OJ 11991

Easy Problem from Rujia Liu?

Prof. Edson Alves

Faculdade UnB Gama

"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.

“Embora Rujia Liu geralmente crie problemas difíceis para as competições (por exemplo, competições regionais como Xi’an 2006, Beijing 2007 e Wuhan 2009, ou competições do OJ como Rujia Liu’s Presents 1 e 2), de vez em quando ele cria um problema fácil (por exemplo, ‘the Coco-Cola Store’ no OJ), para encorajar mais pessoas a resolverem seus problemas :D”

Dado um vetor, sua tarefa é encontrar a k-ésima ocorrência (da esquerda para direita) de um inteiro v. Para tornar o problema mais difícil (e interessante!), você deverá responder m consultas deste tipo.

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.

Entrada

Há vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros n, m $(1 \leq n, m \leq 100,000)$, o número de elementos no vetor e o número de consultas. A próxima linha contém n inteiros positivos menores ou iguais a 1,000,000. Cada uma das m linhas seguintes contém dois inteiros k e v ($1 \leq k \leq n, 1 \leq v \leq 1,000,000$). A entrada é terminada por fim de arquivo (EOF).

Saída

Para cada consulta imprima a localização da ocorrência, começando em 1. Se não existe tal elemento, imprima '0'.

Exemplo de entrada e saída

```
8 4
```

Exemplo de entrada e saída

Exemplo de entrada e saída

```
8 4

1 3 2 2 4 3 2 1
```

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1 ← *elementos*

Exemplo de entrada e saída

```
8 4

1 3 2 2 4 3 2 1

1 3
```

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ← *do elemento 3*

↑

primeira ocorrência

Exemplo de entrada e saída

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

1 3
```

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ 2

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ 2

2 4

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ **2**

2 4 ⟵ *do elemento 4*

segunda ocorrência

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ 2

2 4 ⟶ 0

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ 2

2 4 ⟶ 0

3 2

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ 2

2 4 ⟶ 0

3 2

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ 2

2 4 ⟶ 0

3 2 ⟶ 7

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ 2

2 4 ⟶ 0

3 2 ⟶ 7

4 2

Exemplo de entrada e saída

8 4

1 3 2 2 4 3 2 1

1 3 ⟶ 2

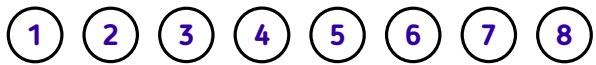
2 4 ⟶ 0

3 2 ⟶ 7

4 2 ⟶ 0

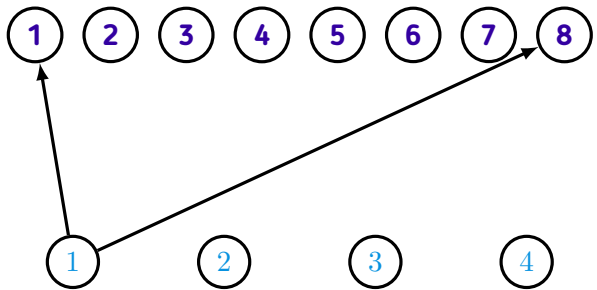
Solução

Solução

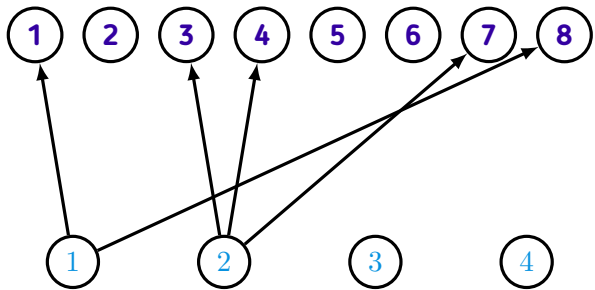


Solução

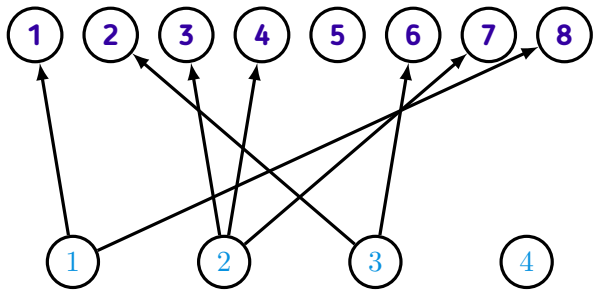
Solução



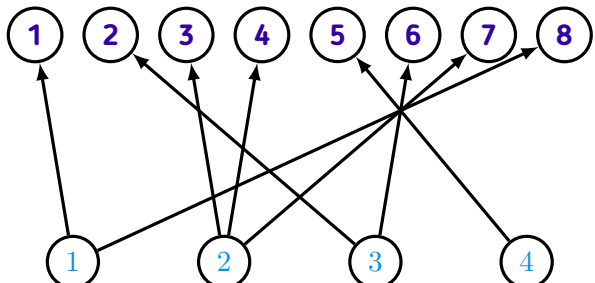
Solução



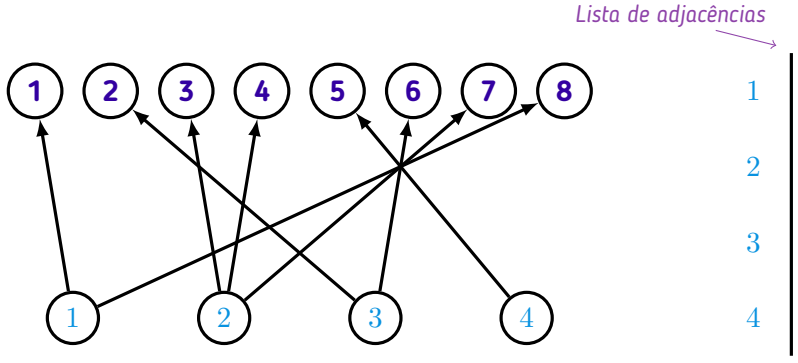
Solução



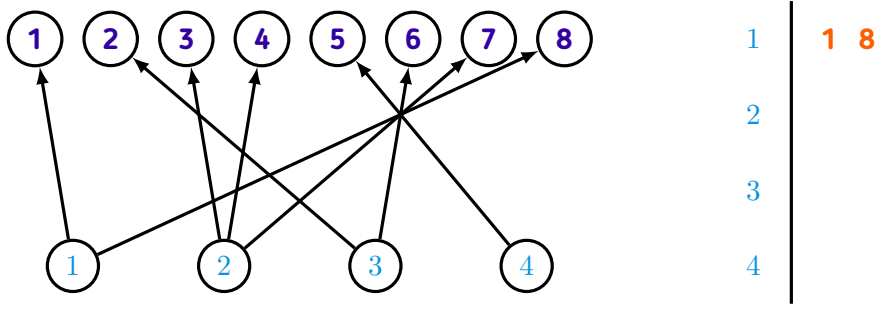
Solução



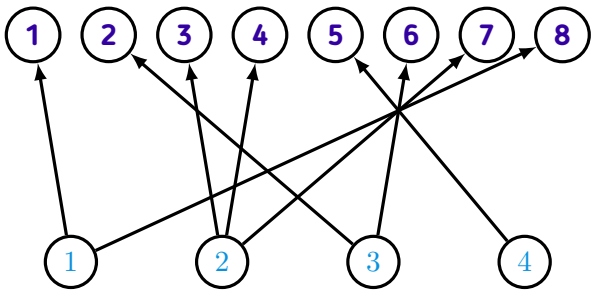
Solução



Solução



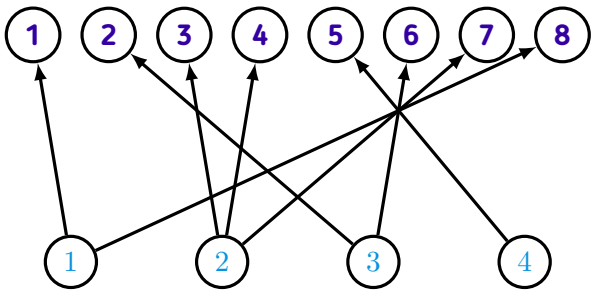
Solução



Lista de adjacências

1	1 8
2	3 4 7
3	
4	

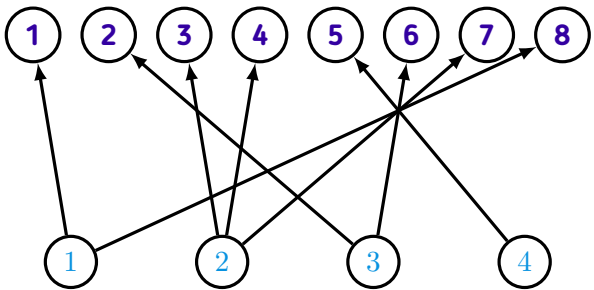
Solução



Lista de adjacências

1	1 8
2	3 4 7
3	2 6
4	

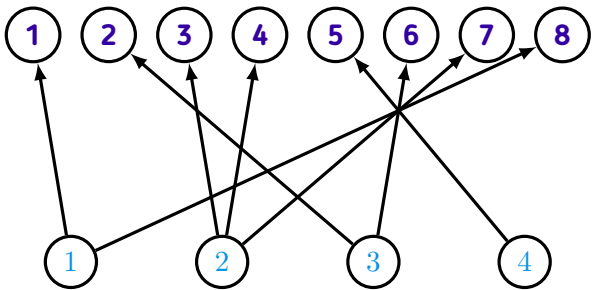
Solução



Lista de adjacências

1	1 8
2	3 4 7
3	2 6
4	5

Solução



Lista de adjacências

1	1 8
2	3 4 7
3	2 6
4	5

Consultas em $O(1)$

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

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

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

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

    for (int i = 1; i <= N; ++i)
        vs[xs[i]].push_back(i);

    vector<int> ans;
```

```
    for (auto [k, v] : qs)
        ans.push_back(k <= (int) vs[v].size() ? vs[v][k - 1] : 0);

    return ans;
}

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

    int N, M;

    while (cin >> N >> M)
    {
        vector<int> xs(N + 1);

        for (int i = 1; i <= N; ++i)
            cin >> xs[i];
```

```
        vector<ii> qs(M);

        for (int i = 0; i < M; ++i)
            cin >> qs[i].first >> qs[i].second;

        auto ans = solve(N, xs, qs);

        for (const auto& x : ans)
            cout << x << '\n';
    }

    return 0;
}
```