Projeto e Análise de Algoritmo
- Trabalho Prático 2 -
Aluno: Leandro Balbino de Andrade

Visão geral

Este projeto implementa uma biblioteca para manipulação e análise de grafos em diferentes estruturas de representação: matriz de adjacência, matriz de incidência e lista de adjacência. Ele fornece algoritmos fundamentais para busca e processamento de grafos, incluindo:

- Busca em profundidade (DFS)
- Busca em largura (BFS)
- Árvore geradora mínima (AGM)

A interface gráfica desenvolvida permite que o usuário carregue grafos a partir de arquivos, selecione a estrutura de representação e execute algoritmos com opções personalizadas.

Estruturas de representação de grafos

1. Matriz de adjacência: Representa o grafo como uma matriz onde o elemento na linha (i) e coluna (j) indica o peso da aresta entre os vértices (i) e (j).

2. Matriz de incidência: Representa o grafo como uma matriz onde cada coluna representa uma aresta e cada linha representa um vértice. Os valores indicam se um vértice é origem ou destino da aresta.

3. Lista de adjacências: Utiliza um mapa para armazenar listas de adjacências de cada vértice. É eficiente para grafos esparsos e permite buscas rápidas pelos vértices adjacentes.

Algoritmos implementados

1. Busca em profundidade (DFS)

   O algoritmo de busca em profundidade é implementado para classificar as arestas do grafo em quatro categorias:

   - Aresta de árvore: Conecta um vértice descoberto com um novo vértice.
   - Aresta de retorno: Indicam ciclos no grafo.
   - Aresta de avanço: Conecta um vértice a outro já finalizado (em grafos orientados).
   - Aresta de cruzamento: Conecta dois vértices que não estão na mesma árvore de busca.
   - Além disso, a DFS registra os tempos de descoberta/finalização de cada vértice.

2. Busca em largura (BFS)

   A BFS é implementada para calcular:

   - Árvore de busca em largura: Conjunto de arestas que formam a estrutura de exploração.
   - Distâncias: Número de arestas do vértice inicial até cada outro vértice.
   - Predecessores: Representa o caminho percorrido até cada vértice.

3. Árvore geradora mínima (AGM)

   A implementação permite ao usuário selecionar um vértice inicial, embora o algoritmo funcione independentemente do ponto de partida. Utilizando o algoritmo de Kruskal, a AGM é calculada com os seguintes passos:

   - Ordenação das arestas pelo peso.
   - Adição de arestas ao conjunto da AGM enquanto não formam ciclos.
   - Uso da estrutura de união e busca para garantir a conectividade.

4. Caminho mínimo

   A implementação do cálculo de caminho mínimo permite ao usuário encontrar o menor caminho entre dois vértices em um grafo ponderado, utilizando o algoritmo de Dijkstra.

   - Inicialização das distâncias de todos os vértices para o valor infinito.
   - Uso de uma fila de prioridade para processar os vértices de acordo com a menor distância acumulada.
   - A cada iteração, o algoritmo examina os vértices adjacentes ao vértice atual, atualizando suas distâncias caso um caminho mais curto seja encontrado.
   - O algoritmo continua até processar todos os vértices ou até que o vértice de destino seja alcançado.
   - Reconstrução do caminho mínimo a partir do mapa de predecessores, retornando a sequência de arestas que formam o caminho de menor custo.

Interface gráfica

A interface gráfica foi desenvolvida utilizando o Java Swing e oferece as seguintes funcionalidades:

- Carregar o grafo: Permite selecionar um arquivo (.txt) com a descrição do grafo, além de diferentes estruturas de representação.
- Escolher o algoritmo: Permite selecionar entre os diferentes tipos de implementação de busca em grafo (Busca em profundidade, etc).
- Configurar o vértice inicial: Permite escolher o vértice inicial para buscar no grafo.
- Resultados: Exibe os resultados obtidos conforme especificado no PDF de descrição.

Limitações

- O algoritmo de fluxo máximo não foi implementado.

- Apenas grafos conectados são suportados para o cálculo da AGM.
