UNIVERSIDADE FEDERAL DE LAVRAS

INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

GCC218 – ALGORITMOS EM GRAFOS

TURMAS 10A/14A – SEMESTRE 2024/01

Prof.: Mayron Moreira.

Enunciado de Trabalho Prático adaptado da descrição do prof. Douglas Aquino.

Objetivo

Implementar algoritmos para análise de propriedades de grafos, avançando gradualmente à medida que a disciplina progrida.

Descrição

- Os alunos serão desafiados a implementar um sistema modular para análise de propriedades de grafos. A implementação será progressiva, com novas funcionalidades adicionadas conforme o avanço da disciplina. A estrutura básica consistirá em um menu interativo onde o usuário poderá selecionar, via linha de comando, a propriedade do grafo que desejam analisar.
- Os alunos poderão implementar nas linguagens definidas na especificação do trabalho no Beecrowd, a saber: C++, Python, Java, Javascript.
- **Os alunos possuem tem até às 11h da manhã do dia 21/08/2024 para submeter a versão final do trabalho, com as funcionalidades indicadas no enunciado.**
- A avaliação se dará em cima da porcentagem de funções requeridas implementadas.
    - O aluno poderá escolher qualquer estrutura de dados em grafos, desde que a escolha não impacte em ineficiência do código em relação ao tempo limite de resposta.
    - Para cada função requerida, pode-se escolher qual o método/algoritmo a ser utilizado, desde que a escolha não impacte em ineficiência do código em relação ao tempo limite de resposta.
    - Funcionalidade extra: dados os resultados de cada execução do programa, criar um visualizador que apresente a figura do grafo resultante. Nota extra: até 13% do valor do trabalho (3,9 pontos).
        - **O visualizador não será enviado no Beecrowd. Ele estará contido no repositório público do *GitHub*.**
    - A atividade poderá ser realizada em trios, duplas ou sozinho.
    - Desenvolva o seu código no *GitHub*, para que se familiarizem com esse sistema de controle de versões. No entanto, o código final deve ser obrigatoriamente submetido no Beecrowd.
    - Alguns *guidelines* para realizar seus *commits* no *GitHub* podem ser vistos aqui: https://github.com/git-guides/git-commit
    - Apenas **um arquivo fonte** deve ser submetido no **Beecrowd**.

O nome de cada membro do grupo, assim como sua matrícula e turma, deverão constar no cabeçalho das submissões. Caso isso não ocorra, haverá uma penalidade de até 3 pontos na nota final do trabalho (10%) para todos os membros do grupo.

- Apenas um membro do grupo submeterá a resposta no Beecrowd.
  - Caso deseje enviar o visualizador para a avaliação, apenas um membro do grupo (o mesmo que enviará o código no Beecrowd) deverá postar o link no repositório GitHub em uma atividade presente no Campus Virtual.

Casos de teste

- O padrão de leitura dos grafos seguirá o seguinte.

  #vertices #arestas

  direcionado_ou_nao_direcionado

  id_aresta u v p

  ...

  Considerando um grafo G = (V,E) não-direcionado e não-ponderado, V = {0,1,2,3} e E = {(0,1), (1,2), (1,3), (2,3)}, tem-se:

  3 4

  nao_direcionado

  0 0 1 1

  1 1 2 1

  2 1 3 1

  3 2 3 1

- Qualquer tentativa de plágio na submissão a ser avaliada implicará em nota zero para todos os membros dos grupos envolvidos.
- Cuidado com o "plágio indireto": um grupo A copia a resposta de alguma fonte da web, e outro grupo B, que não teve contato com o grupo A, copia do mesmo repositório.
- Funções a serem testadas:

  <u>Verificar</u>

  1. Conexo
  2. Bipartido
  3. Euleriano

4. Possui ciclo

Listar

5. Componentes conexas

6. Componentes fortemente conexas

7. Um caminho Euleriano (priorizando a ordem lexicográfica dos vértices)

8. Um caminho Hamiltoniano (priorizando a ordem lexicográfica dos vértices)

9. Vértices de articulação

10. Arestas ponte

Gerar

11. Árvore de profundidade (priorizando a ordem lexicográfica dos vértices; 0 é a origem)

12. Árvore de largura (priorizando a ordem lexicográfica dos vértices; 0 é a origem)

13. Árvore geradora mínima (priorizando a ordem lexicográfica dos vértices ou arestas)

14. Ordem topológica (Esta função não fica disponível em grafos não direcionado; deve-se priorizar a ordem lexicográfica dos vértices)

15. Valor do caminho mínimo entre dois vértices (priorizando a ordem lexicográfica dos vértices, dependendo do algoritmo; 0 é a origem; n-1 é o destino)

16. Valor do fluxo máximo (Esta função não fica disponível em grafos não direcionado; deve-se priorizar a ordem lexicográfica dos vértices; 0 é o vértice origem; n-1 é o vértice destino)

17. Fecho transitivo (Esta função não fica disponível em grafos não direcionado; deve-se priorizar a ordem lexicográfica dos vértices; 0 é o vértice escolhido)

- Exemplo de saída para cada função, em relação ao grafo apresentado.

  1. 1

  2. 0

  3. 1

  4. 1

  5. 0 1 2 3

  6. -1

7. 0 1 2 3

8. 0 1 2 3

9. -1

10. -1

11. 0,1 1,3 3,2

12. 0,1 0,2 1,3

13. 0,1 0,2 1,3

14. -1

15. 2

16. -1

17. -1

- Exemplo de entrada/saída de um caso de teste.

  Entrada:

  5 1 3 6 8 17

  Saída:

  1

  1

  -1

  0 1 2 3

  -1

Critérios de Avaliação

- Nota do Beecrowd: 60%
  - Códigos com *TimeLimit* serão atribuídos até 50% desse quesito.
  - Códigos com *Presentation Error* serão considerados como corretos.
- Documentação e Comentários (10%):
  - O código deve ser bem comentado, facilitando o entendimento do funcionamento dos algoritmos.
- Arguição Oral (30%):
  - Capacidade de demonstrar o simulador de forma clara e organizada.
  - Domínio dos conceitos teóricos abordados no trabalho.
  - O docente fará perguntas aos membros dos grupos.
  - Tempo máximo da arguição: 10 minutos.
- Os grupos que não obtiverem ao menos 36% dos 60% destinados à nota do Beecrowd poderão obter, no máximo, 10% nos 30% destinados à entrevista.