Grafos

Fundamentos: problemas resolvidos

Prof. Edson Alves - UnB/FGA

2019

Sumário

UVA 11991 – Easy Problem from Rujia Liu?

Problema

"Though Rujia Liu usually sets hard problems for contests (for example, regional contests like Xi'an 2006, Beijing 2007 and Wuhan 2009, or UVa OJ contests like Rujia Liu's Presents 1 and 2), he occasionally sets easy problem (for example, 'the Coco-Cola Store' in UVa OJ), to encourage more people to solve his problems :D"

Given an array, your task is to find the k-th occurrence (from left to right) of an integer v. To make the problem more difficult (and interesting!), you'll have to answer m such queries.

Entrada e saída

Input

There are several test cases. The first line of each test case contains two integers n, m $(1 \leq n, m \leq 100,000)$, the number of elements in the array, and the number of queries. The next line contains n positive integers not larger than 1,000,000. Each of the following m lines contains two integer k and v $(1 \leq k \leq n, 1 \leq v \leq 1,000,000)$. The input is terminated by end-of-file (EOF).

Output

For each query, print the 1-based location of the occurrence. If there is no such element, output '`0`' instead.

Exemplo de entradas e saídas

Sample Input

```
8 4
1 3 2 2 4 3 2 1
1 3
2 4
3 2
4 2
```

Sample Output

```
2
0
7
0
```

Solução com complexidade $O(M)$

- Cada *query* pode ser respondida em $O(1)$, se o problema for interpretado como uma lista de adjacências
- Para tal, associe a um vértica cada número inteiro positivo de 1 a N e a cada valor distinto do vetor a
- Se o valor v ocorre na i-ésima posição do vetor a, adicione a aresta direcionada (a, i) ao grafo G
- A *query* (v, k) pode ser respondida em $O(1)$ se o grafo G for representado por uma lista de adjacências, usando um `vector` para cada lista
- Basta verificar o tamanho do `vector` associado ao vértice v: se ele tem k ou mais elementos, basta retornar o valor que ocupa a posição $k - 1$
- Caso contrário, retorne zero

Solução com complexidade $O(M)$

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

using namespace std;
using ii = pair<int, int>;

const int MAX { 1000005 };
vector<int> vs[MAX];

vector<int> solve(const vector<int>& xs, const vector<ii>& qs)
{
    for (int i = 0; i < MAX; ++i)
        vs[i].clear();

    for (size_t i = 0; i < xs.size(); ++i)
        vs[xs[i]].push_back(i + 1);

    vector<int> ans;

    for (const auto& q : qs)
    {
        auto k = q.first, v = q.second;
```

Solução com complexidade $O(M)$

```
23          ans.push_back(k <= (int) vs[v].size() ? vs[v][k-1] : 0);
24      }
25
26      return ans;
27  }
28
29  int main()
30  {
31      ios::sync_with_stdio(false);
32
33      int N, M;
34
35      while (cin >> N >> M)
36      {
37          vector<int> xs(N);
38
39          for (int i = 0; i < N; ++i)
40              cin >> xs[i];
41
42          vector<ii> qs(M);
43
```

Solução com complexidade $O(M)$

```
44          for (int i = 0; i < M; ++i)
45              cin >> qs[i].first >> qs[i].second;
46
47          auto ans = solve(xs, qs);
48
49          for (const auto& x : ans)
50              cout << x << '\n';
51      }
52
53      return 0;
54  }
```

Codeforces Round #464 (Div. 2) – Problem A: Love Triangle

Problema

As you could know there are no male planes nor female planes. However, each plane on Earth likes some other plane. There are n planes on Earth, numbered from 1 to n, and the plane with number i likes the plane with number f_i, where $1 \leq f_i \leq n$ and $f_i \neq i$.

We call a love triangle a situation in which plane A likes plane B, plane B likes plane C and plane C likes plane A. Find out if there is any love triangle on Earth.

Entrada e saída

Input

The first line contains a single integer n $(2 \leq n \leq 5000)$ – the number of planes.

The second line contains n integers $f_1, f_2, \ldots, f_n$ $(1 \leq f_i \leq n, f_i \neq i)$, meaning that the i-th plane likes the f_i-th.

Output

Output «YES» if there is a love triangle consisting of planes on Earth. Otherwise, output «NO».

Exemplo de entradas e saídas

Sample Input	Sample Output
5 2 4 5 1 3	YES
5 5 5 5 5 1	NO

Solução com complexidade $O(N)$

- Considere que a cada plano seja associado um vértice u
- De cada vértice u parte uma única aresta direcionada (u, f_u)
- O problema consiste em verificar se o caminho

$$A \to (B = f_A) \to (C = f_B) \to (D = f_C)$$

  é um ciclo, isto, é, se $A = D$
- Como o caminho tem apenas 3 arestas para qualquer vértices, o algoritmo é $O(N)$

Solução AC com complexidade $O(N)$

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

using namespace std;

bool solve(int N, const vector<int>& fs)
{
    for (int A = 1; A <= N; ++A)
    {
        auto B = fs[A];
        auto C = fs[B];

        if (fs[C] == A)
            return true;
    }

    return false;
}

int main()
{
    ios::sync_with_stdio(false);
```

Solução AC com complexidade $O(N)$

```
22
23      int N;
24      cin >> N;
25
26      vector<int> fs(N + 1);
27
28      for (int i = 1; i <= N; ++i)
29          cin >> fs[i];
30
31      auto ans = solve(N, fs);
32
33      cout << (ans ? "YES" : "NO") << '\n';
34
35      return 0;
36  }
```

Referências

1. 11991 – Easy Problem from Rujia Liu?
2. Codeforces Round #464 (Div. 2) – Problem A: Love Triangle