Caminhos mínimos

Algoritmo de Bellman-Ford: problemas resolvidos

Prof. Edson Alves - UnB/FGA

2019

Sumário

UVA 10959 – The Party, Part I

Problema

Don Giovanni likes to dance–especially with girls! And everyone else in the party enjoyed the dance, too. Getting a chance to dance with the host (that is Don Giovanni) is the greatest honour; failing that, dancing with someone who has danced with the host or will dance with the host is the second greatest honour. This can go further. Define the Giovanni number of a person as follows, at the time after the party is over and therefore who has danced with whom is completely known and fixed:

1. No one has a negative Giovanni number.
2. The Giovanni number of Don Giovanni is 0.
3. If a person p is not Don Giovanni himself, and has danced with someone with Giovanni number n, and has not danced with anyone with a Giovanni number smaller than n, then p has Giovanni number $n + 1$

Problema

4. If a person's Giovanni number cannot be determined from the above rules (he/she has not danced with anyone with a finite Giovanni number), his/her Giovanni number is ∞. Fortunately, you will not need this rule in this problem.

Your job is to write a program to compute Giovanni numbers.

Entrada e saída

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line has two numbers P and D; this means there are P persons in the party (including Don Giovanni) and D dancing couples ($P \leq 1000$ and $D \leq P(P-1)/2$). Then D lines follow, each containing two distinct persons, meaning the two persons has danced. Persons are represented by numbers between 0 and $P-1$; Don Giovanni is represented by 0.

As noted, we design the input so that you will not need the ∞ rule in computing Giovanni numbers.

Entrada e saída

We have made our best effort to eliminate duplications in listing the dancing couples, e.g., if there is a line "4 7" among the D lines, then this is the only occurrence of "4 7", and there is no occurrence of "7 4". But just in case you see a duplication, you can ignore it (the duplication, not the first occurrence).

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Output $P-1$ lines. Line i is the Giovanni number of person i, for $1 \leq i \leq P-1$. Note that it is $P-1$ because we skip Don Giovanni in the output.

Exemplo de entradas e saídas

Sample Input

```
1

5 6
0 1
0 2
3 2
2 4
4 3
1 2
```

Sample Output

```
1
1
2
2
```

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

- Observe que o número de Giovanni é, de fato, a distância do nó em questão até o nó zero, em número de arestas
- Esta distância pode ser computada por meio de uma BFS
- Como há garantia de conectividade do grafo de entrada (condição 4 do problema), uma única travessia é suficiente
- Como a complexidade da BFS é $O(V + E)$, a complexidade da solução é $O(T(V + E))$, onde T é o número de casos de teste

Solução com complexidade $O(T(V+E))$

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

using namespace std;

const int MAX { 1010 };

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

vector<int> solve(int N)
{
    vector<int> ans(N);
    queue<int> q;

    q.push(0);
    found[0] = true;
    ans[0] = 0;

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

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

```
22          q.pop();
23
24          for (const auto& v : adj[u])
25          {
26              if (not found[v])
27              {
28                  found[v] = true;
29                  ans[v] = ans[u] + 1;
30                  q.push(v);
31              }
32          }
33      }
34
35      return ans;
36  }
37
38  int main()
39  {
40      ios::sync_with_stdio(false);
41
42      int T;
```

Solução com complexidade $O(T(V+E))$

```
43      cin >> T;
44
45      for (int test = 0; test < T; ++test)
46      {
47          found.reset();
48
49          for (int i = 0; i < MAX; ++i)
50              adj[i].clear();
51
52          int P, D;
53          cin >> P >> D;
54
55          while (D--)
56          {
57              int x, y;
58              cin >> x >> y;
59
60              adj[x].push_back(y);
61              adj[y].push_back(x);
62          }
63
```

Solução com complexidade $O(T(V+E))$

```
64          auto ans = solve(P);
65
66          if (test)
67              cout << '\n';
68
69          for (int i = 1; i < P; ++i)
70              cout << ans[i] << '\n';
71      }
72
73      return 0;
74  }
```

Referências

1. UVA 10959 – The Party, Part I