EDITORIA MEIA MARATONA EU 2023

Palavrão

O desafio proposto demandava o conhecimento sobre os comandos de leitura de strings e o cálculo do tamanho de uma string.

```
#include <iostream>                                                    usi

int main() {
    string palavra;
    cin >> palavra;
    if(palavra.size() >= 10){
        cout << "palavrao\n";
    } else {
        cout << "palavrinha\n";
    }
}
```

A Batalha dos Cinco Exércitos

Para resolver esse problema, é necessário analisar os números de cada exército apresentado na entrada. Os valores são referentes aos exércitos de humanos (H), elfos (E), anões (A), orcs (O), wargs (W) e águias (X). A ideia é somar os números dos exércitos do lado do bem (humanos, elfos, anões e águias) e comparar com a soma do lado do mal (orcs e wargs).

```
#include <iostream>
using namespace std;
int main(){
    int n1 = 0, n2 = 0, a, entrada;
    for(int i = 0; i < 3; i++){
        cin >> entrada;
        n1 += entrada;
    }
    for(int i = 0; i < 2; i++){
        cin >> entrada;
        n2 += entrada;
    }
    cin >> a;
    if(n1 <= n2){
        n1+=a;
        if(n1 < n2){
            cout << "Sauron has returned.\n";
        }else{
            cout << "Middle-earth is safe.\n";
        }
    }else{
        cout << "Middle-earth is safe.\n";
    }
}
```

O Rolê Bad Vibes

Primeiramente, a entrada do programa será o número de linhas (N) e colunas (M) da matriz, seguido pelos valores da matriz em si. Em seguida, é preciso ordenar essa matriz de forma que os problemas de vida (V) apareçam primeiro, independentemente da sua criticidade, seguidos pelos problemas de disciplina (D), também ordenados pela criticidade.

A saída deve apresentar um relatório ordenado conforme o pedido por Úrisson, mostrando os problemas resolvidos, primeiro os de vida ordenados e, em seguida, os de disciplina ordenados pela criticidade.

Para resolver esse problema, será necessário ler os dados de entrada, ordenar a matriz de acordo com os critérios especificados e exibir o relatório ordenado como saída.

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main(){
    vector<string> vida;
    vector<string> disc;
    int linhas, colunas;
    cin >> linhas >> colunas;
    for(int linha = 0; linha < linhas; linha++) {
        for(int coluna = 0; coluna < colunas; coluna++) {
            string palavra;
            cin >> palavra;
            if(palavra[1] == 'V') {
                vida.push_back(palavra);
            } else {
                disc.push_back(palavra);
            }
        }
    }
    sort(vida.begin(), vida.end());
    sort(disc.begin(), disc.end());
    for(int i = vida.size() - 1; i >= 0; i--){
        cout << vida.at(i) << endl;
    }
    for(int i = disc.size() - 1; i >= 0; i--){
        cout << disc.at(i) << endl;
    }
    return 0;
}
```

Envelope

O programa precisa verificar se as dimensões do envelope menor são estritamente menores que as do envelope maior em ambas as dimensões (largura e altura). Se isso for verdade para ambos os lados, o primeiro envelope pode ser colocado dentro do segundo.

```cpp
#include <iostream>
#include <map>
using namespace std;
int main(){
    int n; cin >> n;
    while(n--){
        short l1{}, b1{}, l2{}, b2{};
        cin >> l1 >> b1 >> l2 >> b2;
```

```
        if(((l1 < l2) && (b1 < b2)) || ((l1 < b2) && (b1 < l2))){
            cout << "S" << endl;
        } else {
            cout << "N" << endl;
        }
    }
    return 0;
}
```

ClickBait

A entrada do problema é : n (o número de espectadores) e k (a meta de visualizações). Cada espectador é representado por dois inteiros a_i e b_i. Dado um tempo $t \geq a_i$, para saber quantas vezes um dado espectador i assistiu o filme, podemos realizar o seguinte cálculo:

$$num\_visualizacoes = \lceil (t - a_i)/b_i \rceil$$

Para encontrar o primeiro momento em que a meta foi atingida podemos realizar uma busca binária.

O enunciado garante que o tempo em que a meta é atingida é menor ou igual $R = 2 \times 10^18$. A complexidade de tempo será $\mathcal{O}(NlogR)$

A seguir, a solução apresentada pela equipe 404- AC Not Found.

```
#include <bits/stdc++.h>
#define fi first
#define se second
#define all(x) x.begin(), x.end()
#define rall(x) x.rbegin(), x.rend()
using namespace std;
using i64 = long long;
using pii = pair<int, int>;
using pll = pair<i64, i64>;

const int MAXN = 1e5+5;
const int INF = 0x3f3f3f3f;

void solve() {
    i64 n, k; cin >> n >> k;

    vector<pll> v;

    for (int i = 0; i < n; ++i) {
        i64 a, b;
        cin >> a >> b;

        v.push_back({a, b});
    }

    i64 lo = 1, hi = 2e18, ans = -1;
    while (lo <= hi) {
        i64 mid = lo + (hi - lo) / 2;
        // mid é a hora que vou testar

        // cout << lo << ' ' << hi << '\n';

        i64 cnt = 0;
        for (int i = 0; i < n; ++i) if (v[i].fi <= mid) {
```

```
            if (cnt >= k) break;
            cnt += (mid - v[i].fi) / v[i].se + 1;
        }

        if (cnt >= k) {
            ans = mid;
            hi = mid - 1;
        } else {
            lo = mid + 1;
        }
    }

    cout << ans << '\n';
}

int main() {
    cin.tie(0)->sync_with_stdio(0);

    int tc = 1;
    while (tc--)solve();
}
```

Pudim dos Guerreiros

Pulo do Sapo

Cartas Para o Noel

Ajudando Gust-Avô

Esse problema envolve encontrar o maior valor i para o qual a sequência B^i (gerada a partir da sequência B original) é uma subsequência da sequência A fornecida.

Dado duas palavras A e B e um inteiro i. Se B^i não é uma subsequência de A, então B^j não é subsequência de A para todo $j \geq i$.

Por outro lado, Dado duas palavras A e B e um inteiro i. Se B^i é uma subsequência de A, então B^j é subsequência de A para todo $j \leq i$.

Utilizaremos uma busca binária para encontrar o maior valor i tal que B^i é uma subsequência da sequência A. O custo para checar se B^i é uma subsequência da sequência A é $\mathcal{O}(nmi)$, onde n é o comprimento de A e m é o comprimento de B.

O tamanho do maior valor de i será $T = \lceil \frac{n}{m} \rceil$. A complexidade de tempo do algoritmo $\mathcal{O}(nmTlogT)$

```
#include <bits/stdc++.h>
#define fi first
#define se second
#define all(x) x.begin(), x.end()
#define rall(x) x.rbegin(), x.rend()
#define sz(x) (int) x.size()
using namespace std;
using i64 = long long;
using pii = pair<int, int>;
using pll = pair<i64, i64>;

const int MAXN = 1e5+5;
```

```cpp
const int INF = 0x3f3f3f3f;

string a, b, c;

bool ok(int test) {
    // cout << a << '-' << c << '\n';

    int j = 0;

    for (int i {}; i < sz(a); ++i) {
        if (c[j] == a[i]) {
            ++j;
        }
    }

    return (j == c.size());
}

void solve() {
    cin >> a >> b;

    int p1 {1}, p2 {(int)a.size() / (int)b.size() + 2};
    int ans {};

    while (p1 <= p2) {
        int mid {(p2 + p1) / 2};

        // cout << p1 << " " << p2 << " " << mid << '\n';

        c.clear();
        for (int i = 0; i < b.size(); ++i) {
            for (int j = 0; j < mid; ++j) {
                c += b[i];
            }
        }

        if (ok(mid)) {
            p1 = mid + 1;
            ans = mid;
        } else {
            p2 = mid - 1;
        }
    }

    cout << ans << '\n';
}

int main() {
    cin.tie(0)->sync_with_stdio(0);


    int tc; cin >> tc;
    while (tc--)solve();
}
```