TRABALHO PRÁTICO
TEORIA DE GRAFOS 2018/2

Professor: Rafael Santos Coelho
2 de novembro de 2018

Descrição do problema
Uma importante universidade federal brasileira está passando por um período de grande reforma infraestrutural e, por isso, sua intranet (a rede interna que liga seus principais laboratórios de pesquisa) será modernizada. Entretanto, por escassez de verba, a reitoria teve que fazer uma dura escolha:

1) ou manter os roteadores que dão suporte à intranet (que funcionam bem, embora alguns poucos deles já estejam tecnologicamente ultrapassados) e trocar o cabeamento da rede por fibra ótica,

2) ou manter o atual cabeamento (que vem apresentando defeitos há anos) e instalar novos roteadores.

A opção escolhida, por uma questão de custo *versus* benefício, foi a primeira, visto que

1) os roteadores (mesmo que alguns sejam um tanto quanto antigos) ainda conseguem operar normalmente e

2) a fibra ótica é um material resistente a falhas e bastante durável.

O problema é o seguinte: para minimizar custos, é essencial projetar a nova topologia da rede universitária de forma econômica, isto é, utilizando a menor quantidade total possível de fibra ótica (porque fibra ótica custa caro) capaz de interconectar os diferentes dispositivos de rede espalhados pelos campi dessa universidade. Como dito anteriormente, alguns desses dispositivos são mais limitados e só possuem uma entrada, isto é, só podem se conectar a um dispositivo apenas na rede. **Sabendo a quantidade (em quilômetros) de fibra ótica necessária para conectar cada par de dispositivos e tendo em mãos a lista dos dispositivos limitados (os que só podem se ligar a exatamente um outro dispositivo na rede), sua tarefa é projetar um programa de computador que calcule a melhor topologia (a que minimiza a quantidade total de fibra ótica a ser gasta) para rede universitária.**

Entrada
Os dados de entrada **deverão ser lidos do teclado exatamente no formato descrito a seguir**. O *input* começa com uma linha contendo um número **N (um inteiro positivo)** que denota o número de campi da universidade. Nas linhas seguintes aparecem dados referentes a cada um desses N campi. Para cada campus, primeiramente é informada em uma linha a quantidade **D (um inteiro positivo)** de dispositivos (numerados de 1 até D) que compõem a intranet daquele campi. Em seguida, nas próximas D linhas, é informada uma **matriz simétrica M**, **com D linhas e D colunas, em que M(i,j)** (isto é, a entrada da matriz M localizada na i-ésima linha e j-ésima coluna, para 1 <= i <= D e 1 <= j <= D) **representa a quantidade de fibra ótica necessária (em quilômetros) para ligar o dispositivo i ao dispositivo j**. Depois disso, na próxima linha, é informada a quantidade **L (um inteiro positivo menor do que D)** de dispositivos limitados da intranet daquele campus. Por fim, na linha seguinte, **é informada a lista dos L dispositivos limitados**, **cada um identificado pelo seu número** (ou seja, nessa linha haverá uma sequência de L inteiros positivos finalizada com um \n, onde cada inteiro nessa sequência, que não seja o último, vai estar separado do próximo por exatamente um espaço em branco).

Saída
Para cada campus da universidade, seu algoritmo **deve imprimir na tela do computador** (**exatamente da forma descrita a seguir**) uma mensagem (**sem as aspas, é claro**) iniciando com "Campus p*: "* **(ou seja, a palavra Campus, seguida de exatamente um espaço em branco, seguido de um número inteiro positivo p, seguido do caracter :, seguido de exatamente um espaço em branco)**, onde p é o número do campus em questão (**começando a contagem a partir de 1 e seguindo a ordem natural sucessivamente**). Após isso, você deve informar, **na mesma linha** (logo após o esbaço em branco que vem depois do caracter :), **o custo total da topologia encontrada** (a quantidade total em quilômetros de fibra ótima que deve ser gasta para interconectar todos os dispositivos do campus em questão, considerando as restrições dos dispositivos limitados) e, por fim, você deve finalizar a linha com \n.

Observações importantes
Você pode assumir que, em todos os testes realizados, as seguintes restrições serão respeitadas:

- N <= 100
- D <= 1000
- As entradas da matriz M serão inteiros positivos <= 200
- L < D

Exemplo de entrada (comentários escritos abaixo em rosa explicam cada linha)
2 **<= Número total de campi da universidade**
8 **<= Número de dispositivos do campus 1**
0 9 7 1 2 8 1 5 **<= Primeira linha da matriz M do campus 1**
9 0 13 3 2 10 1 19 **<= Segunda linha da matriz M do campus 1**
7 13 0 4 5 6 7 18 **<= Terceira linha da matriz M do campus 1**
1 3 4 0 60 61 63 5 **<= Quarta linha da matriz M do campus 1**
2 2 5 60 0 11 12 13 **<= Quinta linha da matriz M do campus 1**
8 10 6 61 11 0 5 6 **<= Sexta linha da matriz M do campus 1**
1 1 7 63 12 5 0 5 **<= Sétima linha da matriz M do campus 1**
5 19 18 5 13 6 5 0 **<= Oitava linha da matriz M do campus 1**
3 **<= Número de dispositivos limitados do campus 1**
6 7 8 **<= Lista dos dispositivos limitados do campus 1**
4 **<= Número de dispositivos do campus 2**
0 19 77 6 **<= Primeira linha da matriz M do campus 2**
19 0 31 32 **<= Segunda linha da matriz M do campus 2**
77 31 0 8 **<= Terceira linha da matriz M do campus 2**
6 32 8 0 **<= Quarta linha da matriz M do campus 2**
1 **<= Número de dispositivos limitados do campus 2**
2 **<= Lista dos dispositivos limitados do campus 2**

Exemplo de saída (relativo ao exemplo de entrada exibido acima)
Campus 1: 21
Campus 2: 33

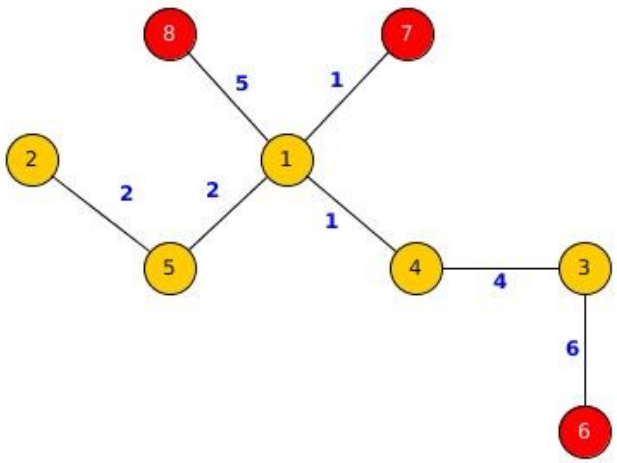

Figura 1: exemplo de uma solução ótima para o campus 1 da entrada especificada acima; vértices vermelhos indicam os dispositivos limitados; números em azul indicam as quantidades (em quilômetros) de fibra ótica para conectar pares de vértices

Sobre a entrega e sobre os critérios de correção do trabalho

- Este trabalho somente poderá ser feito em grupos de no máximo 6 integrantes da mesma turma. Trabalhos feitos por grupos com mais de 6 integrantes ou com integrantes de turmas diferentes não serão corrigidos e, portanto, ficarão com nota 0.
- As únicas linguagens de programação permitidas para resolução deste trabalho são Python (versão 2 ou versão 3), C, C++ e Java. Trabalhos feitos em qualquer outra linguagem que não seja alguma dessas não serão corrigidos e, portanto, ficarão com nota 0. Se seu grupo fez o trabalho em Python versão 3, deixe isso explícito nos comentários do código.
- Os códigos deverão ser entregues via Moodle até as 23h59m59s do dia 02/12/2018 (2 de dezembro de 2018, um domingo). Apenas um membro de cada grupo deve submeter o trabalho em nome do grupo inteiro. A submissão deve estar compactada em um arquivo .zip e apenas o arquivo compactado deverá ser enviado via Moodle. Submissões atrasadas (não importa quão pequeno seja o atraso) não serão aceitas e, portanto, ficarão com nota 0.
- Dentro do arquivo .zip submetido via Moodle o grupo deve manter apenas uma pasta chamada trab e dentro dessa pasta o grupo deverá colocar os arquivos fonte do código, um Makefile para automatizar a compilação do código (somente se o grupo tiver feito o trabalho em C, C++ ou Java) e um arquivo chamado membros.txt contendo os nomes completos dos membros do grupo. Submissões feitas sem um Makefile ou com um Makefile com bugs (de novo, somente caso o grupo tenha feito o trabalho em C, C++ ou Java) serão penalizadas e terão um desconto na nota de 10%.
- A nota dada pelo trabalho vai depender de três fatores: corretude do código (ou seja, se seu algoritmo encontra as respostas corretas), desempenho do código (ou seja, se seu algoritmo consome pouco tempo e pouco espaço de memória para resolver o problema e não tem vazamentos de memória) e legibilidade e organização do código (código bem documentado, bem indentado, bem modularizado e estruturado e de fácil compreensão). O primeiro fator terá peso de 50%, o segundo terá peso de 40% e o terceiro terá peso de 10%.
- Para avaliar a corretude do código, será executado um “lote” com 10 casos de teste (ou seja, uma entrada com N = 10), cada um com 1000 dispositivos (D = 1000). Para cada resposta incorreta, serão descontados da nota 5%.
- Para avaliar o desempenho do código, serão executados 4 “lotes” de testes, cada lote contendo 100 casos de teste (ou seja, 4 entradas diferentes, cada uma com N = 100). Será fixado um limite máximo de tempo de execução por lote de 7 segundos (ou seja, para cada entrada com N = 100, seu código precisa resolver os 100 casos de teste em no máximo 7 segundos). Para cada lote de testes em que o código estourar o limite máximo de tempo de execução serão descontados 10% na nota.
- Qualquer tentativa de fraude (plágios) identificada poderá acarretar nota 0 a todos os envolvidos (isto é, plagiados e plagiadores).
- Códigos com erros de compilação ficarão com nota 0 automaticamente. Códigos com erros de execução (por exemplo, falha de segmentação) serão penalizados. Para cada erro de execução encontrado, serão descontados da nota 20%.
- Para resolver o trabalho, é necessário usar pelo menos um dos algoritmos de grafos ensinados em sala (possivelmente com adaptações, é claro), isto é, BFS (busca em largura), DFS (busca em profundidade), algoritmo de Prim ou algoritmo de Dijkstra. Trabalhos feitos baseados em outros algoritmos terão um desconto na nota de 20%.
- Para resolver o trabalho, é permitido usar estruturas de dados/algoritmos genéricos (por exemplo, algoritmos de ordenação, algoritmos de busca, filas, pilhas, vetores, listas, matrizes, dicionários, tabelas hash, conjuntos, strings, árvores, etc) “pré-prontos” nativos das bibliotecas padrão linguagens em questão (ou seja, estruturas de dados oriundas de bibliotecas externas às linguagens não poderão ser usadas).

- Para resolver o trabalho, não é permitido usar estruturas de dados/algoritmos de manipulação de grafos “pré-prontos” (nativos ou externos) das linguagens em questão. Em outras palavras, todas as estruturas de dados de representação de grafos e todos os algoritmos de grafos usados para resolver o trabalho deverão ser implementados “do zero”.
- Atenção: os códigos serão compilados (usando o Makefile disponibilizado, se for o caso) e testados em ambiente Linux (Ubuntu). Portanto, antes de fazer a submissão final do seu grupo, certifiquem-se de que o código compila e executa corretamente em Linux. Sugestão: tentem compilar e testar o código de vocês nos computadores dos laboratórios de graduação do INF.
- Códigos com erros de leitura de dados (isto é, códigos que não respeitam a formatação dos dados de entrada especificada neste documento) ficarão com nota 0 automaticamente. Códigos com erros de formatação de saída (por exemplo, espaços em branco sobrando, linhas a mais, etc) serão penalizados (o tamanho do desconto a ser dado na nota vai depender de caso a caso).
- Os testes serão automatizados por meio de scripts. O input será passado ao código compilado por meio do comando < (redirecionamento de input) do terminal do Linux. É essencial que vocês prestem atenção no formato de entrada e de saída dos dados. A checagem das respostas corretas será feita usando o utilitário diff do Linux. Para cada caso de teste, será feito um diff entre o arquivo com as respostas esperadas e o arquivo contendo as respostas geradas pelo seu código.
- Atenção: após submeter o arquivo .zip no Moodle, certifiquem-se de que o .zip não está corrompido. Façam o seguinte: após o upload do .zip no Moodle, baixem o .zip, descompactem em algum diretório, testem o Makefile (se for o caso) e testem o código rodando alguns casos de teste. Submissões cujos .zip’s estiverem corrompidos serão desconsideradas e ficarão com nota 0.