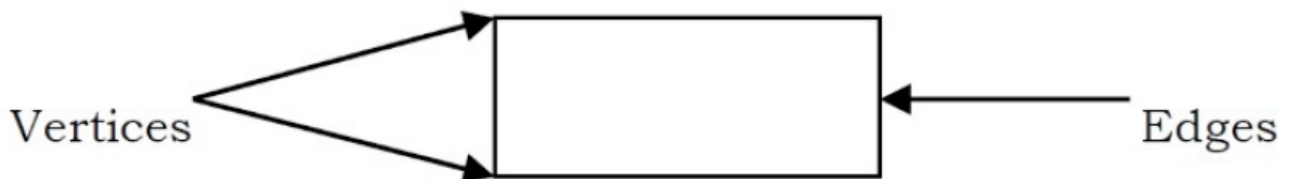


Atividade Prática 1
Valor: 30% da 1ª Avaliação
Entrega: 29/09/2019 (Individual)

1. DESCRIÇÃO

A árvore geradora (ou árvore de expansão) de grafo é um subgrafo que contém todos os vértices e também é uma árvore. Um grafo pode ter muitas árvores geradoras. Como exemplo, considere um grafo com 4 vértices como mostrado abaixo. Vamos supor que os cantos do grafo sejam vértices.



Para este grafo, nós podemos ter múltiplas árvores geradoras como ilustra a figura abaixo:



A árvore geradora mínima de um grafo não direcionado G é uma árvore formada por arestas do grafo que conectam todos os vértices de G com o menor custo (Somatório dos pesos). Existem dois algoritmos conhecidos que resolvem esse problema: Algoritmo de Prim e Algoritmo de Kruskal.

Neste trabalho será utilizado o algoritmo de Kruskal, por ter uma etapa que necessita da execução de um algoritmo de ordenação.

O algoritmo de Kruskal inicia com V diferentes árvores (V são vértices do grafo). Este algoritmo encontra a aresta segura (e com menor peso) procurando em todas as arestas que conectam quaisquer duas árvores na floresta A . A cada passo o algoritmo de Kruskal adiciona à floresta a aresta com menor peso possível.

O algoritmo de Kruskal utiliza conjuntos disjuntos, em cada conjunto contém os vértices de uma árvore na floresta atual. A operação $\text{FIND-SET}(u)$ retorna um elemento representativo do conjunto que contém u . Assim, para determinar se dois vértices u e v pertencem à mesma árvore, basta testar se $\text{FIND-SET}(u)$ é igual a $\text{FIND-SET}(v)$. A combinação de árvores é executada com o procedimento UNION . A seguir temos o pseudoalgoritmo:

```

MST-KRUSKAL( $G, w$ )
1   $A \leftarrow \emptyset$ 
2  for each vertex  $v \in V[G]$ 
3      do MAKE-SET( $v$ )
4  sort the edges of  $E$  into nondecreasing order by weight  $w$ 
5  for each edge  $(u, v) \in E$ , taken in nondecreasing order by weight
6      do if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ )
7          then  $A \leftarrow A \cup \{(u, v)\}$ 
8              UNION( $u, v$ )
9  return  $A$ 

```

Para mais detalhes podem ser encontrados no capítulo 23 do livro, Algoritmos: teoria e prática (CORMEN et al., 2002) e nas notas de aula (slide 20, aula sobre Kruskal).

Será disponibilizada a implementação do algoritmo do Kruskal implementado em Python e Java.

Você deverá IMPLEMENTAR os algoritmos de ordenação estudados em sala de aula (**InsertSort**, **SelectSort**, **ShellSort**, **QuickSort**, **MergeSort**) para serem utilizados na etapa de ordenação das arestas realizada pelo algoritmo de Kruskal.. A especificação e desenvolvimento dos tipos abstratos de dados a serem utilizados no trabalho são de sua competência e fazem parte da avaliação.

Os algoritmos de ordenação deverão se aplicados da seguinte maneira:

- 1) **QuickSort**
- 2) **MergeSort**
- 3) **InsertSort**
- 4) **Quicksort + Inserção Parcial:** O Quicksort deve ser executado recursivamente (como padrão) e, quando você obtiver uma partição de tamanho menor ou igual a L , essa partição deverá ser imediatamente ordenada usando o algoritmo de Inserção.
- 5) **Quicksort + Inserção Final:** O Quicksort deve ser executado recursivamente (como padrão) e, quando você obtiver uma partição de tamanho menor ou igual a L , essa partição deverá ser ignorada e continua particionando as partições maiores. Uma vez que todas as partições tenham comprimento menor ou igual a L , o algoritmo deverá executar o algoritmo de inserção.
- 6) **Mergesort + Inserção Parcial:** O Mergesort deverá ser executado recursivamente (como padrão) e uma vez que suas listas da esquerda e direita tiverem comprimento menor ou igual a L , elas deverão ser ordenadas independentemente usando o algoritmo de Inserção e, então, efetivamente mescladas.
- 7) **Mergesort + Inserção Final:** Mergesort deverá ser executado recursivamente (como padrão) e uma vez que suas listas da esquerda e direita tiverem comprimento menor ou igual a L , o Mergesort deve parar e imediatamente mesclar essas duas listas (certamente, isso não faria sentido desde o início). A mesclagem é feita com listas não ordenadas). Finalmente o algoritmo de Inserção deve ser executado em toda a lista

Discuta no relatório os resultados obtidos por essas abordagens. A substituição do algoritmo de Inserção pelos algoritmos SelectSort e ShellSort resultaria em redução no tempo de ordenação?

Você deverá carregar o vértices e arestas do grafo de através da leitura de arquivo.

Seu programa deverá ser executado passando-se opções de execução via linha de comando. Esse tipo de execução é bastante comum em sistemas Unix / Linux e no antigo DOS. Por exemplo, os parâmetros do programa podem ser definidos assim:

<Chamada ao Kruskal> <arquivo> <algoritmo ordenacao> <parâmetros ordenacao>

2. Descrição sobre os arquivos de dados

Serão utilizados arquivos de tamanho pequeno, média e grande definidos pela quantidade de arestas e vértices do grafo.

3. Análise dos resultados

A análise deve ser feita sobre o número de comparações, atribuições e tempo de execução dos algoritmos. Procure organizar inteligentemente os dados coletados em tabelas, e também construa gráficos a partir dos dados. Então, disserte sobre os dados nas tabelas e gráficos. Grande parte da avaliação será dedicada a análise dos resultados, ou seja, sobre o que você dissertar.

4. Entrega

- Código fonte do programa em C/C++ , Java ou Python (bem indentado e comentado).
- Relatório do trabalho
- Upload no SIGAA.

O Relatório deve apresentar:

1. Implementação: descrição sobre a implementação do programa. Devem ser detalhadas as estruturas de dados utilizadas (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
2. Análise de complexidade: apresentar o estudo de complexidade das principais funções implementadas usando a notação O.
3. Testes: apresentação dos testes realizados.
4. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação
5. Referências: referências utilizadas no desenvolvimento do trabalho.