INE5413 - Grafos - Relatório da atividade 2

David Grunheidt Vilela Ordine - 16202253

Para a criação do grafo, foi utilizado uma classe **'Directed Graph'**, a qual herda métodos da classe **'NotDirectedGraph'**, implementada na **Atividade 1**. Essa classe tem quase todos os métodos iguais a um grafo não direcionado. Porém, os métodos de adicionar uma aresta e remover uma aresta são diferentes.

Exercício 1: Para o algoritmo de componentes fortemente conexas, foram utilizadas três estruturas, sendo elas duas lista e um conjunto. As listas foram chamadas de 'stack' e 'result_search'. A variável 'stack' é responsável por guardar a ordem em que os vértices são visitados durante a execução do algoritmo de busca em profundidade. Já a variável 'result_search' é responsável por guardar todos cada componente fortemente conexa, ou seja, cada posição da lista tem um conjunto de vértices. Por fim, a variável 'visited' armazena o conjunto de vértices que foram visitados durante a execução da busca em profundidade.

Exercício 2: Para este exercício foi usado um algoritmo genérico de ordenação topológica, semelhante ao das anotações. Quanto às estruturas de dados, há uma lista e um conjunto. A variável 'stack' é uma lista e armazena a ordem em que os vértices foram visitados durante a busca em profundidade. Porém, diferente do exercício 1, estes vértices são armazenados na 'stack' de forma inversa, ou seja, são adicionados sempre no índice 0. Já a variável 'visited' é um conjunto que armazena todos os vértices visitados. O que difere a variável 'stack' da 'visited' é que, na primeira, os elementos são adicionados após a execução de uma interação recursiva da busca em profundidade, enquanto que, na segunda, os elementos são adicionados no início dessa execução. Outra lista é usada para fazer a conversão da lista 'stack', a qual armazena os ids de cada vértice visitado, para uma lista que armazena seus rótulos.

Exercício 3: Por fim, para o exercício 3, foi escolhido o algoritmo de árvore geradora mínima de prim, visto que o de kruskal envolve ordenações de arestas que são difíceis de fazer segundo a estrutura de dados de grafos adotada no trabalho. Foi usado somente uma estrutura de dados, sendo essa um dicionário. As entradas são os ids de cada vértices do grafo, e um valor é, novamente, um dicionário, contendo 3 chaves: 'parent' (o id do vértice parente aquele vértice), 'weight' (o peso da aresta entre aquele vértice e seu parente) e 'visited' (se já foi visitado).