Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Ciência da Computação
Algoritmos em Grafos
Profª. Raquel Mini

Trabalho Prático

Data de Entrega: 17/05/2020 no verde
Valor: 20 pontos

Resolva os problemas listados abaixo **individualmente**, usando a linguagem C++. Ao final, submeta na plataforma verde.

Movimentos do Cavalo

Pedro está fazendo uma pesquisa sobre o problema do movimento do cavalo em um tabuleiro de xadrez e incumbiu você da tarefa de encontrar o menor conjunto de movimentos possíveis, podendo sair de qualquer quadrado a e podendo chegar em qualquer quadrado b dentro do tabuleiro, sendo que a e b são quadrados diferentes. Ele pensa que a parte mais difícil do problema é determinar o menor número de movimentos do cavalo entre 2 quadrados fornecidos e que uma vez que você está comprometido com esta tarefa, encontrar a sequência de movimentos entre estes 2 quadrados será uma tarefa muito fácil.

É claro que você sabe que o movimento é vice versa. Portanto você deve fornecer a Pedro um programa que resolva esta questão.

Seu trabalho então será escrever um programa que, pegando dois quadrados a e b como entrada, determine o número de movimentos para encontrar a rota mais curta de a até b.

Entrada

A entrada contém um ou mais casos de teste. Cada caso de teste consiste de uma linha contendo dois quadrados separados por um espaço. Um quadrado será uma string consistindo de uma letra (a-h) representando a coluna e um dígito (1-8) representando a linha do tabuleiro de xadrez (veja figura acima).

Saída

para cada caso de teste imprima uma linha dizendo "`To get from xx to yy takes n knight moves.`". No caso *xx* é a origem, yy é o destino e *n* é a quantidade de movimentos necessários para ir de *xx* até *yy*.

Exemplo de Entrada	Exemplo de Saída
e2 e4 a1 b2 b2 c3 a1 h8 a1 h7 h8 a1 b1 c3 f6 f6	To get from e2 to e4 takes 2 knight moves. To get from a1 to b2 takes 4 knight moves. To get from b2 to c3 takes 2 knight moves. To get from a1 to h8 takes 6 knight moves. To get from a1 to h7 takes 5 knight moves. To get from h8 to a1 takes 6 knight moves. To get from b1 to c3 takes 1 knight moves. To get from f6 to f6 takes 0 knight moves.

Ir e Vir

Numa certa cidade há **N** interseções ligadas por ruas de mão única e ruas com mão dupla de direção. É uma cidade moderna, de forma que muitas ruas atravessam túneis ou têm viadutos. Evidentemente é necessário que se possa viajar entre quaisquer duas intersecções, isto é, dadas duas intersecções **V** e **W**, deve ser possível viajar de **V** para **W** e de **W** para **V**.

Sua tarefa é escrever um programa que leia a descrição do sistema de tráfego de uma cidade e determine se o requisito de conexidade é satisfeito ou não.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois números inteiros **N** e **M**, separados por um espaço em branco, indicando respectivamente o número de intersecções ($2 \leq \mathbf{N} \leq 2000$) e o número de ruas ($2 \leq \mathbf{M} \leq \mathbf{N}(\mathbf{N}-1)/2$). O caso de teste tem ainda mais **M** linhas, que contêm, cada uma, uma descrição de cada uma das **M** ruas. A descrição consiste de três inteiros **V**, **W** e **P**, separados por um espaço em branco, onde **V** e **W** são identificadores distintos de intersecções ($1 \leq \mathbf{V}, \mathbf{W} \leq \mathbf{N}$, $\mathbf{V} \neq \mathbf{W}$) e **P** pode ser 1 ou 2; se **P** = 1 então a rua é de mão única, e vai de **V** para **W**; se **P** = 2 então a rua é de mão dupla, liga **V** e **W**. Não existe duas ruas ligando as mesmas intersecções.

O ultimo caso de teste é seguido por uma linha que contém apenas dois números zero separados por um espaço em branco.

Saída

Para cada caso de teste seu programa deve imprimir uma linha contendo um inteiro G, onde G é igual a 1 se o requisito de conexidade está satisfeito, ou G é igual a 0, caso contrário.

Exemplo de Entrada	Exemplo de Saída
4 5 1 2 1 1 3 2 2 4 1 3 4 1 4 1 2 3 2 1 2 2 1 3 2 3 2 1 2 2 1 3 1 4 2 1 2 2 3 4 2 0 0	1 1 0 0

Componentes Conexos

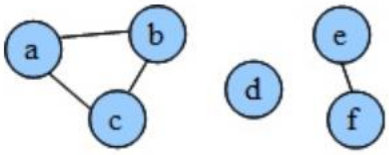
Com base nestas três definições:

Grafo conexo: Um grafo **G(V,A)** é conexo se para cada par de vértices *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 vértices *a, b, c*. O segundo é formado unicamente pelo vértice *d* e o terceiro componente é formado pelos vértices *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 virgula 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

Resgate em Queda Livre

Ó, meu Deus! Um grupo de pessoas está caindo em queda livre! Elas saltaram todas exatamente ao mesmo tempo de vários aviões que estavam exatamente à mesma altura. A intenção era realizar o maior e mais belo salto sincronizado da História. No entanto, o malévolo Loki, para se deleitar com a insignificância humana, sabotara os paraquedas, e agora a única esperança está numa ação conjunta do Homem-Aranha com o Homem-de-Ferro. Como ambos são muito nerds, notaram que as pessoas estavam caindo todas num mesmo plano paralelo ao solo, a despeito da resistência do ar e de outros fatores. Então, bolaram um plano infalível. Primeiro, o aracnídeo unirá todas as pessoas através de cabos de teia entre elas. Uma vez que não haja pessoa que não esteja conectada ao grupo, o playboy poderá eletromagnetizar o grupo todo e, segurando na mão de uma apenas das pessoas do grupo, pousar todas elas em segurança.

Mas não há muito tempo para divagações. O Homem-Aranha precisa agir rápido, o que no caso dele significa gastar o mínimo possível de teia. Para tanto, o Homem-de-Ferro em seu screen projetou numa malha cartesiana o plano em que as pessoas estão, usando o centímetro como unidade de medida, e obteve as coordenadas de cada pessoa na malha. Agora, J.A.R.V.I.S. está computando qual o mínimo necessário de teia de que o Homem-Aranha precisará. Dependendo da resposta, o Homem-de-Ferro não esperará pelo garoto e improvisará alguma outra peripécia.

Entrada

A entrada é constituída por vários casos de teste. A primeira linha de entrada contém um inteiro **C** que determina a quantidade de casos de teste. Cada caso de teste começa com um inteiro positivo **n** (**n** ≤

500), o qual representa o número de pessoas no grupo. Seguem, então, **n** linhas, cada uma designando uma pessoa do grupo pelas suas coordenadas **x** e **y** na malha ($0 \leq \mathbf{x}, \mathbf{y} \leq 10^4$).

Saída

Para cada caso de teste, seu programa deverá imprimir uma linha contendo o valor com precisão de duas casas decimais correspondente ao comprimento mínimo de teia, em metros, necessário para se conectarem todas as pessoas do grupo. Atente para que o separador das casas decimais seja **.** (ponto), não **,** (vírgula).

Exemplo de Entrada	Exemplo de Saída
2 5 0 0 0 100 100 200 200 400 300 300 4 1 5 1 4 2 3 3 2	6.06 0.04

Colorindo Grafos

Seja G um grafo simples com N vértices coloridos e M arestas. Nós desejamos saber se é possível adicionar exatamente P arestas em G de tal forma que o grafo resultante seja simples, conexo e nenhuma aresta conecte dois vértices da mesma cor.

Entrada

A entrada contém múltiplos casos testes. A primeira linha contém a quantidade de casos testes **T** (**T** < 70). Cada caso teste começa com 4 inteiros na seguinte ordem: o número de vértices **N** (1 <= **N** <= 10^3), o número de arestas no grafo original **M** (0 <= **M** <= 10^5), o número de arestas a serem inseridas **P** (0 <= **P** <= 10^6) e o número de cores **K** (1 <= **K** <= 10^3). A linha seguinte contém **N** números $\mathbf{X_i}$ indicando a cor do i-ésimo vértice (1 <= $\mathbf{X_i}$ <= **K**). As **M** seguintes linhas contém um par de inteiros ($\mathbf{V_i}$, $\mathbf{V_j}$) indicando a presença de uma aresta entre os vértices $\mathbf{V_i}$ e $\mathbf{V_j}$. (1 <= $\mathbf{V_i}$,$\mathbf{V_j}$ <= **N**).

Saída

Para cada caso teste, imprima uma única lina com "**Y**" (sem aspas) se é possível construir tal grafo ou "**N**" caso contrário.

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