|  |  |
| --- | --- |
| [A](https://codeforces.com/gym/319807/problem/A) | [Distinct Numbers](https://codeforces.com/gym/319807/problem/A) |

Используем структуру данных set или map. Можно также решать сортировкой.

**Решение Python**

input**()**

**print(**len**({\***input**().**split**()}))**

**Решение C++**

#include <iostream>

#include <set>

int main**()** **{**

int n**;**

std**::**cin **>>** n**;**

std**::**set**<**int**>** st**;**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

int x**;**

std**::**cin **>>** x**;**

st**.**insert**(**x**);**

**}**

std**::**cout **<<** st**.**size**()** **<<** '\n'**;**

**}**

|  |  |
| --- | --- |
| [B](https://codeforces.com/gym/319807/problem/B) | [Скобочная последовательность](https://codeforces.com/gym/319807/problem/B) |

Используем структуру данных стек. В качестве стека можно использовать обычную строку.

**Решение C++**

#include <iostream>

#include <string>

#include <map>

#include <algorithm>

#include <cassert>

**using** **namespace** std**;**

int main**()**

**{**

string s**;**

cin **>>** s**;**

string st**;**

**for** **(**int i **=** 0**;** i **<** s**.**length**();** i**++)** **{**

**if** **(**st**.**empty**())** **{**

st **+=** s**[**i**];**

**continue;**

**}**

**if** **(**st**.**back**()** **==** '[' **&&** s**[**i**]** **==** ']' **||**

st**.**back**()** **==** '{' **&&** s**[**i**]** **==** '}' **||**

st**.**back**()** **==** '(' **&&** s**[**i**]** **==** ')'

**)** **{**

st**.**pop\_back**();**

**}**

**else**

st **+=** s**[**i**];**

**}**

**if** **(**st**.**length**()** **==** 0**)**

cout **<<** "YES"**;** **else**

cout **<<** "NO"**;**

**return** 0**;**

**}**

|  |  |
| --- | --- |
| [C](https://codeforces.com/gym/319807/problem/C) | [Система регистрации](https://codeforces.com/gym/319807/problem/C) |
|  |  |

Классическая задача на применение структуры данных **map**

**Решение Python**

n **=** int**(**input**())**

db **=** **{}**

**for** i **in** range**(**n**):**

name **=** input**()**

**if** name **not** **in** db**:**

db**[**name**]** **=** 1

**print(**"OK"**)**

**else:**

**print(**f'{name}{db[name]}'**)**

db**[**name**]** **+=** 1

**Решение C++**

#include<iostream>

#include<map>

**using** **namespace** std**;**

int main**(){**

map**<**string**,** int**>** bag**;**

int n**;**

cin **>>** n**;**

**while(**n**--){**

string s**;**

cin **>>** s**;**

**if(**bag**[**s**]** **==** 0**)** cout **<<** "OK" **<<** endl**;**

**else** cout **<<** s **<<** bag**[**s**]** **<<** endl**;**

bag**[**s**]++;**

**}**

**}**

|  |  |
| --- | --- |
| [D](https://codeforces.com/gym/319807/problem/D) | [Клетки не под боем](https://codeforces.com/gym/319807/problem/D) |

При добавлении новой ладьи одна строка и один столбец вычеркиваются. При условии, что на этой строке/столбце никто ранее не стоял. То есть, если изначально количество свободных клеток n\*m. То, при добавлении ладьи их становится (n-1)\*(m-1). Нужно только проверять была ли уже вычеркнута строка или столбец. Это проще всего сделать с помощью ассоциативного массива или структуры данных set.

**Решение Python**

n**,**m**=**map**(**int**,**input**().**split**())**

s1**=**set**()**

s2**=**set**()**

**for** i **in** range**(**m**):**

a**,**b**=**map**(**int**,**input**().**split**())**

s1**.**add**(**a**)**

s2**.**add**(**b**)**

**print((**n**-**len**(**s1**))\*(**n**-**len**(**s2**)),** end **=** " "**)**

**Решение C++**

#include<bits/stdc++.h>

#define ll long long

**using** **namespace** std**;**

int main**()**

**{**

ll n**,** m**,** a**,** b**;**

cin **>>** n **>>** m**;**

set **<**ll**>** x**,** y**;** // тут будут храниться вычеркнуты строки и столбцы

**while(**m**--)**

**{**

cin **>>** a **>>** b**;**

x**.**insert**(**a**);**

y**.**insert**(**b**);**

cout **<<** **(**n**-**x**.**size**())\*(**n**-**y**.**size**())** **<<** " "**;**

**}**

cout **<<** endl**;**

**return** 0**;**

**}**

|  |  |
| --- | --- |
| [E](https://codeforces.com/gym/319807/problem/E) | [Съесть массив](https://codeforces.com/gym/319807/problem/E) |

Задача решается с помощью структуры данных стек. При добавлении нового элемента смотрим 2 верхних числа в стеке. Если новое число может выбить верхнее число в стеке (оно должно быть меньше соседей), то выкидываем его из стека. Каждое новое число может выкинуть несколько чисел из стека.

**Решение C++**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <map>

#include <algorithm>

#include <cassert>

#include <vector>

**using** **namespace** std**;**

int main**()**

**{**

int n**;**

cin **>>** n**;**

vector **<**int**>** stack**;**

int a**,** b**,** c**;**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

cin **>>** c**;**

**while** **(**stack**.**size**()** **>=** 2**)** **{**

int len **=** stack**.**size**();**

a **=** stack**[**len **-** 2**];**

b **=** stack**[**len **-** 1**];**

**if** **(**b **<** a **&&** b **<** c**)** **{**

stack**.**pop\_back**();**

**continue;**

**}**

**break;**

**}**

stack**.**push\_back**(**c**);**

**}**

cout **<<** stack**.**size**();**

**return** 0**;**

**}**

|  |  |
| --- | --- |
| [F](https://codeforces.com/gym/319807/problem/F) | [Sliding Median](https://codeforces.com/gym/319807/problem/F) |

Не самая простая идейно и в плане реализации задача.

Заведем два сета. В одном будем хранить k/2 меньших чисел на отрезке, а другом k/2 больших. На каждом шаге при добавлении нового числа будем обновлять эти 2 сета.

**Решение C++**

#include <string>

#include <vector>

#include <set>

#include <algorithm>

#include <map>

#include <stack>

#define ll long long

**using** **namespace** std**;**

multiset**<**ll**>** stl**,** str**;**

ll t**,** n**,** x**,** y**,** a**,** b**,** c**,** k**,** m**[**200200**];**

int main**()** **{**

scanf**(**"%d%d"**,** **&**n**,** **&**k**);**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

scanf**(**"%d"**,** **&**c**);**

m**[**i**]** **=** c**;**

**}**

**if** **(**k **==** 1**)** **{**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

c **=** m**[**i**];**

printf**(**"%d "**,** c**);**

**}**

**return** 0**;**

**}**

**for** **(**int i **=** 0**;** i **<** **(**k **+** 1**)** **/** 2**;** i**++)** **{**

stl**.**insert**(**m**[**i**]);**

**}**

**for** **(**int i **=** **(**k **+** 1**)** **/** 2**;** i **<** k**;** i**++)** **{**

a **=** **\*(--**stl**.**end**());**

**if** **(**m**[**i**]** **>=** a**)** **{**

str**.**insert**(**m**[**i**]);**

**}**

**else** **{**

stl**.**erase**(--**stl**.**end**());**

stl**.**insert**(**m**[**i**]);**

str**.**insert**(**a**);**

**}**

**}**

c **=** **\*(--**stl**.**end**());**

printf**(**"%d "**,** c**);**

**for** **(**int i **=** k**;** i **<** n**;** i**++)** **{**

a **=** **\*(**str**.**begin**());**

**if** **(**m**[**i **-** k**]** **<** a**)** **{**

stl**.**erase**(**stl**.**find**(**m**[**i **-** k**]));**

str**.**erase**(**str**.**begin**());**

stl**.**insert**(**a**);**

**}**

**else** **{**

str**.**erase**(**str**.**find**(**m**[**i **-** k**]));**

**}**

b **=** **\*(--**stl**.**end**());**

**if** **(**m**[**i**]** **>=** b**)** **{**

str**.**insert**(**m**[**i**]);**

**}**

**else** **{**

stl**.**erase**(--**stl**.**end**());**

stl**.**insert**(**m**[**i**]);**

str**.**insert**(**b**);**

**}**

c **=** **\*(--**stl**.**end**());**

printf**(**"%d "**,** c**);**

**}**

**}**

|  |  |
| --- | --- |
| [G](https://codeforces.com/gym/319807/problem/G) | [Towers](https://codeforces.com/gym/319807/problem/G) |

Решение задачи аналогично

**Нахождение наидлиннейшей возрастающей подпоследовательности**

<https://e-maxx.ru/algo/longest_increasing_subseq_log>

Можно использовать сет. В нем будут храниться последние элементы каждой башни. При добавлении нового нужно найти позиции для него с помощью upper\_bound.

Можно вместо сета поддерживать отсортированный массив и бинарным поиском искать позицию.

**Решение Python**

**import** bisect

n **=** int**(**input**())**

a **=** **[**int**(**i**)** **for** i **in** input**().**split**()]**

li**,** cnt **=** **[],** 0

**for** i **in** a**:**

indx **=** bisect**.**bisect\_right**(**li**,** i**)**

**if** indx **>=** cnt**:**

li **+=** **[**i**]**

cnt **+=** 1

**else:**

li**[**indx**]** **=** i

**print(**cnt**)**

**Решение C++**

#include <iostream>

#include <iterator>

#include <set>

**using** **namespace** std**;**

int main**(){**

int n**,**x**;**

cin**>>**n**;**

multiset**<**int**>**s**;**

**for(**int i**=**0**;**i**<**n**;++**i**)**

**{**

cin**>>**x**;**

auto it**=**s**.**upper\_bound**(**x**);**

**if(**it**==**s**.**end**())**s**.**insert**(**x**);**

**else**

**{**

s**.**erase**(**it**);**

s**.**insert**(**x**);**

**}**

**}**

cout**<<**s**.**size**()<<**endl**;**

**}**

|  |  |
| --- | --- |
| [H](https://codeforces.com/gym/319807/problem/H) | [Nearest Smaller Values](https://codeforces.com/gym/319807/problem/H) |

Пример реализации поиска ссылки на ближайшее слева число меньше данного. O(N)

|  |
| --- |
| int arr[10100]; int link[100100];  int main() {      int n;     cin >> n;     arr[0] = -1e-9;     for (int i = 1; i <= n; i++) {                cin >> arr[i];           int pos = i - 1;            while (arr[pos] >= arr[i])                pos = link[pos];           link[i] = pos;     } } |

Также можно использовать структуру данных стек.

|  |  |
| --- | --- |
| [I](https://codeforces.com/gym/319807/problem/I) | [Слияние равных элементов](https://codeforces.com/gym/319807/problem/I) |

Будем складывать все заданные числа в вектор *a*. Пусть очередное считанное число равно *x*. Тогда пока *x* равно последнему числу в векторе, будем удалять последнее число из вектора и увеличивать *x* на единицу. Когда *x* стало не равно последнему элементу вектора, добавим его в вектор *a* и перейдем к следующему числу. После считывания и обрабатывания всех чисел нужно просто вывести элементы вектора *a*.

Также данная задача имело другое решение. Оно более стандартное, но пишется сложнее, чем предыдущее.

Для решения нам понадобится два массива *l* и *r*, а также set *eq*. В *l*[*i*] будет храниться позиция ближайшего неудаленного соседа слева, либо  - 1, если такого нет. В *r*[*i*] будет храниться позиция ближайшего неудаленного соседа справа, либо  - 1, если такого нет. В *eq* будут храниться левые позиции всех соседних пар одинаковых элементов на текущий момент. Также нам понадобиться массив *del*, где мы будем помечать удаляемые позиции.

Проинициализируем *l* и *r*, а также положим в *eq* все *i* такие, что *a*[*i*] = *a*[*i* + 1].

Пока *eq* не пустой будем выполнять следующее. Достанем из начала *eq* позицию *pos*. Это левая позиция самой левой пары одинаковых элементов, а правая позиция этой пары равна *r*[*pos*]. Удалим *pos* из *eq*. Увеличим *a*[*pos*] на единицу, так как мы будем удалять текущую пару чисел, а вместо нее добавлять новое число *a*[*pos*] + 1 (фактически, мы удаляем правое число, а левое увеличиваем на единицу). Присвоим *del*[*r*[*pos*]] значению true, так как мы удаляем правое число из текущей пары. Если в *eq* содержалось значение *r*[*pos*], то его нужно удалить, так как теперь этого числа нет.

Теперь нужно пересчитать значения *l* и *r* следующим образом: *r*[*pos*] = *r*[*r*[*pos*]] и *l*[*r*[*r*[*pos*]]] = *pos*.

Теперь у числа в позиции *pos* новый сосед справа (если он был у *r*[*pos*]), а также значение *a*[*pos*] могло стать равным значению *a*[*l*[*pos*]]. Поэтому в *eq* нужно добавить значение *l*[*pos*], если *a*[*pos*] имеет левого соседа и стало ему равно. Также нужно добавить в *eq* значение *r*[*pos*], если *a*[*pos*] теперь имеет нового правого соседа и равно ему.

После того, как *eq* стал пустым, нужно с помощью массива взять все *a*[*i*], которые неудалены. Это можно сделать с помощью массива *del* — просто взять все такие *i*, что *del*[*i*] имеет значение false, так как это указывает на то, что число в позиции *i* не было удалено.

**Решение Python**

n**,** m **=** int**(**input**()),** 0

b **=** **[**int**(**t**)** **for** t **in** input**().**split**()]**

a **=** **[]**

**for** i **in** range**(**n**):**

a**.**append**(**b**[**i**])**

**while** len**(**a**)** **>** 1 **and** a**[-**2**]** **==** a**[-**1**]:**

**del** a**[-**1**]**

a**[-**1**]** **+=** 1

**print(**len**(**a**))**

**print(**" "**.**join**([**str**(**t**)** **for** t **in** a**]))**

**Решение C++**

#include <bits/stdc++.h>

**using** **namespace** std**;**

const int N **=** 1e6**+**5**;**

int n**,**k**,**x**,**a**[**N**];**

int main**(){**

scanf**(**"%d"**,&**n**);**

**for** **(**int i**=**1**;** i**<=**n**;** i**++)** **{**

cin**>>**x**;**

**while** **(**x**==**a**[**k**])** k**--,**x**++;**

k**++;**

a**[**k**]=**x**;**

**}**

printf**(**"%d\n"**,**k**);**

**for** **(**int i**=**1**;** i**<=**k**;** i**++)**

cout**<<**a**[**i**]<<**" "**;**

**return** 0**;**

**}**

|  |  |
| --- | --- |
| [J](https://codeforces.com/gym/319807/problem/J) | [Movie Festival II](https://codeforces.com/gym/319807/problem/J) |

Отсортируем все отрезки по началу. Будем идти по порядку, поддерживая мультисет, в котором будем хранить время окончания.

**Решение C++**

#include <bits/stdc++.h>

**using** **namespace** std**;**

#define stpr setprecision

#define ll long long

#define ff first

#define ss second

#define MOD 1000000007

#define INF 0x3f3f3f3f

const int maxn **=** 2e5**;**

int n**,** k**,** ans**;**

pair**<**int**,** int**>** a**[**maxn**];**

multiset**<**int**>** s**;**

int main **()** **{**

ios**::**sync\_with\_stdio**(**0**);**

cin**.**tie**(**0**);**

cin **>>** n **>>** k**;**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

cin **>>** a**[**i**].**ss **>>** a**[**i**].**ff**;**

**}**

sort**(**a**,** a**+**n**);**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

auto it **=** s**.**lower\_bound**(**a**[**i**].**ss**+**1**);**

**if** **(**k **&&** it **==** s**.**begin**())** **{**

s**.**insert**(**a**[**i**].**ff**);**

**--**k**;**

**++**ans**;**

**}** **else** **if** **(**it **!=** s**.**begin**())** **{**

**--**it**;**

s**.**erase**(**s**.**find**(\***it**));**

s**.**insert**(**a**[**i**].**ff**);**

**++**ans**;**

**}**

**}**

cout **<<** ans **<<** '\n'**;**

**return** 0**;**

**}**