Segundo Exercício-Programa

Prof. Luciano Antonio Digiampietri

Prazo máximo para a entrega: 16/06/2024

1 O Problema do Caixeiro Viajante

O problema do caixeiro viajante visa a resolver o problema de um vendedor que precisa passar por um conjunto de cidades para vender suas mercadorias e retornar à cidade original. A meta deste EP é encontrar o ciclo de menor custo que passe por todas as cidades do mapa.

Neste EP trabalharemos com mapas via grafos direcionados e ponderados representados usando listas de adjacências em que cada vértice representa uma cidade e cada aresta direcionada e ponderada representará a distância entre duas cidades (por ser um grafo direcionado, é possível haver uma ligação da cidade A para a cidade B sem que haja uma ligação da cidade B para a cidade A).

Alguns conceitos relacionados:

Caminho Hamiltoniano: um caminho hamiltoniano é um caminho em um grafo que passa por todos os vértices uma única vez.

Ciclo Hamiltoniano: um ciclo hamiltoniano é um ciclo em um grafo que passa por todos os vértices uma única vez e retorna ao vértice original.

Caixeiro Viajante: a solução do problema do caixeiro viajante consiste em encontrar o ciclo hamiltoniano de menor custo (ou peso).

Detalhamento:

Para este EP, utilizaremos a mesma estrutura de dados usada nas aulas para representar grafos direcionados e ponderados com listas de adjacências.

O código fornecido para este EP já possui várias funções implementadas de gerenciamento básico de grafos, bem como para realizar alguns testes no EP.

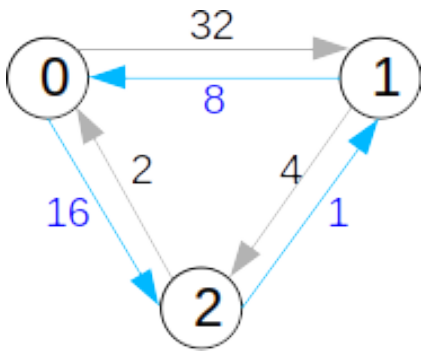
O problema do Caixeiro Viajante tipicamente é resolvido como um problema de Tentativa e Erro do "tipo" que tenta encontrar a melhor solução (no caso, a solução de menor custo). Nos seguintes links estão os slides e duas videoaulas relacionadas a Tentativa e Erro. Observe que nos últimos slides são apresentadas as estruturas gerais para a construção de algoritmos de Tentativa e Erro para encontrar uma única solução e para encontrar a melhor solução.
Slides: `https://www.each.usp.br/digiampietri/ACH2002/aula11.pdf`
Videoaula parte 1: `https://www.youtube.com/watch?v=OwF5paAESOs`
Videoaula parte 2: `https://www.youtube.com/watch?v=iWJIqquY2O0`

Você deverá implementar/completar uma função, podendo, se julgar conveniente, implementar funções adicionais/auxiliares.

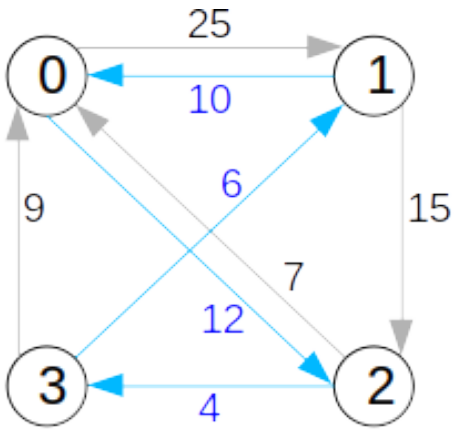
- *void caixeiroAux(Grafo* g, int atual, int numVisitados)*: função (potencialmente recursiva) que revolve o problema do Caixeiro Viajante considerando o mapa representado pelo grafo apontado por g, sendo *atual* o indicador do vértice atual e $numVisitados$ o número de vértices já visitados. Esta função é chamada, inicialmente, pela função *caixeiroViajante* que já está implementada e não deve ser modificada. Caso não haja nenhuma solução para o problema, a função *caixeiroAux* não deverá alterar o valor da variável global $melhorValor$. Caso contrário, a função deverá preencher o arranjo *cicloAtual* com a ordem das cidades a serem visitadas (partindo da cidade 0 [zero]) e atualizar o valor de $melhorValor$ com o custo do melhor ciclo encontrado (ciclo com menor custo). Estas duas variáveis (e algumas outras existentes no código) são globais, inicializadas pela função já implementada *caixeiroViajante* e podem/devem ser atualizadas pelo seu código.

Exemplos: a seguir são apresentados dois exemplos presentes no código fonte do EP.

```
Imprimindo grafo (vertices: 3; arestas: 6).
[ 0] ->  1 (32.00) ->  2 (16.00)
[ 1] ->  0 ( 8.00) ->  2 ( 4.00)
[ 2] ->  0 ( 2.00) ->  1 ( 1.00)
Custo encontrado: 25.00
Ordem das cidades: 0 2 1
```



```
Imprimindo grafo (vertices: 4; arestas: 8).
[ 0] ->  1 (25.00) ->  2 (12.00)
[ 1] ->  0 (10.00) ->  2 (15.00)
[ 2] ->  0 ( 7.00) ->  3 ( 4.00)
[ 3] ->  0 ( 9.00) ->  1 ( 6.00)
Custo encontrado: 32.00
Ordem das cidades: 0 2 3 1
```



2 Material a Ser Entregue

Um arquivo, denominado *NUSP*.c (sendo NUSP o seu número USP, por exemplo: 123456789.c), contendo seu código, incluindo todas as funções relacionadas ao EP. Para sua conveniência, `completeERenomeie.c` será fornecido, cabendo a você então completá-lo e renomeá-lo para a submissão.

Atenção!

1. Não modifique as assinaturas das funções já implementadas e/ou que você deverá completar!
2. Para avaliação, as diferentes funções serão invocadas diretamente (individualmente ou em conjunto com outras funções). Em especial, qualquer código dentro da função `main()` será ignorado.

3 Entrega

A entrega será feita única e exclusivamente via sistema e-Disciplinas, até a data final marcada. Deverá ser postado no sistema um arquivo .c, tendo como nome seu número USP:

`seuNumeroUSP.c  (por exemplo, 123456789.c)`

Não esqueça de preencher o cabeçalho constante do arquivo .c, com seu nome, número USP, turma etc.

A responsabilidade da postagem é exclusivamente sua. Por isso, submeta e certifique-se de que o arquivo submetido é o correto (fazendo seu download, por exemplo). Problemas referentes ao uso do sistema devem ser resolvidos com antecedência.

4 Avaliação

A nota atribuída ao EP será baseada nas funcionalidades solicitadas, porém não esqueça de se atentar aos seguintes aspectos:

1. Documentação: se há comentários explicando o que se faz nos passos mais importantes e para que serve o programa (tanto a função quanto o programa em que está inserida);
2. Apresentação visual: se o código está legível, identado etc;
3. Corretude: se o programa funciona.

Além disso, algumas observações pertinentes ao trabalho, que influenciam em sua nota, são:

- Este exercício-programa deve ser elaborado individualmente;
- Não será tolerado plágio;
- Exercícios com erro de sintaxe (ou seja, erros de compilação), receberão nota ZERO.