



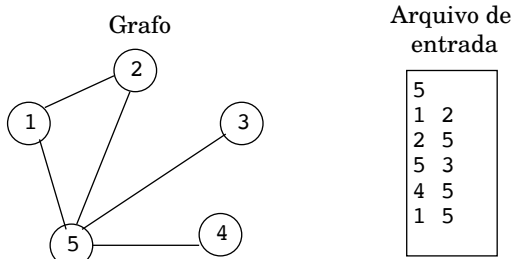
Especificação do Trabalho 1: Funcionalidades básicas em grafos simples (9 pontos)

Este documento traz a especificação do Trabalho 1, atividade avaliativa do tipo “Trabalho Prático”, e serve de base para todas as atividades de implementação cobradas ao longo do curso da disciplina DCC059 - Teoria dos Grafos no semestre letivo 2021-1.

Objeto: desenvolver um Tipo Abstrato de Dados - TAD ou uma Classe que represente grafos simples, orientados e não orientados, ponderados e não ponderados (nos vértices e arestas) e implemente o conjunto de funcionalidades apresentados a seguir, detalhadas em sala de aula. O desenvolvimento dos algoritmos aqui requeridos contribui como base para a resolução das questões das avaliações teóricas.

Orientações:

- seu TAD ou Classe deve ser capaz de representar grafos utilizando **lista de adjacência**;
- o código deve ser desenvolvido em linguagem C ou C++ (padrão ANSI);
- Além do atendimento às funcionalidades, alguns dos elementos avaliados são a clareza e a organização do código (nomes de funções e variáveis, comentários que indiquem o propósito das principais funções e procedimentos, inclusive explicando o que são os parâmetros e o retorno, em caso de função);
- o programa principal que usará o TAD ou a Classe Grafo deve ler os dados do grafo de entrada (direcionados ou não direcionados, ponderados ou não ponderados) a partir de arquivo texto. O formato do arquivo dependerá da origem dos dados de entrada. Assim, cabe ao grupo ler o arquivo README que explica a semântica do arquivo de entrada ou, caso não haja este arquivo, o grupo deve ler o detalhamento do mesmo na fonte de dados, implementando conforme o caso. Entenda-se por formato do arquivo a estrutura em que os dados do grafo aparecem no texto do mesmo. Por exemplo, para algumas instâncias teste, o arquivo pode ser apresentado como segue no exemplo, onde se tem um grafo simples, não ponderado nos vértices e nas arestas, e não direcionado. Neste exemplo, a primeira linha indica o número de vértices e as demais linhas indicam as arestas.



- a informação sobre o tipo de grafo, se direcionado ou não direcionado, deve ser passada ao programa por parâmetro via linha de comando, sendo 0 (zero) para não direcionado e 1 (um) para grafos direcionados. Note que o TAD ou Classe deve prever a existência de duas formas de inclusão das adjacências, conforme cada caso.

- a informação sobre arestas ponderadas ou não deve ser passada ao programa por parâmetro via linha de comando, sendo 0 (zero) para não ponderado nas arestas e 1 (um) para grafos com peso nas arestas.
- a informação sobre a existência de pesos nos vértices deve ser passada ao programa por parâmetro via linha de comando, sendo 0 (zero) quando o grafo não é ponderado nos vértices e 1 (um) para grafos com pesos nos vértices.
- o nome do arquivo a ser lido deve ser informado ao programa via teclado para a função main (utilizar int main (int argc, char ** argv) para passar ao programa todas as informações necessárias ao seu funcionamento;
- cada grupo enviará um unico e-mail contendo **APENAS** os arquivos fonte (extensão c, cc, cpp e h) e os arquivos de entrada utilizados (quando o professor não os tiver encaminhado antes);
- o padrão para compilação a ser utilizado (ambiente Linux ou IOS) será g++ *.c* - o execGrupoX. Onde "GrupoX" indica a qual grupo o trabalho se refere;
- o padrão para a execução a ser utilizado pelo professor será a linha abaixo, executada em ambiente Linux ou IOS:
- ./execGrupoX <arquivo_entrada> <arquivo_saida> <Opc_Direc> <Opc_Peso_Aresta> <Opc_Peso_Nos>, onde < arquivo_entrada> é o nome do arquivo que contém as informações do grafo, < arquivo_saida> é o arquivo onde será gravado o grafo armazenado na memória ao término do;
- o grupo deve enviar um **ÚNICO** arquivo compactado de nome Trabalho_GrupoX.zip para o e-mail indicado em sala de aula, cujo assunto da mensagem seja "Trabalho Grafos Grupo X – 2020-3", onde X indica o número do grupo.

Nota: no arquivo zip deve ter incluso **APENAS** os arquivos com extensão c, cc, cpp ou h e os arquivos das instâncias usadas no experimento.

Funcionalidades:

- O programa deve apresentar em **tela** a saída para as seguintes funcionalidades definidas nas saídas conforme os parâmetros:
 - a) **Parâmetro:** um Id de um vértice de um **grafo direcionado**;
Saída: o fecho transitivo direto deste vértice.
 - b) **Parâmetro:** um Id de um vértice de um **grafo direcionado**;
Saída: o fecho transitivo indireto deste vértice.
 - c) **Parâmetro:** dois IDs de vértices do grafo;
Saída: o caminho mínimo entre estes dois vértices usando algoritmo de Dijkstra;
 - d) **Parâmetro:** dois IDs de vértices do grafo;
Saída: o caminho mínimo entre estes dois vértices usando algoritmo de Floyd;
 - e) **Parâmetro:** um subconjunto X de vértices do grafo;
Saída: uma Árvore Geradora Mínima sobre o subgrafo vértice-induzido por X usando o algoritmo de Prim;
 - f) **Parâmetro:** um subconjunto X de vértices do grafo;
Saída: uma Árvore Geradora Mínima sobre o subgrafo vértice-induzido por X usando o algoritmo de Kruskal;
 - g) **Parâmetro:** um ID de vértice;
Saída: a árvore dada pela ordem de caminhamento em profundidade a partir de nó dado parâmetro, destacando as arestas de retorno;
 - h) **Parâmetro:** o próprio grafo acíclico direcionado D;
Saída: uma ordenação topológica em D ou a informação de que D não é um grafo acíclico direcionado.

- Todas as saídas das funcionalidades acima devem ser apresentadas na tela. Ao final da execução;
- Com exceção do item h, todas as saídas devem ser dadas na linguagem “dot” para geração de grafos para o software Graphviz (vide <http://www.inf.ufes.br/~pdcosta/ensino/2018-2-estruturas-de-dados/material/Tutorial%20Graphviz.pdf>).

Perguntas Frequentes

1. *Quantos membros um grupo pode ter?*
O trabalho pode ter no **máximo 4 pessoas**. Mas, caso haja interesse de fazer o trabalho individualmente, você deve estar ciente de que, uma vez informado ao professor, não poderá integrar um outro grupo depois.
2. *Se eu fizer alguma parte do trabalho individualmente ou em grupo, terei que fazer o trabalho 2 correspondente ao assunto da Unidade 6 da mesma forma (mesmo grupo)?*
Sim, pois a avaliação do trabalho considera a composição de todas as funcionalidades requeridas.
3. *O projeto poderá ter mais de um arquivo fonte (c, cc, cpp e h)?*
Pode (e, para boa organização do código, deve). Como usual, a especificação do trabalho descreve somente a interface a ser implementada. A organização do projeto é livre.
4. *O que será levado em conta na correção?*
Na correção serão levados em conta (entre outros) os seguintes elementos.
 1. Interação com o professor;
 2. Conformidade com a especificação.
 3. Correção da implementação.
 4. **Eficiência** da implementação.
 5. Organização e clareza do código (nomes de funções e variáveis, comentários etc).
5. *Por que a especificação de como o programa será executado é importante?*
Porque o trabalho entregue poderá ser pré-processado por um *script* que depende de a especificação de entrega ser corretamente observada.
6. *O que acontece se a especificação de execução do programa não for corretamente observada?*
Seu trabalho só será corrigido quando houver tempo de fazer manualmente o pré-processamento. **Por isso, haverá desconto na sua nota, proporcional ao trabalho de pré-processamento que tenha que ser feito manualmente.**
7. *Meu trabalho tem um bug. O que vai acontecer com minha nota?*
Haverá algum desconto, dependendo da gravidade do *bug*. Se o problema afetar alguma das funcionalidades requeridas, o desconto será proporcional ao que estiver faltando.
8. *Meu código não compila. Posso enviar assim mesmo?*
Não serão avaliados trabalhos com erros de compilação. Por isso a importância de se usar apenas funções do padrão Ansi.
9. *Tenho outra pergunta/dúvida a respeito do trabalho.*
Procure os professores ou os tutores para tirar suas dúvidas no horário de atendimento ou durante as aulas.

Tenham todos um bom semestre e aproveitem a disciplina!

Profs. Stênio Sã e Luciana Brugiolo