Problema

O problema em questão é o problema E do AtCoder Begginer Contest 284 , disponível em https://atcoder.jp/contests/abc284/tasks/abc284_e.

Código

```cpp
#include <bits/stdc++.h>
using namespace std;

int paths = 0;
void dfs(vector<vector<int>>& adj, int source, bool visited[]){
    if (paths >= (int)1e6){
        return;
    }
    paths += 1;
    visited[source] = true;
    for (int u: adj[source]){
        if (!visited[u]){
            dfs(adj, u, visited);
        }
    }
    visited[source] = false; // so it can be used again
}

int main(){
    int n, m;
    cin >> n >> m;
    bool visited[n] = {false};
    vector<vector<int>> adj(n);

    int u, v;
    for (int i = 0; i < m; i++){
        cin >> u >> v;
        --u; --v;
        adj[u].emplace_back(v);
        adj[v].emplace_back(u);
    }

    dfs(adj, 0, visited);
    cout << paths;
}
```

Verificando a corretude do algoritmo

Podemos facilmente reduzir para o seguinte pseudocódigo (considerando um grafo não direcionado):

function dfs(u)

$paths \leftarrow paths + 1$

$visited[u] \leftarrow true$

for v in the adjacency of u

if $visited[v] = false$

dfs(v)

$visited[u] \leftarrow false$

dfs(s)

Demonstração

Para todo grafo G e para todo vértice s desse grafo, considere G_s o grafo obtido removendo/desconsiderando o vértice s e todas as suas arestas.

Lema 1: o vetor *visited*, que determina qual subgrafo o algoritmo deve considerar, é invariante em relação a aplicação da dfs: é o mesmo antes e depois da execução da função

Prova: a prova segue diretamente por indução. Suponha que vale para todo vértice de todo grafo com $m \leq n_0$ vértices, e seja G um grafo com $n_0 + 1$ vértices e s o vértice inicial sobre o qual chamamos a função. Seja $(v_i)_{1 \leq i \leq k}$ os vizinhos de s, na ordem em que serão visitados no algoritmo. Afirmamos que $dfs(v_i)$ não muda o estado do vetor *visited*, para cada $1 \leq i \leq k$.

Procedemos por indução novamente: o caso $i = 1$ segue pela hipótese de indução anterior, pois o $dfs(v_1)$ age sobre G_s, uma vez que s foi marcado como visitado anteriormente, e G_s possui n_0 vértices. Por outro lado, se $dfs(v_j)$ não altera *visited* para cada $1 \leq j \leq i < k$ então $dfs(v_{i+1})$ age sobre o mesmo subgrafo G_s que todas as outras execuções de dfs, e portanto por esse subgrafo possuir n_0 vértices o vetor *visited* não é alterado.

Portanto, concluímos que nenhuma mudança ao vetor *visited* persiste ao final do loop interno. Então, ao marcar $visited[s] = false$, ao final da dfs, voltamos ao grafo original G, provando o passo indutivo.

Teorema 1: Cada caminho partindo de s para qualquer outro vértice, que não passe por um mesmo vértice duas vezes, é contado exatamente uma vez nesse algoritmo.

Prova: note que um caminho desse tipo é uma sequência da forma $(s, a_1, \ldots, a_n)$ onde $n = 0$ (o caminho trivial, que contém apenas a origem) ou $a_1, \ldots, a_n \in$

$V \setminus \{s\}$ são vértices distintos do subgrafo obtido removendo s, e $(a_i, a_{i+1}) \in E$ (é uma aresta do grafo), $\forall 0 \leq i < n$. Note que foi considerado $a_0 = s$ (por conveniência).

Seja $f(s, G)$ a quantidade desses caminhos no grafo G, e G_s o subgrafo induzido por retirar o vértice s e todas as suas arestas. Concluímos que cada uma dessas sequências $a_1, \ldots, a_n$ é um caminho possível em G_s que começa em um vértice na adjacência de s, e reciprocamente todos os caminhos possíveis com 2 ou mais vértices é dessa forma. Em outras palavras:

$$f(s, G) = 1 + \sum_{(s,a) \in E} f(a, G_s)$$

Podemos provar agora o teorema por indução. O caso $|V| = 1$ é trivial, pois o algoritmo apenas incrementa *paths* de 0 para 1 e termina sua execução, devido à lista de adjacência vazia.

Suponha que o algoritmo vale para todo grafo com $m \leq n_0$ vértices, e que G é um grafo de $n_0 + 1$ vértices. Então, como s foi marcado como visitado no início do algoritmo, nenhuma aresta com s como extremo será considerada, e portanto podemos a primeira execução do loop interno como agindo sobre G_s. **Pelo lema 1, entretanto, dfs não modifica o vetor de visitados, e portanto segue que todas as chamadas da função no loop agem sobre G_s -** ou seja, são independentes entre si.

Porém, pela hipótese de indução, como G_s tem n_0 vértices, temos que $dfs(v)$ incrementa *paths* pela quantidade de caminhos de G_s que iniciam em v e não repetem vértices: que, por definição, é $f(v, G_s)$. Portanto, como *paths* é incrementado por 1 unidade logo no início da função, e como o loop ocorre sobre a adjacência de s, *paths* é incrementado exatamente $1 + \sum_{(s,a) \in E} f(a, G_s) = f(s, G)$ vezes, como queríamos demonstrar.