UNIVERSIDADE FEDERAL DE RONDÔNIA
DEPARTAMENTO ACADÊMICO DE CIÊNCIA DA COMPUTAÇÃO
ESTRUTURA DE DADOS II

TRABALHO 1 - GRAFOS

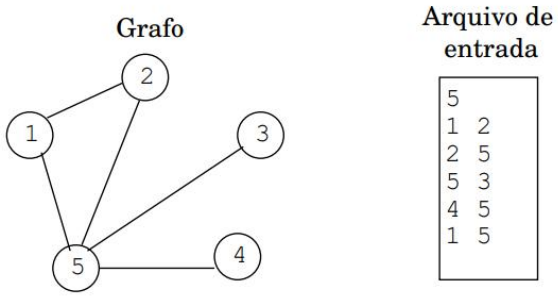
FAZER EM DUPLAS
ENTREGA: 21/06/2024
ENTREGAR PELO SIGAA.
APENAS UM DA DUPLA DEVE ENTREGAR O TRABALHO.

O objetivo do trabalho é projetar e desenvolver uma biblioteca para manipular grafos. A biblioteca deverá ser capaz de representar grafos assim como implementar um conjunto de algoritmos em grafos. Para fins deste trabalho, a biblioteca pode ser um conjunto de funções em linguagem C, dividida em dois arquivos: .h e um .c. A biblioteca deve ser desenvolvida de forma a respeitar o conceito de Tipo Abstrato de Dados Grafo, para que possa ser facilmente utilizada em outros programas. O trabalho também deve conter um arquivo .c que faça as chamadas das funções da biblioteca desenvolvida.

Deve ser entregue também um relatório informando as decisões do projeto e implementação das funcionalidades. Além disso, o relatório deve relsponder às perguntas relacionadas aos estudos de caso.

PARTE 1 - GRAFOS NÃO PONDERADOS

DESCRIÇÃO
Segue abaixo as funcionalidades que precisam ser oferecidas pela biblioteca nesta parte do trabalho. Sua biblioteca irá trabalhar apenas com grafos não-direcionados.



Exemplo de grafo e formato de entrada

1. **Entrada**. Sua biblioteca deve ser capaz de ler um grafo de um arquivo texto. O formato do grafo no arquivo será o seguinte. A primeira linha informa o número de vértices do grafo. Cada linha subsequente informa as arestas. Um exemplo de um grafo e seu respectivo arquivo texto é dado na figura 1.

2. **Saída**. Sua biblioteca deve ser capaz de gerar um arquivo texto com as seguintes informações sobre o grafo: número de vértices, número de arestas, grau mínimo, grau máximo, grau médio, e mediana de grau. Além disso, imprimir informações sobre as componentes conexas (ver abaixo).

3. **Representação de grafos**. Sua biblioteca deve ser capaz de representar grafos utilizando tanto uma matriz de adjacência, quanto uma lista ou vetor de adjacência. O usuário da biblioteca (programa que irá usá-la) poderá escolher a representação a ser utilizada.

4. **Busca em grafos: largura e profundidade**. Sua biblioteca deve ser capaz de percorrer o grafo utilizando busca em largura e busca em profundidade. O vértice inicial será dado pelo usuário da biblioteca. A respectiva árvore de busca deve ser gerada assim como o nível de cada vértice na árvore (nível da raiz é zero). Estas informações devem ser impressas em um arquivo. Para descrever a árvore gerada, basta informar o pai de cada vértice e seu nível no arquivo de saída.

5. **Distâncias e diâmetro**. Sua biblioteca deve ser capaz de determinar a distância entre dois vértices do grafo (utilizando como primitiva a BFS) assim como calcular o diâmetro do grafo. Lembrando que o diâmetro é a maior distância entre qualquer par de vértices do grafo (ou seja, o comprimento do maior caminho mínimo do grafo).

6. **Componentes conexos**. Sua biblioteca deve ser capaz descobrir as componentes conexas de um grafo. O número de componentes conexos, assim como o tamanho (em vértices) de cada componente e a lista de vértices pertencentes à componente. Os componentes devem estar listados em ordem decrescente de tamanho (listar primeiro o componente com o maior número de vértices, etc).

ESTUDO DE CASO

Considerando cada um dos grafos indicados nos arquivos PARTE 1, responda às perguntas abaixo:

1. Compare o desempenho em termos de quantidade de memória utilizada pelas duas representações do grafo. Ou seja, determine a quantidade de memória (em MB) utilizada pelo seu programa quando você representa o grafo utilizando uma matriz de adjacência e lista de adjacência. Dica: pause a execução do programa depois de carregar o grafo e verifique a memória sendo utilizada pelo processo.

2. Compare o desempenho em termos de tempo de execução das duas representações do grafo. Execute **100** buscas em largura em cada um dos casos (utilize diferentes vértices como ponto de partida da busca), e obtenha o tempo médio de execução de uma busca.

3. Repita o item anterior para busca em profundidade.

4. Determine o pai dos vértices 10, 20, 30 na árvore geradora induzida pela BFS e pela DFS quando iniciamos a busca nos vértices 1, 2, 3.
5. Determine a distância entre os seguintes pares de vértices (10,20), (10,30), (20,30).

6. Obtenha as componentes conexas do grafo. Quantas componentes conexas tem o grafo? Qual é o tamanho da maior e da menor componente conexa?

7. Determine o diâmetro do grafo. Dica: implemente também um algoritmo aproximativo, que pode ser usado em grafos muito grandes.

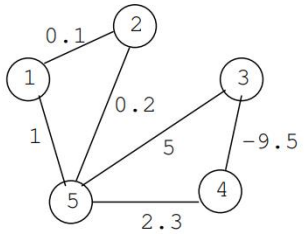
Você deve preparar tabelas com os resultados obtidos onde as colunas representam as características e as linhas representam os diferentes grafos analisados. Inclua esta tabela em seu relatório.

Importante: Todas as medidas de tempo devem ignorar o tempo gasto lendo o grafo do disco e o tempo gasto escrevendo o resultado no disco ou no monitor. Ou seja, contabilize apenas o tempo de execução do algoritmo. Consulte o relógio da máquina antes do algoritmo iniciar e depois do algoritmo terminar, e use a diferença para obter o tempo decorrido.

PARTE 2 - GRAFOS PONDERADOS

Funcionalidades que precisam ser implementadas pela sua biblioteca:

Grafo



Arquivo de entrada

```
5
1 2 0.1
2 5 0.2
5 3 5
3 4 -9.5
4 5 2.3
1 5 1
```

Exemplo de grafo com pesos e o formato do arquivo.

1. **Grafos com pesos.** Sua biblioteca deve ser capaz de representar e manipular grafos não-direcionados que possuam pesos nas arestas. Os pesos, que serão representados por valores reais, devem estar associados às arestas. Você deve decidir a melhor forma de estender sua biblioteca de forma a implementar esta nova funcionalidade. O arquivo de entrada será modificado, tendo agora uma terceira coluna, que representa o peso da aresta (podendo ser qualquer número em ponto flutuante). Um exemplo de um grafo não-direcionado com pesos e seu respectivo arquivo de entrada está ilustrado na figura 1.

2. **Distância e caminho mínimo.** Sua biblioteca deve ser capaz de encontrar a distância de um vértice qualquer para todos os outros vértices do grafo, assim como os caminhos mínimos (árvore geradora induzida pela busca). Se o grafo possuir pesos não negativos, o algoritmo de Dijkstra deve ser utilizado (caso contrário informar que a biblioteca ainda não implementa caminhos mínimos com pesos negativos).

   Você deve implementar o algoritmo de Dijkstra de duas formas: (1) utilizando um vetor para armazenar as estimativas de distâncias para cada vértice; (2) utilizando um heap para armazenar as estimativas de distâncias para cada vértice. No caso do heap, você pode implementar seu próprio heap ou utilizar uma biblioteca que possua um heap que possa ter as chaves dos seus elementos modificados (atenção com este aspecto).

ESTUDOS DE CASO

Considere os grafos com pesos disponíveis na pasta Parte 2. Para cada grafo, responda às perguntas abaixo:

1. Calcule a distância e o caminho mínimo entre o vértice 10 e os vértices 20, 30, 40, 50, 60. Apresente os resultados em uma tabela.

2. Determine o tempo médio para calcular a distância entre um vértice e todos os outros vértices do grafo para cada uma das implementações do algoritmo de Dijkstra (com e sem heap). Escolha k vértices iniciais de forma aleatória (ex. $k = 100$), calcule o tempo total de execução, e faça a média amostral. Apresente os resultados em uma tabela.

Considere a rede de colaboração entre pesquisadores da área de Computação disponível no website da disciplina, onde o peso da aresta indica a proximidade entre os pesquisadores (peso é inversamente proporcional ao número de artigos publicados em co-autoria). Responda às perguntas abaixo.

1. Calcule a distância e o caminho mínimo entre Edsger W. Dijkstra (o pesquisador) e os seguintes pesquisadores da rede de colaboração: Alan M. Turing, J. B. Kruskal, Jon M. Kleinberg, Éva Tardos, Daniel R. Figueiredo. Utilize exatamente estes nomes (strings) para identificar os índices dos vértices no grafo.