Travessia de Grafos

Aplicações: problemas resolvidos

Prof. Edson Alves - UnB/FGA

2019

Sumário

UVA 10505 – Montesco vs Capuleto

Problema

Romeo and Juliet have finally decided to get married. But preparing the wedding party will not be so easy, as it is well-known that their respective families –the Montesco and the Capuleto– are bloody enemies. In this problem you will have to decide which person to invite and which person not to invite, in order to prevent a slaugther.

We have a list of N people who can be invited to the party or not. For every person i, we have a list of his enemies: $E_1, E_2, \ldots, E_p$. The "enemy" relationship has the following properties:

Anti-transitive. If a is an enemy of b, and b is an enemy of c, then a is a friend of c. Also, the enemies of the friends of a are his enemies, and the friends of the friends of a are his friends.

Symmetrical. If a is an enemy of b, then b is an enemy of a (although it may not be indicated in his list of enemies).

Problema

One person will accept an invitation to the party if, and only if, he is invited, all his friends are invited and none of his enemies is invited. You have to find the maximum number of people that can be invited, so that all of them accept their invitation.

For instance, if $N = 5$, and we know that: 1 is enemy of 3, 2 is enemy of 1, and 4 is enemy of 5, then we could invite a maximum of 3 people. These people could be 2, 3 and 4, but for this problem we only want the number of people invited.

Entrada e saída

Input

The first line of the input file contains the number M of test cases in this file. A blank line follows this number, and a blank line is also used to separate test cases. The first line of each test case contains an integer N, indicating the number of people who have to be considered. You can assume that $N \leq 200$. For each of these N people, there is a line with his list of enemies. The first line contains the list of enemies of person 1, the second line contains the list of enemies of person 2, and so on. Each list of enemies starts with an integer p (the number of known enemies of that person) and then, there are p integers (the p enemies of that person). So, for example, if a person's enemies are 5 and 7, his list of enemies would be: '2 5 7'.

Entrada e saída

Output

For each test case, the output should consist of a single line containing an integer, the maximum number of people who can be invited, so that all of them accept their invitation.

Exemplo de entradas e saídas

Sample Input

```
3

5
1 3
1 1
0
1 5
0

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5

3
2 2 3
1 3
1 1
```

Sample Output

```
3
5
0
```

Solução com complexidade $O(M(V + E))$

- Observe que o grafo que corresponde à entrada do problema é direcionado e não necessariamente conectado
- Logo, para cada componente conectado, se as duas propriedades forem verificadas o componente será bipartido
- Neste caso, deve ser adicionado à resposta maior entre os números B e R, os quais correspondem ao número de nós azuis e vermelhos no componente, respectivamente
- Se o componente for bipartido, ele não contribuirá para a resposta final
- É preciso tomar, porém, alguns cuidados na implementação
- Como são múltiplos casos de teste, é preciso reiniciar as variáveis globais que foram utilizadas
- Além disso, na lista de inimigos da entrada podem aparecer pessoas cujo índice está fora do intervalo $[1, N]$, as quais devem ser desprezadas

Solução com complexidade $O(M(V+E))$

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

using namespace std;

const int MAX = 210, NONE = 0, BLUE = 1, RED = 2;
int color[MAX];
vector<int> adj[MAX];

int coloring(int s)
{
    int blue = 0, red = 0;
    bool bipartite = true;
    queue<int> q;

    q.push(s);
    color[s] = BLUE;
    ++blue;

    while (not q.empty())
    {
        auto u = q.front(); q.pop();
```

Solução com complexidade $O(M(V+E))$

```
23          for (const auto& v : adj[u])
24              if (color[v] == NONE)
25              {
26                  color[v] = 3 - color[u];
27                  color[v] == BLUE ? ++blue : ++red;
28                  q.push(v);
29              } else if (color[v] == color[u])
30                  bipartite = false;
31      }
32
33      return bipartite ? max(blue, red) : 0;
34  }
35
36  int solve(int N)
37  {
38      auto ans = 0;
39
40      for (int u = 1; u <= N; ++u)
41          if (color[u] == NONE)
42              ans += coloring(u);
43
```

Solução com complexidade $O(M(V+E))$

```
44      return ans;
45  }
46
47  int main()
48  {
49      ios::sync_with_stdio(false);
50
51      int M;
52      cin >> M;
53
54      while (M--)
55      {
56          int N, p, e;
57          cin >> N;
58
59          memset(color, 0, sizeof(color));
60
61          for (int u = 1; u <= N; ++u)
62              adj[u].clear();
63
```

Solução com complexidade $O(M(V+E))$

```
64          for (int u = 1; u <= N; ++u)
65          {
66              cin >> p;
67
68              while (p--)
69              {
70                  cin >> e;
71
72                  if (e < 1 or e > N)
73                      continue;
74
75                  adj[u].push_back(e);
76                  adj[e].push_back(u);
77              }
78          }
79
80          cout << solve(N) << '\n';
81      }
82
83      return 0;
84  }
```

Educational Codeforces Round 33 (Rated for Div. 2) – Problem C: Rumor

Problema

Vova promised himself that he would never play computer games... But recently Firestorm – a well-known game developing company – published their newest game, World of Farcraft, and it became really popular. Of course, Vova started playing it.

Now he tries to solve a quest. The task is to come to a settlement named Overcity and spread a rumor in it.

Vova knows that there are n characters in Overcity. Some characters are friends to each other, and they share information they got. Also Vova knows that he can bribe each character so he or she starts spreading the rumor; i-th character wants c_i gold in exchange for spreading the rumor. When a character hears the rumor, he tells it to all his friends, and they start spreading the rumor to their friends (for free), and so on.

Problema

The quest is finished when all n characters know the rumor. What is the minimum amount of gold Vova needs to spend in order to finish the quest?

Take a look at the notes if you think you haven't understood the problem completely.

Entrada e saída

Input

The first line contains two integer numbers n and m $(1 \leq n \leq 10^5, 0 \leq m \leq 10^5)$ – the number of characters in Overcity and the number of pairs of friends.

The second line contains n integer numbers c_i $(0 \leq c_i \leq 10^9)$ – the amount of gold i-th character asks to start spreading the rumor.

Then m lines follow, each containing a pair of numbers (x_i, y_i) which represent that characters x_i and y_i are friends $(1 \leq x_i, y_i \leq n, x_i \neq y_i)$. It is guaranteed that each pair is listed at most once.

Output

Print one number – the minimum amount of gold Vova has to spend in order to finish the quest.

Exemplo de entradas e saídas

Sample Input	Sample Output
5 2 2 5 3 4 8 1 4 4 5	10
10 0 1 2 3 4 5 6 7 8 9 10	55
10 5 1 6 2 7 3 8 4 9 5 10 1 2 3 4 5 6 7 8 9 10	15

Solução com complexidade $O(N + M)$

- Observe que este problema pode ser modelado como um grafo não-direcionado, onde cada vértice (e não aresta) tem um custo associado
- Como a pessoa é capaz de espalhar o rumo para todas as pessoas que estejam em seu componente conectado, basta escolher o menor custo em cada um dos componentes conectados
- Estes componentes podem ser identificados através de uma DFS ou uma BFS
- Atente ao fato de que, no pior caso, o custo mínimo pode extrapolar a capacidade de um número inteiro, sendo necessário utilizar o tipo `long long`

Solução AC com complexidade $O(N + M)$

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

using namespace std;
using ll = long long;
const int MAX { 100010 };
const ll oo { 2000000000000000000LL };

vector<int> adj[MAX];
bitset<MAX> visited;

void dfs(int u, const function<void(int)>& process)
{
    if (visited[u])
        return;

    visited[u] = true;
    process(u);

    for (const auto& v : adj[u])
        dfs(v, process);
}
```

Solução AC com complexidade $O(N + M)$

```
22
23 ll solve(int N, const vector<ll>& cs)
24 {
25     ll ans = 0;
26
27     for (int u = 1; u <= N; ++u)
28     {
29         if (visited[u])
30             continue;
31
32         ll cost = oo;
33
34         dfs(u, [&](int u) { cost = min(cost, cs[u]); });
35
36         ans += cost;
37     }
38
39     return ans;
40 }
41
```

Solução AC com complexidade $O(N + M)$

```
42 int main()
43 {
44     ios::sync_with_stdio(false);
45
46     int N, M;
47     cin >> N >> M;
48
49     vector<ll> cs(N + 1);
50
51     for (int u = 1; u <= N; ++u)
52         cin >> cs[u];
53
54     while (M--)
55     {
56         int x, y;
57         cin >> x >> y;
58
59         adj[x].push_back(y);
60         adj[y].push_back(x);
61     }
62
```

Solução AC com complexidade $O(N + M)$

```
63      auto ans = solve(N, cs);
64
65      cout << ans << '\n';
66
67      return 0;
68  }
```

Referências

1. UVA 10505 – Montesco vs Capuleto
2. Educational Codeforces Round 33 (Rated for Div. 2) – Problem C: Rumor