Instituto Federal de Educação Ciência e Tecnologia de Brasília.
Alunos: Carlos Eduardo Pereira Santana; Danyelle da Silva Oliveira Angelo; Raquel Pinheiro da Costa;
Disciplina: Teoria dos Grafos - Professor: Raimundo Cláudio Vasconcelos

Trabalho Avaliativo 1 - Parte 2

1 Instruções de compilação

Abra o terminal na pasta libGraph execute os seguintes comandos:

gcc -c nomeArquivo.c

*F*aça isso para todos os arquivos .c, esse comando irá gerar arquivos objetos (.o), depois que estes arquivos tiverem sido gerados, digite ocomando abaixo:

ar rcs nomeBilbioteca.a arquivo1.o arquivo2.o ...

2 Instruções para execução

Na pasta raíz do programa (a biblioteca já deve ter sido gerada) execute os comandos(em negrito) abaixo:

gcc main.c -o nomeSaida nomeBiblioteca.a

Indique o caminho da biblioteca, caso ela esteja em outra pasta.

./nomeSaida

3 Arquivos

- Diretório raíz
  - main.c : arquivo principal;
  - busca_listAdj.txt : arquivo usado para montar árvores de busca para lista de adjascência;
  - busca_MatAdj.txt :arquivo usado para montar árvores de busca para matriz de adjascência;
  - grafo.txt : arquivo de entrada,contém as informações do grafo (número de vértices, e os pares de vértices);

- saida.txt : arquivo de saída com as informaços do grafo (número de arestas e vértices, grau de cada vértice).

- libGraph

  Para cada um dos arquivos .c listados abaixo, temos um arquivo objeto (.o)

  - graph.c : contém a função de inicialização do grafo, a que prenche o arquivo saida.txt, as funções de busca (para todas as representações) e o algoritmo de dijkstra;
  - list.c :contém as funções de lista e pilha;
  - listAdj.c : monta a lista de adjascência propriamente dito;
  - matAdj.c: monta a matriz de adjascência;
  - graph.h, list.h,listAdj.h, matAdj.h: contrato dos arquivos .c que recebem o seu nome, este arquivo apresenta os protótipos das funções e um apanhado geral de cada função implementada, tudo que é declarado nele, deve estar no arquivo .c associado a ele.
  - libGraph.a : biblioteca com as funções implementadas nos arquivos citados acima, na hora de compilar um arquivo que precisa usar as funções dos arquivos citados acima, basta compilar o arquivo em questão junto a essa biblioteca.

4 Estudo de Caso

Para fazer esta parte do trabalho optamos por fazer a seguinte mudança na parte 1: na matriz de adjascência, ao invés de indicarmos as conexões entre os vértices 'i' e 'j' (linha i, coluna j) com o inteiro 1, indicamos ela com o peso dos vértices. Para a lista de adjascência apesar de fazermos a leitura do peso indicado no arquivo de entrada, não usamos este dado de fato.
Desta forma a única forma de calcular o menor caminho é através da matriz de adjascência, optamos por fazer desta forma por acreditar que assim seria mais fácil de manipular os dados.

Consideramos também neste trabalho (como aconselhado pelo professor) que uma aresta não pode ter peso zero, assim não precisaríamos inicializar toda a matriz de adjascência com valores diferentes de zero (diminuindo o custo computacional).

Com relação à verificação das arestas com pesos negativos, criamos um teste na função coust_min disponível arquivo graph.c que retorna um valor negativo à função que o invocou(dijkstra) caso não seja achado um caminho para um próximo vértice(ou não existem arestas com pesos positivos, ou o vértice analisado é uma folha). Neste caso, a função invocadora exibe uma mensagem de erro ao usuário.

Agora considere o grafo disponibilizado no site da disciplina. Ao substituirmos os dados do arquivo grafo.txt (usado como entrada na nossa biblioteca) pelos valores disponibilizados, obteremos os seguintes resultados ao buscar o menor caminho entre dois vértices (opção 4 do menu).

1. **1 à 10**

2. **1 à 100**

3. **1 à 1000**

4. **1 à 10000**