Isabella Carneiro
18/0019066

Grafos

Projeto de Algoritmos Turma 01 - 2023/1

Exercícios

- Desvio de Rota - Nível 4
- Babel - Nível 8
- Estradas escuras - Nível 5
- Detectando Pontes - Nível 6
- Componentes conexos - Nível 5

Isabella Carneiro
18/0019066

Beecrowd 1123

Desvio de Rota

Desvio de Rota

Maratona de Programação da SBC Brazil

Timelimit: 1

O sistema rodoviário de um país interliga todas as suas **N** cidades de modo que, a partir de uma cidade qualquer, é possível chegar a cada uma das outras cidades trafegando pelas estradas existentes. Cada estrada liga duas cidades distintas, tem mão dupla e um único posto de pedágio (o pedágio é pago nos dois sentidos de tráfego). As estradas não se intersectam a não ser nas cidades. Nenhum par de cidades é interligado por duas ou mais estradas.

A Transportadora Dias oferece um serviço de transporte de encomendas entre as cidades. Cada encomenda deve ser levada de uma cidade A para uma outra cidade B. A direção da Transportadora Dias define, para cada encomenda, uma rota de serviço, composta por **C** cidades e **C**−1 estradas: a primeira cidade da rota de serviço é a origem da encomenda, a última o destino da encomenda. A rota de serviço não passa duas vezes pela mesma cidade, e o veículo escolhido para fazer o transporte de uma encomenda pode trafegar apenas pela rota de serviço definida.

Certo dia, no entanto, o veículo que executava uma entrega quebrou e precisou ser levado para conserto em uma cidade que não está entre as cidades de sua rota de serviço. A direção da Transportadora Dias quer saber qual é o menor custo total, em termos de pedágio, para que o veículo entregue a encomenda na cidade destino, a partir da cidade em que foi consertado, mas com uma restrição adicional: se em algum momento o veículo passar por uma das cidades que compõem a sua rota de serviço, ele deve voltar a obedecer a rota de serviço.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém quatro inteiros **N**, **M**, **C** e **K** ($4 \leq N \leq 250$, $3 \leq M \leq N \times (N-1)/2$, $2 \leq C \leq N-1$ e $C \leq K \leq N-1$), representando, respectivamente, o número de cidades do país, o número de estradas, o número de cidades na rota de serviço e a cidade em que o veículo foi consertado. As cidades são identificadas por inteiros de 0 a **N**−1. A rota de serviço é 0, 1, ... , **C**−1, ou seja, a origem é 0, de 0 passa para 1, de 1 para 2 e assim por diante, até o destino **C**−1.

As **M** linhas seguintes descrevem o sistema rodoviário do país. Cada uma dessas linhas descreve uma estrada e contém três inteiros **U**, **V** e **P** ($0 \leq U, V \leq N-1$, $U \neq V$, $0 \leq P \leq 250$), indicando que há uma estrada interligando as cidades **U** e **V** com custo de pedágio **P**. O último caso de teste é seguido por uma linha contendo quatro zeros separados por espaço em branco.

Saída

Para cada caso de teste, o seu programa deve imprimir uma única linha, contendo um único inteiro **T**, o custo total mínimo necessário, em termos de pedágio, para que o veículo chegue ao destino.

Exemplo de Entrada	Exemplo de Saída
4 6 3 3 0 1 10 1 2 10 0 2 1 3 0 1 3 1 10 3 2 10 6 7 2 5 5 2 1 2 1 10 1 0 1 3 0 2 3 4 2 3 5 3 5 4 2 5 5 2 4 0 1 1 1 2 2 2 3 3 3 4 4 4 0 5 0 0 0 0	10 6 6

Isabella Carneiro
18/0019066

Beecrowd 1123

Desvio de Rota

CÓDIGO FONTE

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

#define INF INT_MAX

using namespace std;

int n, m, c, k;
int graph[1010][1010];
int cost[1010];

queue<int> q;

void initialize() {
    for(int i=0; i<=n; i++) {
        cost[i]= INF;
        for(int j=0; j<=n; j++) graph[i][j]=INF;
    }
}

int dijkstra(int start, int end) {
    cost[start] = 0;
    q.push(start);
    while(!q.empty()) {
        int i = q.front();
        q.pop();
        for(int j=0; j<n; j++) {
            if(graph[i][j] != INF && cost[j] > cost[i] + graph[i][j]) {
                cost[j] = cost[i] + graph[i][j];
                q.push(j);
            }
        }
    }
    return cost[end];
}

int main () {

    while(scanf("%d %d %d %d", &n, &m, &c, &k) == 4 && (n || m || c || k)) {

        initialize();

        for(int i=1; i<=m; i++) {
            int u, v, p;
            scanf("%d %d %d", &u, &v, &p);

            if(u>=c && v>=c) {
                graph[u][v]=p;
                graph[v][u]=p;
            }
            if(u>=c && v<c) graph[u][v]=p;
            if(u<c && v>=c) graph[v][u]=p;
            if(u<c && v<c && abs(u-v)==1) {
                graph[u][v]=p;
                graph[v][u]=p;
            }
        }

        printf("%d\n", dijkstra(k, c-1));
    }
    return 0;
}

```

Isabella Carneiro
18/0019066

beecrowd

AO VIVO
O que os outros estão resolvendo.

LISTAR
Liste todas as suas submissões.

TENTADO
Problemas ainda não resolvidos.

FAQS
Precisa de ajuda?

RESPOSTAS
O que isso significa?

CÓDIGO FONTE

EDITAR & ENVIAR

VISUALIZE O CÓDIGO FONTE DE SUAS SUBMISSÕES, JUNTO COM ALGUNS DETALHES EXTRAS.

SUBMISSÃO # 33066538

PROBLEMA:	1123 - Desvio de Rota
RESPOSTA:	Accepted
LINGUAGEM:	C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]
TEMPO:	0.000s
TAMANHO:	1,29 KB
MEMÓRIA:	-
SUBMISSÃO:	18/04/2023 22:05:02

CÓDIGO FONTE

```
1  #include<bits/stdc++.h>
2
3  #define INF INT_MAX
4
5  using namespace std;
6
7  int n, m, c, k;
8  int graph[1010][1010];
9  int cost[1010];
10
11 queue<int> q;
12
13 void initialize() {
14     for(int i=0; i<=n; i++) {
15         cost[i]= INF;
16         for(int j=0; j<=n; j++) graph[i][j]=INF;
17     }
```

Isabella Carneiro
18/0019066

Beecrowd 1085 Babel

beecrowd | 1085

Babel

Por Pedro Demasi Brasil

Timelimit: 5

Joãozinho e Mariazinha são dois irmãos que estão muito empolgados com suas aulas de idiomas, cada um está fazendo vários diferentes cursinhos. Ao chegar em casa comentam sobre gramática, vocabulário, cultura dos países etc. Numa dessas conversas perceberam que algumas palavras são comuns a mais de um idioma, mesmo que não necessariamente tenham o mesmo significado. Por exemplo, "amigo" existe em português e espanhol e tem o mesmo significado, enquanto que "date" é uma palavra comum entre francês e inglês mas que pode ter significados diferentes, uma vez que "date" também se refere a um encontro em inglês, além de "data" de calendário. Já "red" em espanhol se refere a uma rede, enquanto que em inglês se refere à cor vermelha. Outro exemplo seria "actual" que, em inglês significa algo real e, em espanhol, tem o significado de presente, atual (como em português).

Empolgados com essas descobertas, resolveram escrever num caderno todas as palavras em comum que conseguiram pensar, associando cada uma a um par de idiomas. Observador como é, Joãozinho propôs um desafio a Mariazinha: dados um idioma de origem e um de destino, escrever uma série de palavras sendo que a primeira necessariamente deveria pertencer ao idioma de origem e a última ao de destino. Duas palavras adjacentes nessa seqüência deveriam necessariamente pertencer a um mesmo idioma. Por exemplo, se o idioma de origem fosse português e o de destino francês, Mariazinha poderia escrever a seqüência amigo actual date (português/espanhol, espanhol/inglês, inglês/francês).

Para a surpresa de Joãozinho, Mariazinha conseguiu resolver o problema com muita facilidade. Irritado com o sucesso de sua irmã, ele resolveu complicar ainda mais o problema com duas restrições: Mariazinha deve encontrar a solução que tenha o menor comprimento da seqüência total não contando os espaços entre as palavras e duas palavras consecutivas não podem ter a mesma letra inicial.

Sendo assim, a solução anterior passa a ser inválida, pois "amigo" e "actual" têm a mesma letra inicial. é possível, porém, encontrar outra solução, que no caso seria amigo red date, cujo comprimento total é 12. Joãozinho fez uma extensa pesquisa na internet e compilou uma enorme lista de palavras e desafiou Mariazinha a resolver o problema. Como é possível que haja mais de uma solução, ele pediu para que ela apenas respondesse o comprimento da seqüência encontrada dadas as restrições ou se não há solução possível. Você seria capaz de ajudar Mariazinha?

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro M ($1 \leq M \leq 2000$), representando o total de palavras compiladas por Joãozinho. A segunda linha contém duas cadeias de caracteres distintas O e D, separadas por um espaço em branco, indicando os idiomas de origem e destino respectivamente. Cada uma das M linhas seguintes contém três cadeias de caracteres I1, I2 e P, separadas por um espaço em branco, representando dois idiomas e uma palavra comum entre ambos (I1 e I2 são sempre diferentes). Todas as cadeias de caracteres terão tamanho mínimo 1 e máximo 50 e conterão apenas letras minúsculas. Um mesmo par de idiomas pode ter várias palavras diferentes associadas a ele, porém uma mesma palavra P nunca será repetida.

O final da entrada é indicado por uma linha que contém apenas um zero.

Saída

Para cada caso de teste da entrada seu programa deve imprimir um único inteiro, o comprimento da menor seqüência que satisfaça as restrições de Joãozinho, ou `impossivel` (em minúsculas, sem acento) caso não seja possível.

Exemplo de Entrada	Exemplo de Saída
4 portugues frances ingles espanhol red espanhol portugues amigo frances ingles date espanhol ingles actual 4 portugues alemao ingles espanhol red espanhol portugues amigo frances ingles date espanhol ingles actual 6 portugues frances ingles espanhol red espanhol portugues amigo frances ingles date frances espanhol la portugues ingles a espanhol ingles actual 0	12 impossivel 5

Maratona de Programação da SBC 2008.

Isabella Carneiro
18/0019066

Beecrowd 1085 Babel

CÓDIGO FONTE

```cpp
#define INF 999999
using namespace std;

typedef long long ll;

int dist[6000][26];

int main(){
    ios_base :: sync_with_stdio(0); cin.tie(0);

    vector<vector<pair<int, string> > > adj_list;
    ll n;
    string start_city, end_city;
    map<string, int> city_index;
    int vertex_count;
    int i;
    string city1, city2, road_name;

    while(1){

        cin >> n;
        if(!n)
            break;

        vertex_count = 1;
        cin >> start_city >> end_city;

        city_index[start_city] = 1;
        city_index[end_city] = 2;
        vertex_count = 3;

        adj_list.assign(2 * n + 30, vector<pair<int, string> >());

        for(i=0; i<n; i++){
            cin >> city1 >> city2 >> road_name;
            int x, y;
            if(!city_index.count(city1)){
                city_index[city1] = vertex_count;
                x = vertex_count;
                vertex_count++;
            }
            else x = city_index[city1];
            if(!city_index.count(city2)){
                city_index[city2] = vertex_count;
                y = vertex_count;
                vertex_count++;
            }else y = city_index[city2];

            adj_list[x].push_back(make_pair(y, road_name));
            adj_list[y].push_back(make_pair(x, road_name));
        }

        for (int i = 0 ; i < 6000; ++i) for (int j = 0 ; j < 26; ++j)
            dist[i][j] = INF;

        for (int i = 0 ; i < 26; ++i)
            dist[3][i] = 0;

        priority_queue< pair<int, pair<int, char> >, vector<pair<int, pair<int, char> > >, greater
        pq.push(make_pair(0, make_pair(3, 'a')));

        while(!pq.empty()){
            pair<int, pair<int, char> > front = pq.top();
            pq.pop();
            int d = front.first;
            int v = front.second.first;
            char current_letter = front.second.second;
            for (int i = 0 ; i < adj_list[v].size(); ++i)
            {
                char letter = adj_list[v][i].second[0];
                if (v != 3 && letter == current_letter) continue;

                if (dist[adj_list[v][i].first][letter - 'a'] >= dist[v][current_letter - 'a'] +
                {
                    dist[adj_list[v][i].first][letter - 'a'] = dist[v][current_letter - 'a'] +
                    pq.push(make_pair(dist[adj_list[v][i].first][letter - 'a'], make_pair(adj_l
                }
            }
        }
        int mn = INF;
        for (int i = 0 ; i < 26; ++i)
            mn = min(mn, dist[2][i]);


        if (mn == INF) cout << "Impossivel\n";
        else cout << mn << '\n';

        city_index.clear();
        adj_list.clear();
    }
}
```

Isabella Carneiro
18/0019066

beecrowd

AO VIVO
O que os outros estão resolvendo.

LISTAR
Liste todas as suas submissões.

TENTADO
Problemas ainda não resolvidos.

FAQS
Precisa de ajuda?

RESPOSTAS
O que isso significa?

CÓDIGO FONTE

EDITAR & ENVIAR

VISUALIZE O CÓDIGO FONTE DE SUAS SUBMISSÕES, JUNTO COM ALGUNS DETALHES EXTRAS.

SUBMISSÃO # 33058679

PROBLEMA:	1085 - Babel
RESPOSTA:	**Accepted**
LINGUAGEM:	C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]
TEMPO:	0.145s
TAMANHO:	2,95 KB
MEMÓRIA:	-
SUBMISSÃO:	18/04/2023 18:29:40

CÓDIGO FONTE

```
#define INF 999999
using namespace std;

typedef long long ll;

int dist[4005][26];

int main(){
    ios_base :: sync_with_stdio(0); cin.tie(0);

    vector<vector<pair<int, string> > > adj_list;
    ll n;
    string start_city, end_city;
    map<string, int> city_index;
    int vertex count;
```

Isabella Carneiro
18/0019066

Beecrowd 1152

Estradas escuras

beecrowd | 1152

Estradas Escuras

University of Ulm Local Contest 🇩🇪 Alemanha

Timelimit: 3

Nestes dias se pensa muito em economia, mesmo em Byteland. Para reduzir custos operacionais, o governo de Byteland decidiu otimizar a iluminação das estradas. Até agora, todas as rotas eram iluminadas durante toda noite, o que custava 1 Dólar Byteland por metro a cada dia. Para economizar, eles decidiram não iluminar mais todas as estradas e desligar a iluminação de algumas delas. Para ter certeza que os habitantes de Byteland continuem a se sentirem seguros, eles querem otimizar o sistema de tal forma que após desligar a iluminação de algumas estradas à noite, sempre existirá algum caminho iluminado de qualquer junção de Byteland para qualquer outra junção.

Qual é a quantidade máxima de dinheiro que o governo de Byteland pode economizar, sem fazer os seus habitantes sentirem-se inseguros?

Entrada

A entrada contém vários casos de teste. Cada caso de teste inicia com dois números **m** ($1 \leq m \leq 200000$) e **n** ($m-1 \leq n \leq 200000$), que são o número de junções de Byteland e o número de estradas em Byteland, respectivamente. Seguem n conjuntos de três valores inteiros, **x**, **y** e **z**, especificando qual será a estrada bidirecional entre **x** e **y** com **z** metros ($0 \leq x, y < m$ e $x \neq y$).

A entrada termina com **m=n=0**. O grafo especificado em cada caso de teste é conectado. O tamanho total de todas as estradas em cada caso de teste é menor do que 2^{31}.

Saída

Para cada caso de teste imprima uma linha contendo a máxima quantidade diária de dólares de Byteland que o governo pode economizar.

Exemplo de Entrada	Exemplo de Saída
7 11 0 1 7 0 3 5 1 2 8 1 3 9 1 4 7 2 4 5 3 4 15 3 5 6 4 5 8 4 6 9 5 6 11 0 0	51

University of Ulm Local Contest 2009

Isabella Carneiro
18/0019066

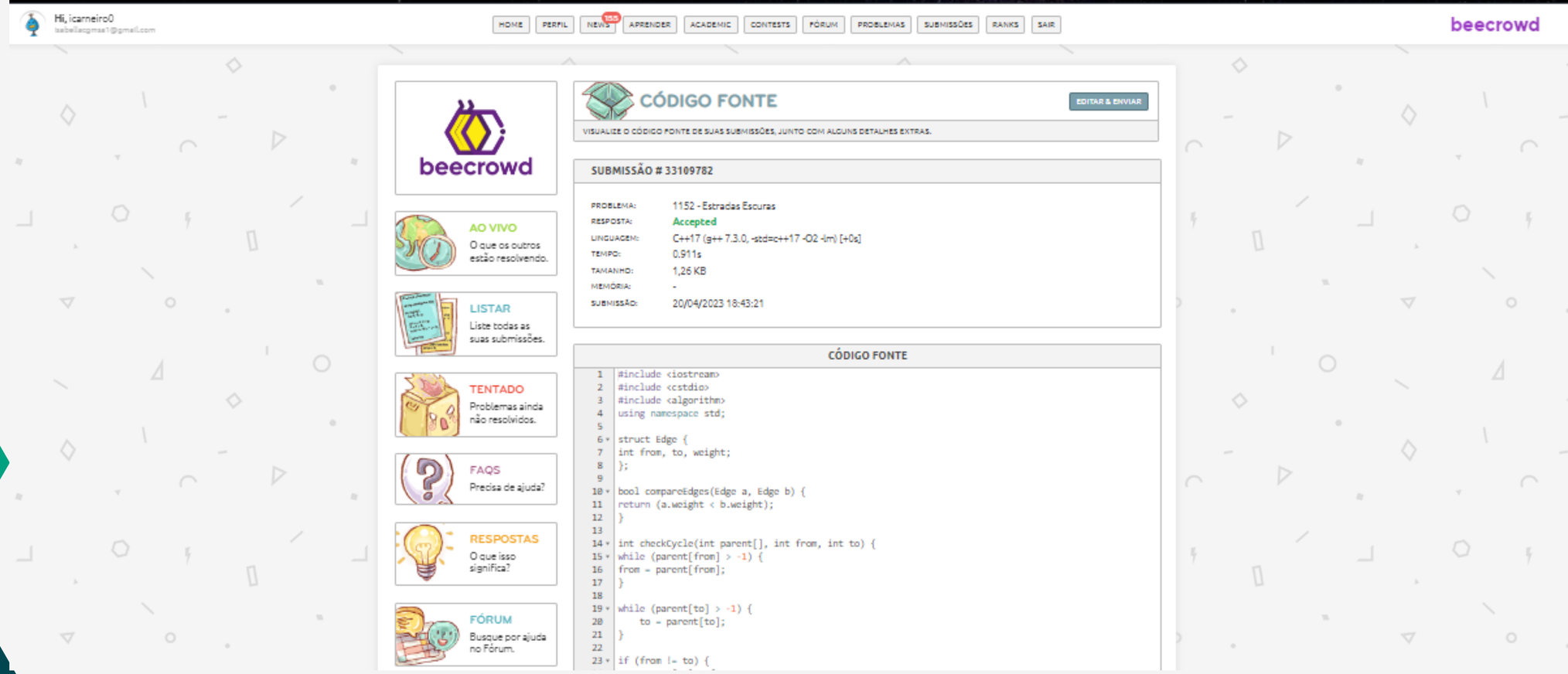
Beecrowd 1152

Estradas escuras

CÓDIGO FONTE

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;

struct Edge {
int from, to, weight;
};

bool compareEdges(Edge a, Edge b) {
return (a.weight < b.weight);
}

int checkCycle(int parent[], int from, int to) {
while (parent[from] > -1) {
from = parent[from];
}

while (parent[to] > -1) {
    to = parent[to];
}

if (from != to) {
    parent[to] = from;
    return 1;
}

return 0;
}

int main() {
int vertices, edges;
int parent[200000]; Edge edgeList[200000];

while (cin >> vertices >> edges) {
    if (vertices == 0 && edges == 0) {
        return 0;
    }

    int totalCost = 0, minCost = 0;

    for (int i = 0; i < vertices; i++) {
        parent[i] = -1;
    }

    for (int i = 0; i < edges; i++) {
        cin >> edgeList[i].from >> edgeList[i].to >> edgeList[i].weight;
        totalCost += edgeList[i].weight;
    }

    sort (edgeList, edgeList + edges, compareEdges);

    int i = 0, j = 1;
    while (j < vertices && i < edges) {
        if (checkCycle(parent, edgeList[i].from, edgeList[i].to)) {
            j++;
            minCost += edgeList[i].weight;
        }

        i++;
    }

    cout << totalCost - minCost << endl;
}

return 0;
}
```

Isabella Carneiro
18/0019066



Isabella Carneiro
18/0019066

Beecrowd 1790

Detectando Pontes

beecrowd | 1790

Detectando Pontes

Por Marcos Castro, Universidade Federal de São Paulo Brazil

Timelimit: 1

Pedrinho Ritchie mora em um país chamado Grafolândia. As cidades desse país estão interligadas através de pontes. Não existem cidades isoladas e nenhuma ponte é inserida mais de uma vez. Seu professor propôs um desafio: detectar a quantidade de pontes que não estão contidas em qualquer ciclo. Podemos dizer que um ciclo começa e termina na mesma cidade e nenhuma cidade se repete.

Pedrinho gosta muito de desafios de programação, mas precisa de sua ajuda para resolver esse problema, será que você consegue ajudá-lo?

Entrada

A entrada termina em **EOF**. Para cada caso de teste, a primeira linha contém dois inteiros positivos **C** e **P** que representam respectivamente a quantidade de cidades (**2** <= **C** <= **50**) e a quantidade de pontes (**1** <= **P** <= **1250**). Seguem-se **P** linhas onde cada linha contém dois inteiros positivos **X** e **Y** (indexados a partir do 1) indicando que há uma ponte interligando as cidades **X** e **Y**.

Saída

Seu programa deve imprimir a quantidade de pontes que não estão contidas em qualquer ciclo.

Exemplo de Entrada	Exemplo de Saída
4 3 1 2 2 3 3 4	3

Isabella Carneiro
18/0019066

Beecrowd 1790

Detectando Pontes

CÓDIGO FONTE

```cpp
#include <iostream>
#include <vector>
#include <cstring>  // for memset

using namespace std;

const int MAX_NODES = 60;  // Use const instead of #define for better type-checking
const int UNVISITED = -1;

int dfs_count, bridges, dfs_low[MAX_NODES], dfs_parent[MAX_NODES], dfs_num[MAX_NODES];
vector<vector<int>> adj_list;  // rename a to a more descriptive name

void dfs(int u)
{
    dfs_low[u] = dfs_num[u] = dfs_count++;

    for (int j = 0; j < (int)adj_list[u].size(); j++)
    {
        int v = adj_list[u][j];
        if (dfs_num[v] == UNVISITED)
        {
            dfs_parent[v] = u;
            dfs(v);

            if (dfs_low[v] > dfs_num[u])
                bridges++;
            dfs_low[u] = min(dfs_low[u], dfs_low[v]);
        }
        else if (v != dfs_parent[u])
        {
            dfs_low[u] = min(dfs_low[u], dfs_num[v]);
        }
    }
}

int main()
{
    int num_nodes, num_edges, x, y;

    while (cin >> num_nodes >> num_edges)
    {
        memset(dfs_low, UNVISITED, sizeof dfs_low);
        memset(dfs_parent, UNVISITED, sizeof dfs_parent);
        memset(dfs_num, UNVISITED, sizeof dfs_num);
        dfs_count = bridges = 0;

        adj_list.assign(num_nodes, vector<int>());

        for (int i = 0; i < num_edges; i++)
        {
            cin >> x >> y;
            x--, y--;  // shift to 0-based indexing
            adj_list[x].push_back(y), adj_list[y].push_back(x);
        }

        dfs(0);

        cout << bridges << endl;

        adj_list.clear();
    }

    return 0;
}
```

Isabella Carneiro
18/0019066

beecrowd

AO VIVO
O que os outros estão resolvendo.

LISTAR
Liste todas as suas submissões.

TENTADO
Problemas ainda não resolvidos.

FAQS
Precisa de ajuda?

SUBMISSÃO # 33114423

PROBLEMA:	1790 - Detectando Pontes
RESPOSTA:	Accepted
LINGUAGEM:	C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]
TEMPO:	0.000s
TAMANHO:	1,55 KB
MEMÓRIA:	-
SUBMISSÃO:	20/04/2023 22:27:04

CÓDIGO FONTE

```
1  #include <iostream>
2  #include <vector>
3  #include <cstring>  // for memset
4
5  using namespace std;
6
7  const int MAX_NODES = 60;  // Use const instead of #define for better type-checking
8  const int UNVISITED = -1;
9
```

Isabella Carneiro
18/0019066

Beecrowd 1082 Componentes conexos

beecrowd | 1082

Componentes Conexos

Por Neilor Tonin, URI Brasil

Timelimit: 1

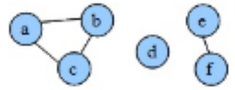
Com base nestas três definições:

Grafo conexo: Um grafo ***G(V,A)*** é conexo se para cada par de nodos u e v existe um caminho entre u e v. Um grafo com apenas um componente é um grafo conexo.

Grafo desconexo: Um grafo ***G(V,A)*** é desconexo se ele for formado por 2 ou mais componentes conexos.

Componente conexo: Componentes conexos de um grafo são os subgrafos conexos deste grafo.

O grafo a seguir possui 3 componentes conexos. O primeiro é formado pelos nodos **a**,**b**,**c**. O segundo é formado unicamente pelo nodo **d** e o terceiro componente é formado pelos nodos **e**,**f**.



Com base nestes conceitos, onde cada entrada fornecida que tem a identificação de cada um dos vértices, arestas e as ligações entre os vértices através destas arestas, liste cada um dos componentes conexos que existem no grafo, segundo a entrada fornecida.

Entrada

A primeira linha do arquivo de entrada contém um valor inteiro **N** que representa a quantidade de casos de teste que vem a seguir. Cada caso de teste contém dois valores **V** e **E** que são, respectivamente, a quantidade de **V**értices e arestas (**E**dges) do grafo. Seguem **E** linhas na sequência, cada uma delas representando uma das arestas que ligam tais vértices. Cada vértice é representado por uma letra minúscula do alfabeto (*'a'-'z'*), ou seja, cada grafo pode ter no máximo 26 vértices. Cada grafo tem no mínimo 1 componente conexo.

Obs: Os vértices de cada caso de teste sempre iniciam no 'a'. Isso significa que um caso de teste que tem 3 vértices, tem obrigatoriamente os vértices 'a', 'b' e 'c'.

Saída

Para cada caso de teste da entrada, deve ser apresentada uma mensagem ***Case #n:*** onde ***n*** indica o número do caso de teste (conforme exemplo abaixo). Segue a listagem dos vértices de cada segmento, um segmento por linha, separados por vírgula (inclusive com uma vírgula no final da linha). Finalizando o caso de teste, deve ser apresentada uma mensagem indicando a quantidade de componentes conexos do grafo (em inglês). Todo caso de teste deve ter uma linha em branco no final, inclusive o último caso de teste.

Obs: os nodos devem sempre ser apresentados em ordem crescente e se há caminho de a até b significa que há caminho de b até a.

Exemplo de Entrada	Exemplo de Saída
3 3 1 a c 10 10 a b a c a g b c c g e d d f h i i j j h 6 4 a b b c c a e f	Case #1: a,c, b, 2 connected components Case #2: a,b,c,g, d,e,f, h,i,j, 3 connected components Case #3: a,b,c, d, e,f, 3 connected components

Isabella Carneiro
18/0019066

Beecrowd 1082 Componentes conexos

CÓDIGO FONTE

```c
#include <stdio.h>

int graph[26][26], component[26];

void initialize(int vertices);
void dfs(int vertex, int vertices, int current_component);


void initialize(int vertices)
{
    int i, j;

    for (i = 0; i < vertices; i++)
    {
        component[i] = 0;

        for (j = 0; j < vertices; j++)
        {
            graph[i][j] = 0;
        }
    }
}

void dfs(int vertex, int vertices, int current_component)
{
    int j;

    if (component[vertex])
    {
        return;
    }

    component[vertex] = current_component;

    for (j = 0; j < vertices; j++)
    {
        if (vertex != j && graph[vertex][j])
        {
            dfs(j, vertices, current_component);
        }
    }
}


int main(void)
{
    int num_cases, i, j, case_num, vertices, edges, current_component;
    char a, b;

    scanf("%d", &num_cases);

    for (case_num = 1; case_num <= num_cases; case_num++)
    {
        scanf("%d %d", &vertices, &edges);
        current_component = 0;
        initialize(vertices);

        for (i = 0; i < edges; i++)
        {
            scanf(" %c %c", &a, &b);
            a -= 97;
            b -= 97;
            graph[a][b] = graph[b][a] = 1;
        }

        printf("Case #%d:\n", case_num);

        for (i = 0; i < vertices; i++)
        {
            if (!component[i])
            {
                current_component++;
                dfs(i, vertices, current_component);

                for (j = 0; j < vertices; j++)
                {
                    if (component[j] == current_component)
                    {
                        printf("%c,", j + 97);
                    }
                }
                printf("\n");
            }
        }
        printf("%d connected components\n\n", current_component);
    }

    return 0;
}
```

Isabella Carneiro
18/0019066

beecrowd

AO VIVO
O que os outros estão resolvendo.

LISTAR
Liste todas as suas submissões.

TENTADO
Problemas ainda não resolvidos.

FAQS
Precisa de ajuda?

CÓDIGO FONTE

EDITAR & ENVIAR

VISUALIZE O CÓDIGO FONTE DE SUAS SUBMISSÕES, JUNTO COM ALGUNS DETALHES EXTRAS.

SUBMISSÃO # 33174177

PROBLEMA:	1082 - Componentes Conexos
RESPOSTA:	**Accepted**
LINGUAGEM:	C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]
TEMPO:	0.005s
TAMANHO:	1,83 KB
MEMÓRIA:	-
SUBMISSÃO:	24/04/2023 17:23:28

CÓDIGO FONTE

```
1  #include <stdio.h>
2
3  int graph[30][30], component[30];
4
5  void initialize(int vertices);
6  void dfs(int vertex, int vertices, int current_component);
7
8
9  void initialize(int vertices)
10 {
11     int i, j;
```

Obrigada!