сложно догадаться, что нужно посчитать количество одинаковых. Максимальное из этих чисел и будет ответом.

Можно использовать структуру данных map. Хотя ограничения задачи (10 000) позволяют использовать ассоциативный массив.

## Решение C++

```
#include <bits/stdc++.h>
#define ll long long

using namespace std;
int n , d[10005];

int main()
{
    ios::sync_with_stdio(0);
    cin >> n;
    int ans = 0;
    for (int i=0;i<n;i++)
    {
        int x;
        cin >> x;
        d[x]++;
        ans = max(ans , d[x]);
    }
    cout << ans << endl;
    return 0;
}
```

## Решение Python

```
n = int(input())
A = list(map(int, input().split()))
g = {}
for i in range(n):
    if A[i] not in g:
        g[A[i]] = 0
    g[A[i]] += 1
ans = 1
for x in g:
    ans = max(ans, g[x])
print(ans)
```

## F Максимальная сумма на подотрезке

Будем поддерживать две переменные: ответ и текущая сумма чисел. Будем идти по массиву слева направо и добавлять новое число к текущей сумме. Если текущая сумма стала отрицательной, то обнуляем её. Если она больше ответа, то обновляем ответ.

### Решение C++

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

int main() {
    int n;
    cin >> n;
    long long res = -1e9, sum = 0;

    for (int i = 0; i < n; i++){
        int v;
        cin >> v;
        sum += v;
        if (sum > res){
            res = sum;
        }
        if (sum < 0)
            sum = 0;
    }

    cout << res;

    return 0;
}
```

### Решение Python

```
n = int(input())
a = list(map(int, input().split()))
sum = 0
best = -2147483647 - 1
for i in range(0, n):
    sum += a[i]
    best = max(sum, best)
    if (sum < 0):
        sum = 0
print(best)
```

Еще вариант решением динамическим программированием с использованием префиксных сумм. Как известно сумма чисел на отрезке от L до R равна  $\text{pref}[R] - \text{pref}[L-1]$ , где  $\text{pref}[n]$  – сумма первых n чисел. Будем идти по массиву слева направо и считать максимальное значение на подотрезке, заканчивающимся на текущей позиции. Для этого надо хранить минимум среди всех префиксов слева.

## G Переправа

Жадное решение. Метод двух указателей. Отсортируем всех по весу в порядке увеличения. Будем идти от самых легких.

К самому легкому будем пытаться добавить самого тяжелого. Если мы не можем этого сделать, то самый тяжелый ни с кем в паре уже точно поехать не сможет, поэтому возем его в одиночку и сдвигаем правый указатель от самого тяжелого влево.  $O(N)$

### Решение C++

```
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <unordered_set>
#include <cstring>
#include <unordered_map>
#include <algorithm>
#include <numeric>
#include <iomanip>
#include <cmath>
#include <queue>
#include <stack>
#include <fstream>
#include <utility>

typedef int64_t intt;

const intt MOD = 1000000007;
const intt inf = 1e18;

int main()
{
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(NULL);

    intt n, x; std::cin >> n >> x;
    std::vector<intt> v(n);
    for(int i = 0; i < n; i++) std::cin >> v[i];
    intt ans = 0;
    std::sort(v.begin(), v.end());
    intt i = 0, j = n-1;
    while(i <= j)
    {
        if(v[i] + v[j] <= x)
        {
            i++;
            j--;
        }
        else j--;
        ans++;
    }
    std::cout << ans;
}
```

## Решение Python

```
n, x = map(int, input().split())
P = [int(i) for i in input().split()]
i = 0
j = n - 1
num = 0
P.sort()
while i <= j:
    if P[i] + P[j] <= x:
        i += 1
    num += 1
    j -= 1
print(num)
```

## [Н Ресторан](#)

Это стандартная задача на поиск максимального количества пересекающихся отрезков на прямой. Отсортируем в одном массиве начала и концы отрезков с указанием, что это начало (+1) и конец (-1). Пройдем слева направо и будем добавлять к текущей сумме данные числа. Максимальное значение суммы и будет ответом. Это что-то похожее на проверку глубины вложенности скобочной последовательности.

## Решение C++

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long
#define ar array

ar<int, 2> positions[400002];
int main() {
    int n;
    cin >> n;

    for(int i=0; i < 2*n; i+=2) {
        int a, b;
        cin >> a >> b;

        positions[i] = {a, 1};
        positions[i+1] = {b, -1};
    }

    sort(positions, positions + 2 * n);
    int mx = 0;
    int cur = 0;
    for(int i=0; i < 2 * n; i++) {
        cur += positions[i][1];
        mx = max(cur, mx);
    }

    cout << mx;
}
```

## I Завод

Заметим, что мы можем легко посчитать количество деталей, которые можем произвести на всех станках за фиксированное время. Для этого просто пройдем по всем машинам и поделим это время на производительность каждой машины. Эти значения сложим (все машины могут работать параллельно). Также заметим, что если мы не сможем произвести нужно количество деталей за время  $T$ , то мы не сможем это сделать и за меньшее время. Это позволяет нам оптимизировать перебор ответа с помощью бинарного поиска.

По сути, это классическая задача на применение бинарного поиска по ответу.

Я рекомендую писать бинарный поиск таким образом, чтобы одна граница была всегда истиной (здесь правая), а другая всегда ложь. Цикл таким образом `while(l+1 < r) {`

### Решение C++

```
#include<bits/stdc++.h>
#define ll long long
#define st string
#define ld long double
#define ss second
#define ff first
#define ok define
using namespace std;
int main(){
    unsigned ll t, n, l = 0, r = 1e18, ans = 0, mid, x;
    cin >> n >> x;
    unsigned ll a[n];
    for(int i = 0; i < n; i++)
        cin >> a[i];
    while(l+1 < r){
        mid = (l + r) / 2;
        for(int i = 0; i < n; i++){
            ans += mid / a[i];
        }
        if(ans >= x)
            r = mid;
        else
            l = mid;
        ans = 0;
    }
    cout << r;
    return 0;
}
```

## J Сумма двух чисел

Данная задача – пример эффективного использования структуры данных **map**.

Будем идти по массиву слева направо. Пусть нам нужно получить сумму  $x$ , а текущее число  $a$ . Тогда нужно быстро проверить, а было ли уже число  $x-a$ . Это позволяет делать **map**. Асимптотика  $O(N \cdot \log N)$

Есть другой подход к решению с той же асимптотикой. Отсортируем массив. Воспользуемся методом двух указателей. Первым будем идти слева, а правый двигать влево в зависимости от суммы двух чисел и требуемого результата.

### Решение C++

```
#include <bits/stdc++.h>
#define ll long long
#define pb push_back
#define N 200000
using namespace std;

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

    map <int, int> mp;

    for (int i=0; i<n; i++)
    {
        int a; cin >> a;

        if (mp.find(x-a) != mp.end())
        {
            cout << mp[x-a] + 1 << " " << i + 1;
            return 0;
        }
        mp[a] = i;
    }

    cout << "-1 ";

    return 0;
}
```

## K Найди дробь

Пусть  $x/y$ -требуемая несократимая дробь. (Дробь  $x/y$  является несократимой в том и только в том случае, если числа  $x$  и  $y$  взаимно простые, то есть их наибольший общий делитель равен 1) Нетрудно доказать, что  $x = \text{НОК}(a_1, a_2, \dots, a_n)$ ,  $y = \text{НОД}(b_1, b_2, \dots, b_n)$ , то есть число  $x$  совпадает с наименьшим общим кратным всех чисел  $a_i$ , а число  $y$  — с наибольшим общим делителем всех чисел  $b_i$ . В частности, если все числа  $a_i$  взаимно простые, то  $x = a_1 \cdot a_2 \cdot \dots \cdot a_n$ . Поскольку все числа  $a_i$  не превосходят  $10^3$ , число  $n$  будет не более  $10^{18}$



В старых стандартах C++ есть встроенная функция `__gcd`, также в C++17 есть шаблонная реализация `std::gcd`

Также можно реализовать ее самостоятельно:

Алгоритм Евклида: на каждом шаге большее значение заменяем на остаток от деления большего на меньшее. Пока меньшее не станет равно нулю. Вроде как такая реализация работает быстрее, чем стандартные функции

#### Решение C++

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

long long gcd(long long x, long long y) {
    return y == 0 ? x : gcd(y, x % y);
}

int main() {
    long long n, a, b;
    long long lcma, gcdb;
    cin >> n;
    for (int i = 0; i < n; i++) {
        if (i == 0) {
            cin >> lcma >> gcdb;
            continue;
        }
        cin >> a >> b;
        lcma = lcma / gcd(lcma, a) * a;
        gcdb = gcd(gcdb, b);
    }
    cout << lcma << " " << gcdb << endl;
}
```

#### Решение Python

```
from math import gcd
def lcm(a, b): return a * b // gcd(a, b)

n = int(input())
c = 1
g = 0
for i in range(n):
    p, q = map(int, input().split())
    c = lcm(c, p)
    g = gcd(q, g)
print(c, g)
```

TO BE CONTINUED....