Grafos

Fundamentos: problemas resolvidos

Prof. Edson Alves - UnB/FGA
2019

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
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ii = pair<int, int>;
5
6 const int MAX { 1000005 };
7 vector<int> vs[MAX];
8
9 vector<int> solve(const vector<int>& xs, const vector<ii>& qs)
10 {
11     for (int i = 0; i < MAX; ++i)
12         vs[i].clear();
13
14     for (size_t i = 0; i < xs.size(); ++i)
15         vs[xs[i]].push_back(i + 1);
16
17     vector<int> ans;
18
19     for (const auto& q : qs)
20     {
21         auto k = q.first, v = q.second;
```

6

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 $O(M)$

```cpp
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 }
```

Referências

1. 11991 – Easy Problem from Rujia Liu?