UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS TRINDADE
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO CIÊNCIAS DA COMPUTAÇÃO

Rita Louro Barbosa – 22203157

NE5413 – Grafos

Atividade Prática A2

Introdução

Este relatório contém a apresentação do trabalho realizado na resolução de 3 problemas propostos. Para essa resolução, foram implementados os seguintes algoritmos: Algoritmo para busca de componentes fortemente conectadas e demais algoritmos auxiliares; Algoritmo de ordenação topológica e demais algoritmo auxiliares e algoritmo de Kruskal para encontrar a árvore geradora mínima.

Observação importante: Procurou-se seguir com fidelidade os algoritmos apresentados no livro. Foram adicionados comentários ao longo do código para facilitar a identificação da implementação de cada etapa do pseudocódigo presente no livro.

1. Questão 1

Para essa questão, foi implementado o algoritmo de busca de componentes fortemente conectadas, descrito na página 70 do livro “Anotações para a disciplina de grafos".
Para isso, foram implementados outros algoritmos auxiliares:

- DFS de Cormen et al.;
- DFS de Cormen et al. adaptado no loop interno;
- DFS-Visit de Cormen et al.
- Algoritmo para busca e coleta das componentes encontradas
-

Foram criadas 4 estruturas complementares para os algoritmos:

- O vetor booleano ‘C’ para determinar o conhecimento dos vértices,
- Vetor T (Time Vector) , usado para armazenar os tempos de descoberta de cada vértice durante uma busca no grafo. Nele, cada elemento T[i] do vetor corresponde a um vértice i do grafo. O valor f32::INFINITY é utilizado para indicar que um vértice ainda não foi descoberto.
- Vetor F, usado para armazenar os tempos de finalização de cada vértice durante uma busca no grafo.
- Vetor A (Antecessor Vector), usado para armazenar o vértice antecessor de cada vértice no caminho percorrido durante a busca. O valor -1 (ou valor_vazio) é utilizado para indicar que um vértice não possui antecessor, geralmente porque é o vértice de início da busca. Durante a busca, este vetor é atualizado para rastrear o caminho percorrido até cada vértice.

Para representar os valores “infinito “ e “vazio”, foram utilizados, respectivamente, o valor máximo da representação de inteiro de 32 bits e o valor -1 (já que Rust não permite a criação de vetores de tipo i32 com posições com valor vazio).

2. Questão 2

Esta questão implementa a ordenação topológica, descrita na página 76 do livro “Anotações para a disciplina de grafos". Para resolução da questão, foram necessárias 2 algoritmos, o principal (DFS para Ordenação Topológica) e o auxiliar (DFS-Visit-OT).

As principais estruturas utilizadas são:

- Vetor C , utilizado para determinar se um vértice já foi visitado durante a ordenação topológica. Nele, o valor false indica que o vértice ainda não foi visitado, enquanto true indica que ele já foi visitado.
- Vetor T, usado para armazenar os tempos de descoberta de cada vértice durante a ordenação topológica. Durante a ordenação topológica, o tempo de descoberta de um vértice é atualizado quando ele é visitado pela primeira vez.
- Vetor F , usado para armazenar os tempos de finalização de cada vértice durante a ordenação topológica. Durante a ordenação topológica, o tempo de finalização de um vértice é atualizado quando toda a busca em seus vizinhos é concluída.

3. Questão 3

Foi implementado o algoritmo de Kruskal, para busca de árvore geradora mínima, descrito na página 81 do livro “Anotações para a disciplina de grafos". Para isso, foram usadas as seguintes estruturas:

- Conjunto de Arestas A, utilizado para armazenar as arestas da árvore geradora mínima. Nele, cada elemento (u, v) do conjunto corresponde a uma aresta entre os vértices u e v do grafo. Durante a execução do algoritmo, as arestas são adicionadas a este conjunto à medida que são selecionadas para fazer parte da árvore geradora mínima.
- Vetor S : Vetor que representa os conjuntos aos quais cada vértice é pertencente
- Vetor El (Lista de Arestas Ordenadas): Usado para armazenar todas as arestas do grafo em ordem crescente de peso. Cada elemento (u, v, w) do vetor corresponde a uma aresta entre os vértices u e v com peso w. Durante a execução do algoritmo, as arestas são ordenadas em ordem crescente de peso para que possam ser processadas em ordem durante a busca pela árvore geradora mínima.

6. Rodando arquivos:

- **Necessário:** Ter Rust e Cargo instalados.

Para isso, siga as instruções, conforme seu sistema operacional:

- Linux: Utilize o comando:

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

- Windows , com Chocolatey instalado, utilize o comando:
choco install rust

Para demais casos, verificar o instalador no site oficial de Rust:
https://doc.rust-lang.org/cargo/getting-started/installation.html

- **Rodando arquivos de questões:**

Vá até a pasta “codigo” e abra o terminal nela.

Comando genérico:
cargo run --bin qx nomeArquivo.net arg1 arg2

cargo run --bin t2 x nomeArquivo.net arg1 arg2

onde: x = número da questão e args conforme necessário

OBS: Arquivos de grafos devem ser colocados na pasta ‘grafos_teste’, localizada na pasta ‘src’

Comandos específicos:

Questão 1:

cargo run --bin t2 1 nomeArquivo.net

Questão 2:

cargo run --bin t2 2 nomeArquivo.net

Questão 3:

cargo run --bin t2 3 nomeArquivo.net