Relatório de desenvolvimento da A1

Aluno: Gabriel de Vargas Coelho

Matrícula: 20200400

Rodando os algoritmos

Para a execução correta dos algoritmos utilize no terminal os seguintes comandos:

1. Buscas:

   python buscas.py [arquivo] [nº inteiro > 0]

   exemplo: python3 buscas.py test.net 2

2. Ciclo Euleriano:

   python ciclo-euleriano.py [arquivo]

   exemplo: python3 ciclo-euleriano.py test.net

3. Bellman-Ford:

   python bellman-ford.py [arquivo] [nº inteiro > 0]

   exemplo: python3 bellman-ford.py test.net 2

4. Floyd-Warshall:

   python floyd-warshall.py [arquivo]

   exemplo: python3 floyd-warshall.py test.net

1. Representação

Na classe Grafo, são usadas 2 estruturas de dados, o vetor para os atributos de rótulo e de arestas existentes, e uma matriz |V|x|V| que armazena os custos entre as arestas. Essas estruturas foram escolhidas pois tornam a implementação dos demais algoritmos mais simples e de fácil compreensão.

2. Buscas

O algoritmo utiliza da classe Grafo para obter a representação esperada, e utiliza 3 vetores para manter o registro dos vetores já conhecidos, a distância até o vértice s de origem, e o ancestral dos vértices, representados pela posição no vetor ancestral. Foram utilizados essas estruturas pela semelhança com os pseudo-códigos apresentados em aula, focando assim na melhor compreensão do algoritmo e não na otimização desse.

Para apresentar o resultado em tela conforme o demandado pelo enunciado da atividade, utilizou-se de uma função específica para isso, que conta com um dicionário python para o agrupamento de vértices que se encontram no mesmo nível de busca. Vale ressaltar que tal função apresenta complexidade na ordem de O(|V|), uma vez que cada vértice do grafo é visitado somente uma vez.

3. Ciclo Euleriano

Para esse algoritmo foram utilizados vetores, dicionários, tuplas e conjuntos python. Essas estruturas de dados foram utilizados da seguinte forma: para manter registro do ciclo e dos subciclos encontrados pelo algoritmo, além de consulta ao vetor de arestas do grafo; como uma forma simples de manter registro das arestas já visitadas; as tuplas como representação de uma arestas; e o conjunto como forma de identificar vértices adjacentes não visitados a um ciclo formado.

O conjunto foi escolhido pois ele armazena somente uma cópia de cada dado nele inserido, dessa forma se no ciclo houvessem dois vértices vizinhos a um terceiro não visitado, ele não seria inserido duas vezes na estrutura, o que foi verificado, durante o desenvolvimento, causar erros na execução do programa.

4. Bellman-Ford

A escolha pelo algoritmo de Bellman-Ford se deu pela capacidade deste conseguir lidar com arestas de custo negativo, e como já foi apresentado, este trabalho focar na compreensão dos algoritmos e não na implementação mais computacionalmente eficiente. Sobre as estruturas de dados utilizadas, somente vetores foram usados, e representam a distância ao vértice informado, e os ancestrais.

Para manter a conformidade com a saída apresentada no enunciado, também foi implementado uma função para a montagem do resultado apresentado no terminal. A função é recursiva e tem como objetivo remontar o caminho utilizado para se chegar a determinado vértice a partir do vértice fonte, s, informado.

5. Floyd-Warshall

Para esse algoritmo foi utilizado somente uma matriz |V|x|V|, conforme apresentado em aula como sendo uma abordagem de menor impacto à memória. Vale ressaltar que como a classe Grafo já mantém um registro da matriz de custo do

grafo, a montagem da matriz W(G) se deu simplesmente pela cópia da matriz de custo.