Standard Template Library (STL)

PCFIM

Estructuras de datos

Vector

Inicializacion

Similar a un array, pero se puede modificar el tamaño durante el tiempo de ejecucion:

```
//vector a de int con tamaño 0
vector <int> a;

//vector b de strings con tamaño 10
vector <string> b(10);

//vector c de pairs de tamaño 5 iniciado con {1, 2}
vector <pair <int, int> > c(5, {1, 2});

//vector x de tamaño 10 de vectores de int
vector <vector <int> > x(10);
```

Estructuras de datos

Vector

Operaciones

```
vector <int> a; // a = {};
a.push_back(1); // a = {1};
a.push_back(3); // a = {1, 3}
a.push_back(2); // a = {1, 3, 2}
a.pop_back(); // a = {1, 3}
int sz = a.size(); // sz = 2;
a.push_back(-1); // a = {1, 3, -1}
a.push_back(6); // a = {1, 3, -1, 6}
a.push_back(4); // a = {1, 3, -1, 6, 4}
sort(a.begin(), a.end()); // a = {-1, 1, 3, 4, 6}
a.clear(); // a = {}
```

Estructuras de datos

Set

Un set es un conjunto ordenado de elementos no repetidos:

```
set <int> s; // O(1), s = {}
s.insert(2); // O(log(n)), s = {2}
s.insert(-1); // O(log(n)), s = {-1, 2}
s.insert(4); // O(log(n)), s = {-1, 2, 4}
s.insert(2); // O(log(n)), s = {-1, 2, 4}

int sz = s.size(); // O(1), sz = 3

if(s.find(4) != s.end()) // O(log(n))
    cout << "El numero 4 esta en el set";

for(auto it : s) // -1 2 4
    cout << *it << " ";

s.clear(); // O(n), s = {}
```

Estructuras de datos

Map

Un Map o Hash-table es una estructura de datos que nos permite asociar una `key` con un `value`:

```
map <char, int> m; // O(1)
//O(5*log(n))
m['p'] = 1; m['c'] = 2; m['f'] = 3; m['i'] = 4; m['m'] = 5;
for(auto it : m) // O(nlog(n))
    cout << it.first << " " << it.second << endl;
/*
c 2
f 3
i 4
m 5
p 1
*/
```

Estructuras de datos

Queue

Un queue es una estructura del tipo FIFO (First In, First Out), puede ser interpretada como una `cola`:

```
//O(1)
queue <int> q;
q.push(1); // q = {1}
q.push(5); // q = {1, 5}
q.push(2); // q = {1, 5, 2}
q.push(0); // q = {1, 5, 2, 0}

q.pop(); // q = {5, 2, 0}
int v = q.front(); // v = 5
int l = q.back(); // l = 0
q.pop(); // q = {2, 0}
q.pop(); // q = {0}
q.pop(); // q = {}
```

Estructuras de datos

Stack

Un stack es una estructura del tipo LIFO (Last In, First Out), puede ser interpretado como una `pila`:

```
//O(1)
stack <int> st;
st.push(3); // st = {3}
st.push(5); // st = {3, 5}
st.push(-2); // st = {3, 5, -2}

st.pop(); // st = {3, 5}
int v = st.top(); // v = 5
st.pop(); // st = {3}
```

Estructuras de datos

Deque

Multiset

Multimap

Funciones de la STL

Sort

Quick sort

Radix sort

Merge sort

STL Sort

```
vector <int> a = {3, 6, 4, 0, 7, 9};
int b[] = {3, 6, 4, 0, 7, 9};
sort(a.begin(), a.end());
sort(b, b+6);
```

Problemas con STL

- https://codeforces.com/problemset/problem/189/A (x)
- https://codeforces.com/problemset/problem/787/A (x)
- https://codeforces.com/problemset/problem/304/A (x)
- https://codeforces.com/contest/614/problem/A (x)
- https://codeforces.com/contest/593/problem/A
- https://codeforces.com/problemset/problem/33/C
- https://codeforces.com/problemset/problem/181/B
- https://codeforces.com/problemset/problem/252/B
- https://codeforces.com/problemset/problem/633/D

Pythagorean Theorem II

Revisar algoritmo de Euclides para la generacion de ternas pitagoricas primitivas:

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

set <pair<int, int> > s;

int main(){
    int n; cin >> n;
    int ans = 0;
    for(int p=1; p*p<=n; p++){
        for(int q=1; q<p; q++){
            int a = min(p*p-q*q, 2*p*q);
            int b = max(p*p-q*q, 2*p*q);
            int c = p*p+q*q;
            if(c > n) break;
            int g = __gcd(__gcd(a, b), c);
            a /= g; b /= g; c /= g;
            if(s.find({a, c}) == s.end()){
                s.insert({a, c});
                ans += n/c;
            }
        }
    }
    cout << ans << '\n';
    return 0;
}
```

Link/Cut Tree

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

int main(){
        long long l, r; cin >> l >> r;
        int k; cin >> k;
        long long pow_k = 1;
        bool found = false;
        for(;;){
                if(pow_k >= l){
                        found = true;
                        cout << pow_k << ' ';
                }
                if(pow_k > r/k) break;
                pow_k *= k;
        }
        if(!found) cout << -1;
        return 0;
}
```