TRABALHO PRÁTICO
TEORIA DE GRAFOS 2019/1

Professor: Rafael Santos Coelho
10 de junho de 2019

Descrição do problema

Neste trabalho, vamos estudar o chamado *problema dos caminhos mínimos de fonte única com pesos nos vértices e nas arestas.* Considere um grafo simples e conexo G com pesos inteiros positivos nos vértices e nas arestas. O custo de um caminho aberto em G é definido como a soma dos pesos das arestas e dos vértices (inclusive os vértices extremos) que fazem parte desse caminho. Dito isso, **dado um grafo simples e conexo G com pesos inteiros positivos nos vértices e nas arestas, um vértice fonte s de G e uma lista de vértices de destino distintos em G (todos diferentes de s), digamos t_1, t_2, …, t_k, encontre os custos de caminhos mínimos em G ligando s a t_1, t_2, … e t_k.**

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 vértices do grafo G **(os vértices do grafo serão numerados de 1 a N, sendo que o vértice 1 sempre será o vértice fonte s)**. Na linha seguinte será informada a lista dos vértices de destino (cada vértice será referenciado pelo seu respectivo número), sendo que nenhum vértice do grafo aparece mais de uma vez nessa lista (e o vértice s nunca aparece nessa lista). Na linha seguinte aparecem os pesos de cada vértice do grafo (começando pelo peso do vértice 1, depois peso do vértice 2 e assim por diante). Em seguida, nas próximas N linhas, é informada uma matriz quadrada simétrica M com N linhas e N 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 <= N e 1 <= j <= N), **se for positivo**, **representa o peso da aresta em G com extremos i e j e se for negativo representa que não há aresta em G ligando o vértice i ao vértice j**.

Saída

Seu algoritmo **deve imprimir na tela do computador** (**exatamente da forma descrita a seguir**) os custos dos caminhos de custo mínimo em G começando em s terminando nos vértices de destino de acordo com a ordem com que esses vértices aparecem no *input*. Pares de custos adjacentes devem ser separados por exatamente um espaço em branco e a lista de custos deve ser finalizada com um \n.

Observações importantes

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

- N <= 100
- pesos dos vértices e das arestas <= 1000

Exemplo de entrada 1 (comentários escritos abaixo em rosa explicam cada linha)
5 **<= Número de vértices do grafo**
2 4 5 **<= Lista de vértices de destino**
1 1 100 2 2 **<= Pesos dos vértices**
-1 2 1 3 -1 **<= Pesos das arestas que saem do vértice 1 (-1's indicam "não-arestas")**
2 -1 3 -1 -1 **<= Pesos das arestas que saem do vértice 2 (-1's indicam "não-arestas")**
1 3 -1 1 2 **<= Pesos das arestas que saem do vértice 3 (-1's indicam "não-arestas")**
3 -1 1 -1 4 **<= Pesos das arestas que saem do vértice 4 (-1's indicam "não-arestas")**
-1 -1 2 4 -1 **<= Pesos das arestas que saem do vértice 5 (-1's indicam "não-arestas")**

Exemplo de saída (relativo ao exemplo de entrada 1)
4 6 12

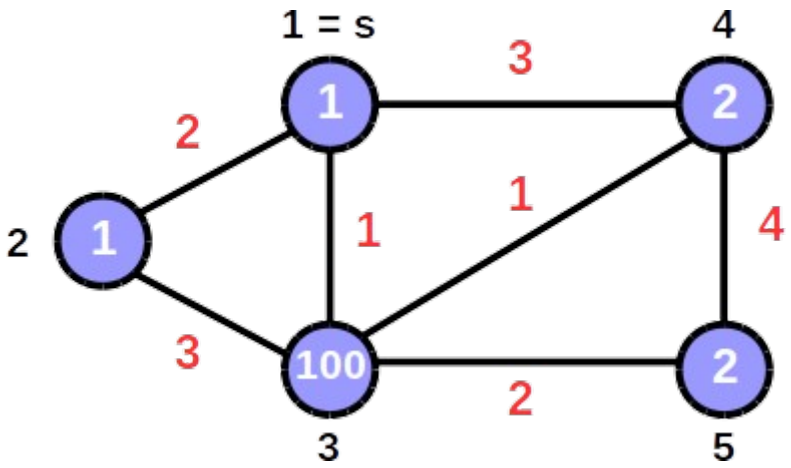

Figura 1: grafo especificado pelo exemplo de entrada 1 mostrado acima. Os números em vermelho indicam os pesos das arestas presentes no grafo. Os números em branco dentro dos vértices indicam os pesos dos vértices. Os números na cor preta em cima/ao lado/abaixo dos vértices indicam a numeração dos vértices (lembre-se de que o vértice 1 sempre é definido como o vértiec fonte s).

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 01/07/2019 (01 de julho de 2019, uma segunda-feira). 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 e numeros de cartão 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, cada um com um grafo com 100 vértices (N = 100). Para cada resposta incorreta, serão descontados da nota 5%.
- Para avaliar o desempenho do código será fixado um limite máximo de tempo de execução por teste de 4 segundos (ou seja, seu código precisa resolver cada teste em no máximo 4 segundos). Para cada teste em que o código estourar o limite máximo de tempo de execução serão descontados 4% 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 10%.
- 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 ao 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.