Soluzioni Training Olinfo

Andrea Maria Lanocita

20 agosto 2024

Sommario

Questo documento vuole essere una raccolta di soluzioni ai problemi di training.olinfo.it, principalmente quelli dell'anno 2023-2024. Insieme al codice sarà inclusa una breve spiegazione del ragionamento effettuato, eventualmente diviso per subtask. Per segnalare errori o imprecisioni potete contattarmi a andrea.lanocita@gmail.com.

Indice

Capitolo 1

Finale OII - Bergamo 2023

1.1 Vetrate Colorate - artemoderna

Subtask 2

E' dimostrabile che esista sempre una soluzione al problema. Infatti:

- Se l'ordine fosse decrescente, basterebbe ruotare insieme l'intera vetrata
- Se l'ordine fosse crescente, si potrebbe ruotare singolarmente ciascun vetro

```
#include <bits/stdc++.h>

using namespace std;

#define pb push_back
#define popb pop_back
#define vi vector<int>
#define pi pair<int, int>
#define vp vector<pi>
#define ll long long int
#define pl pair<ll, ll>
#define vl vector<ll>
#define vpl vector<pl>
#define omap map<int, int>
#define umap unordered_map<int, int>
#define pq priority_queue<pair<ll, int>, vector<pair<ll, int>>,
    greater<pair<ll, int>>>

bool ordina(int N, vector<int> V, vector<int> &L) {
    int i=0;
    while(i!=N-1 && V[i+1]==V[i]) i++;
    if(i==N-1 || V[i+1] < V[i]) {
        L.push_back(N);
        return 1;
    }
    else {
        for(int j=0; j<N; j++) L.push_back(1);
        return 1;
    }
}
```

Subtask 3

L'idea generale prevede di dividere le vetrate in segmenti $[a, b]$ che, girati insieme, rendano la sottosequenza $[0, b]$ ordinata.

Proposizione. *Sia data una sequenza di 0 ed 1 da ordinare in ordine crescente dividendola in sottosequenze contigue e invertendo ciascuna. Se tale ordinamento esiste, è ottenibile dividendo la sequenza dopo ciascuno 0 seguito da un 1.*

Dimostrazione. Assumiamo che esista l'ordinamento cercato e applichiamo l'algoritmo descritto. Sia $[0, a]$ una delle sottosequenze ottenute. Allora:

- La sottosequenza è formata esclusivamente da zeri
- La sottosequenza presenta n 1, nelle posizioni $[0, n-1]$

In ogni altro caso infatti, la sottosequenza conterrebbe uno 0 seguito da un 1, che è impossibile per costruzione. È facile vedere come in ciascuno dei due casi possibili, invertire la sottosequenza porti al suo ordinamento.
Ora è necessario dimostrare come, nel caso in cui con questo algoritmo non ottenessimo una sequenza ordinata, l'ordinamento cercato non esista. Per assurdo, non dividiamo la sequenza dove è presente uno 0 con successivamente un 1, ma in una posizione successiva. Dopo l'inversione della sottostringa, otterremmo per forza un 1 seguito da uno 0, quindi l'ordinamento non sarebbe corretto. Di conseguenza, l'unico algoritmo che può portare ad un risultato valido è quello descritto. □

Nel problema è specificato come tutta la vetrata vada ruotata, quindi bisogna ricordarsi di ruotare la sottosequenza rimasta una volta arrivati alla fine dell'array.

```
#include <bits/stdc++.h>

using namespace std;

#define pb push_back
#define popb pop_back
#define vi vector<int>
#define pi pair<int, int>
#define vp vector<pi>
#define ll long long int
#define pl pair<ll, ll>
#define vl vector<ll>
#define vpl vector<pl>
#define omap map<int, int>
#define umap unordered_map<int, int>
#define pq priority_queue<pair<ll, int>, vector<pair<ll, int>>,
    greater<pair<ll, int>>>

bool ordina(int N, vector<int> V, vector<int> &L) {
    // va girato alla fine di tutti gli zero
    int prec=0;
    for(int i=1; i<N+1; i++) {
        if(i==N) {
            L.push_back(i-prec);
            prec += L[L.size()-1];
        }
        else if(V[i-1]==0 && V[i]==1) {
            L.push_back(i-prec);
            prec += L[L.size()-1];
        }
    }
    prec=0;
    for(int i=0; i<L.size()-1; i++) {
        if(V[L[i+1]+L[i]+prec-1]<V[prec]) return 0;
        prec+=L[i];
    }
    return 1;
}
```

Soluzione Ottimale

È possibile estendere la proposizione precedente al caso generale, per cui la divisione in sottosequenze va effettuata tra (a, b), dovunque $a < b$.

```cpp
#include <bits/stdc++.h>

using namespace std;

#define pb push_back
#define popb pop_back
#define vi vector<int>
#define pi pair<int, int>
#define vp vector<pi>
#define ll long long int
#define pl pair<ll, ll>
#define vl vector<ll>
#define vpl vector<pl>
#define omap map<int, int>
#define umap unordered_map<int, int>
#define pq priority_queue<pair<ll, int>, vector<pair<ll, int>>,
    greater<pair<ll, int>>>

bool ordina(int N, vector<int> V, vector<int> &L) {
    int prec=0;
    for(int i=1; i<N+1; i++) {
        if(i==N) {
            L.push_back(i-prec);
            prec += L[L.size()-1];
        }
        else if(V[i-1]<V[i]) {
            L.push_back(i-prec);
            prec += L[L.size()-1];
        }
    }
    prec=0;
    for(int i=0; i<L.size()-1; i++) {
        if(V[L[i+1]+L[i]+prec-1]<V[prec]) return 0;
        prec+=L[i];
    }
    return 1;
}
```