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

Rita Louro Barbosa – 22203157

NE5413 – Grafos

Atividade Prática A3

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 do fluxo máximo (Edmons karp) e demais algoritmos auxiliares; Algoritmo de emparelhamento máximo (hopcroft_karp) e demais algoritmo auxiliares e algoritmo de coloração (Lawler) e demais algoritmos auxiliares.

Observação importante: Para esse trabalho, foram consultadas alternativas de implementação dos algoritmos do livro. A lógica algorítmica é fiel a apresentada no livro, ainda que a implementação utilize estratégias específicas para a estruturação dessa lógica adaptando-a à linguagem. 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 pelo fluxo máximo de um grafo dirigido e ponderado, descrito na página 90 do livro “Anotações para a disciplina de grafos".
Para isso, foi implementado o algoritmo auxiliar:

- DFS para Edmons Karp;

Foram criadas 4 estruturas complementares para os algoritmos:

- 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.
- A fila ‘Q’ para implementação da ordem de busca
- A matriz F para armazenar o fluxo ao longo da execução do algoritmo

2. Questão 2

Esta questão implementa o emparelhamento, por Hopcroft-Karp, descrito na página 96 do livro “Anotações para a disciplina de grafos". Para resolução da questão, foram necessários 2 algoritmos auxiliares: A BFS adaptada para emparelhamento e a DFS também adaptada para emparelhamento.

As principais estruturas utilizadas são:

- Vetor D , utilizado para indicar a distância de um vértice
- Vetor mate, usado para monitorar o emparelhamento de um vértice
- Inteiro M para contar o número de emparelhamento
- A fila ‘Q’ para a busca em largura

Observação: A implementação dessa solução possui uma instabilidade na resposta encontrada, apresentando diferentes respostas para diferentes execuções. Isso pode estar acontecendo devido a algum possível problema de acesso à memória, não identificado. Caso a resposta encontrada na primeira execução não seja a esperada, próximas execuções do mesmo caso teste podem levar à resposta final.

3. Questão 3

Foi implementado o algoritmo de Lawler, para coloração de grafos, descrito na página 103 do livro “Anotações para a disciplina de grafos". Para resolução da questão, foram necessários 4 algoritmos auxiliares:

- Conjunto potência, que encontra o conjunto de todos os subconjuntos possíveis de um conjunto de vértices dado, incluindo o conjunto vazio e o próprio conjunto de vértices;
- Conjuntos independentes: gera todos os conjuntos independentes (conjuntos de vértices sem arestas entre eles) de um grafo ponderado fornecido;
- Colore: realiza a coloração dos vértices de um grafo para apresentação, já que Lawler só faz a contagem da coloração.
- sorted_set_difference utilizado para calcular a diferença entre conjuntos
-

Foram usadas as seguintes estruturas:

- A lista X, para controlar o número de cores, ela possui tamanho de 2^n com n = número de vértices.
- Vetor de vetores S, contendo o conjunto potência da lista de vértices do grafo
- f: dicionário para implementação da função f(s) que mapeia um subconjunto do conjunto de vértices em um número inteiro obtido a partir de uma representação binária ordenada do conjunto de vértices
- Vetor de vetores “R” que armazena a lista de conjuntos independentes.

Observação: Essa solução apresenta falhas e não funciona. Foi mantida no escopo da entrega para avaliação parcial do desenvolvimento da implementação. O erro possivelmente está na forma de adicionar ítens e utilizar a função f(s) implementada com hashmap. Pode estar também na criação e preenchimento de G’ (no código, G_). Nesta questão 3, a linguagem mais complexa e com recursos diferentes de manipulação de estruturas dificultou a implementação do algoritmo descrito no livro.

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 t3 2 emparelhamento_teste.net
cargo run --bin t3 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 t3 x nomeArquivo.net s t

Questão 2:

cargo run --bin t3 2 nomeArquivo.net

Questão 3:

cargo run --bin t3 3 nomeArquivo.net