unioeste
Universidade Estadual do Oeste do Paraná

Trabalho I de Projeto e Análise de Algoritmos

Ciência da Computação
campus Foz do Iguaçu

Data: Agosto/2021
Prof. Rômulo Silva

Modalidade:

- assíncrono com os horários de aula
- em grupo de 2 alunos

Data de entrega: 08/setembro/2021 até 23:59 hs no ambiente Microsoft Teams. Entregar um arquivo compactado `.zip` contendo SOMENTE os arquivos-fontes.

Fazer um programa em linguagem C, C++ ou Java para implementar os seguintes algoritmos em grafos:

1. **Busca em Profundidade** a partir de um dado vértice de origem
2. **Busca em Largura** a partir de um dado vértice de origem
3. **Bellman-Ford**: dado um vértice de origem calcular os menores caminhos para os demais vértices
4. **Kruskal**: dado um vértice de origem, apresentar uma árvore geradora mínima.

Devem ser observadas as seguintes condições:

- As estruturas de dados de necessárias para representação dos grafos e implementação dos algoritmos devem ser implementadas integralmente, não podendo ser utilizadas bibliotecas prontas.
- NÃO UTILIZAR bibliotecas específicas, que funcionem somente em um ambiente (sistema operacional e/ou IDE).
- O menu do programa deve primeiramente apresentar a opção **Carregar grafo**, que permite informar o caminho de um arquivo texto que contém as informações do grafo, de acordo com a sintaxe definida mais abaixo.
- Em seguida, o programa deverá mostrar um menu com uma opção para cada algoritmo, sendo possível ao usuário executar mais de um algoritmo sobre o mesmo grafo carregado do arquivo.

O arquivo deve conter um único grafo, e as informações contidas no arquivo têm a seguinte sintaxe:

1. `orientado=sim` ou `orientado=nao`: indica se o grafo é orientado ou não. É a primeira linha do arquivo.
2. `V=<n>`: contém o número de vértices do grafo. Os vértices são numerados de `0` a `n-1`. É a segunda linha do arquivo.
3. `(<u>,<v>):<peso>` representa a aresta (u, v) com o respectivo peso. `<u>` e `<v>` são inteiros entre `0` e `n-1`. O peso deve ser um número inteiro, podendo ser negativo. As arestas são especificadas a partir da terceira linha do arquivo, sendo 1 por linha.

No caso de grafo não-orientado, a aresta (u, v) aparecerá apenas uma vez no arquivo, devendo ser considerada a mesma aresta (v, u).

Conteúdo do arquivo para o grafo da Figura 1:

```
orientado=sim
V=5
(0,1):11
(0,2):-4
(1,3):-5
(2,4):7
(3,0):2
(3,2):8
(4,2):19
```

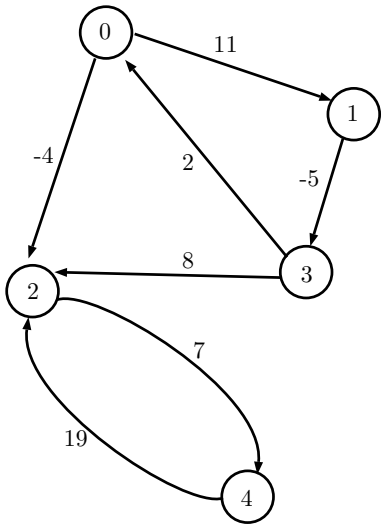

Figura 1: Grafo orientado

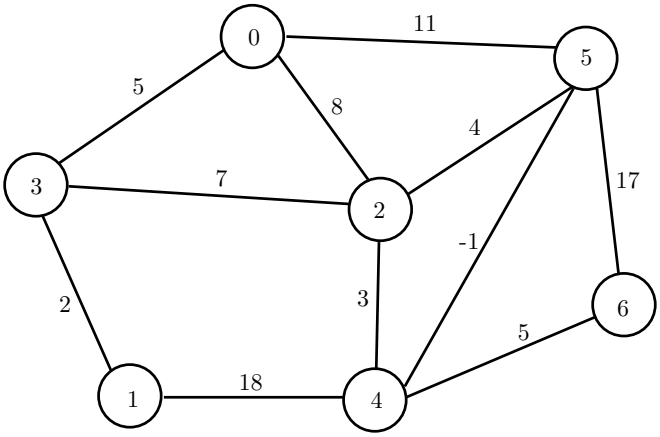

Figura 2: Grafo não-orientado

Conteúdo do arquivo para o grafo da Figura 2:

```
orientado=nao
V=7
(0,2):8
(0,3):5
(0,5):11
(1,3):2
(1,4):18
(2,3):7
(2,5):4
(2,4):3
(4,5):-1
(4,6):5
(5,6):17
```

Todos os algoritmos devem imprimir seu resultado na saída padrão de maneira identificar corretamente o resultado, conforme a seguir:

- **Busca em Profundidade**: ordem de visitação vértices. Por exemplo, considerado o grafo da Figura 2 e tendo como origem o vértice `3`, a saída é:
  `3 - 0 - 2 - 4 - 1 - 5 - 6`

- **Busca em Largura**: ordem de visitação dos vértices. Por exemplo, considerado o grafo da Figura 2 e tendo como origem o vértice `3`, a saída é:
  `3 - 0 - 1 - 2 - 5 - 4 - 6`

- **Bellman-Ford**: vértice de origem e vértice de destino com a respectiva distância. Por exemplo, considerando o grafo da Figura 2 e tendo como origem o vértice `3`, a saída é:

```
origem:  3
destino:  0 dist.:  5 caminho:  3 - 0
destino:  1 dist.:  2 caminho:  3 - 1
destino:  2 dist.:  7 caminho:  3 - 2
destino:  3 dist.:  0 caminho:  3
destino:  4 dist.:  10 caminho:  3 - 2 - 4
destino:  5 dist.:  9 caminho:  3 - 2 - 4 - 5
destino:  6 dist.:  15 caminho:  3 - 2 - 4 - 6
```

- **Kruskal**: arestas que formam a árvore geradora mínima. Por exemplo, considerando o grafo da Figura 2, a saída é:

```
peso total:  21
arestas:  (4,5) (1,3) (2,4) (0,3) (4,6) (2,3)
```

Critérios de avaliação:

- funcionamento correto dos algoritmos: peso 8
- estrutura/qualidade/documentação do código: peso 2

Trabalhos plagiados ou entregues fora do prazo receberão nota ZERO.