unioeste
Universidade Estadual do Oeste do Paraná

Trabalho II de Projeto e Análise de Algoritmos

Ciência da Computação
campus Foz do Iguaçu

Data: Abril/2022
Prof. Rômulo Silva

Modalidade:em grupo de 2 alunos

Data de entrega: 30/05/2022 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**: calcular árvore geradora mínima
5. **Prim**: calcular 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, cuja sintaxe está 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.
- além dos algoritmos mencionados anteriormente, o menu deve ter uma opção **Desenhar grafo**, que apresenta um desenho gráfico do grafo carregado, contendo a rotulação de vértices e pesos das arestas. O desenho de grafos pode ser feito usando qualquer biblioteca pronta para esse propósito, como, por exemplo, a encontrada em http://www.graphviz.org/
- para os algoritmos Kruskal e Prim, além da saída na console, deve ser desenhado o grafo assinalando com cor diferente as arestas que formam a árvore geradora mínima.

O arquivo texto 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:

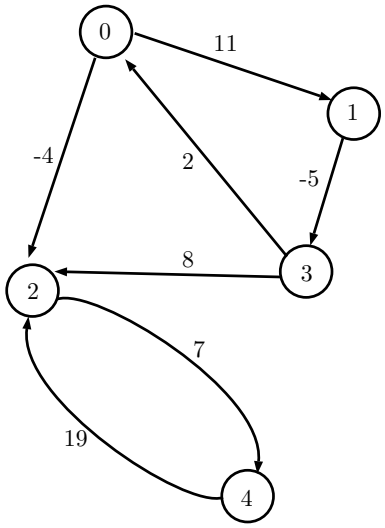

Figura 1: Grafo orientado

```
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
```

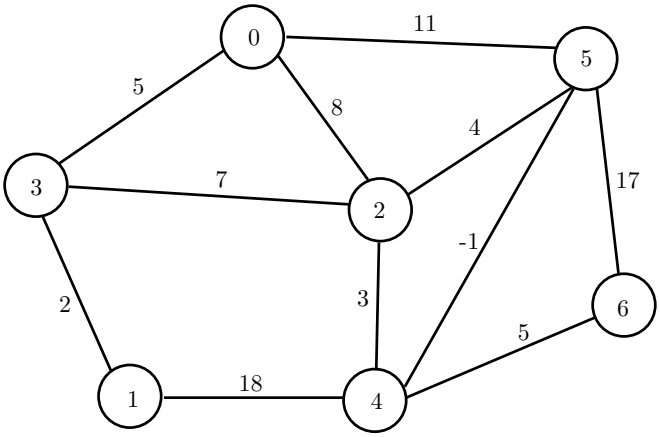

Figura 2: Grafo não-orientado

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
```

Ao fazer a representação do grafo na estrutura de dados, os vértices adjacentes devem estar armazenados em ordem crescente. Assim, por exemplo, no grafo da Figura 2, o vértice `4` tem como vértices adjacentes `1`, `2`, `5` e `6`, **nessa ordem**.

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 1 e tendo como origem o vértice `1`, a saída é:
  ```
  origem:  1
  destino:  0 dist.:  -3 caminho:  1 - 3 - 0
  destino:  1 dist.:  0 caminho:  1
  destino:  2 dist.:  -7 caminho:  1 - 3 - 0 - 2
  destino:  3 dist.:  -5 caminho:  1 - 3
  destino:  4 dist.:  0 caminho:  1 - 3 - 0 - 2 - 4
  ```

- **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)
  ```

- **Prim**: arestas que formam a árvore geradora mínima. Por exemplo, considerando o grafo da Figura 2, e vértice inicial 2,a saída é:
  ```
  vértice inicial:  2
  peso total:  21
  arestas:  (2,3),(2,4),(4,5),(4,6),(3,1),(3,0)
  ```

Critérios de avaliação:

- funcionamento correto dos algoritmos: peso 8
- desenho do grafo: peso 1
- estrutura/qualidade/documentação do código: peso 1

Trabalhos plagiados ou entregues fora do prazo receberão nota ZERO!