README

Programa em linguagem C++ que implementa os seguintes algoritmos em grafos:

1. **Desenhar Grafo**: apresenta um desenho gráfico do grafo carregado, contendo a rotulação de vértices e pesos das arestas. O desenho de grafos é feito usando a biblioteca pronta http://www.graphviz.org/
2. **Busca em Profundidade** a partir de um dado vértice de origem
3. **Busca em Largura** a partir de um dado vértice de origem
4. **Bellman-Ford**: dado um vértice de origem de um grafo orientado, calcular os menores caminhos para os demais vértices
5. **Kruskal**: calcular árvore geradora m´ınima. Além da impressão do resultado na console, deve ser desenhado o grafo, assinalando com uma cor diferente as arestas que formam a árvore geradora m´ınima.
6. **Componentes conexas**: dado um grafo não orientado, apresentar os vértices que formam cada componente conexa do grafo. A componentes devem ser apresentadas por ordem decrescente de tamanho, isto é, da maior para a menor.
7. **Componentes Fortemente Conexas**: dado um grafo orientado, apresentar os vértices que formam cada componente fortemente conexa do grafo. A componentes devem ser apresentadas por ordem decrescente de tamanho, isto é, da maior para a menor. Além da impressão do resultado na console, deve ser desenhado o grafo em que todos os vértices de cada componente fortemente conexa sejam pintados com uma mesma cor, porém diferente da cor das demais componentes fortemente conexas.

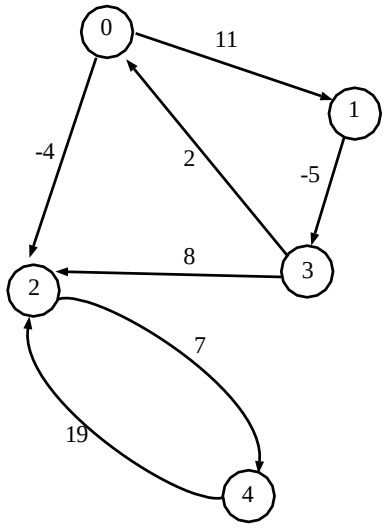

Figura 1: Grafo orientado

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

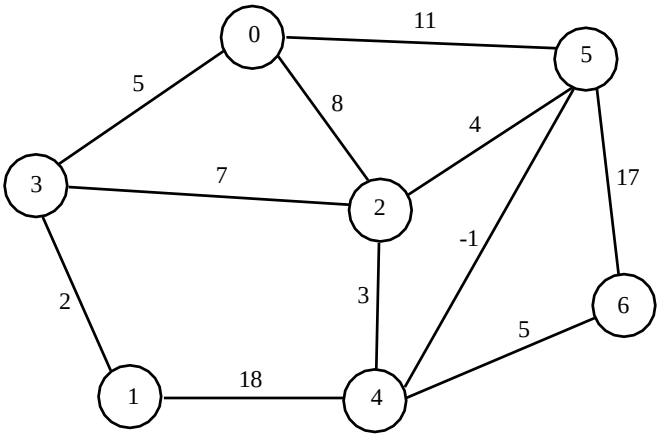

Figura 2: Grafo não-orientado

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

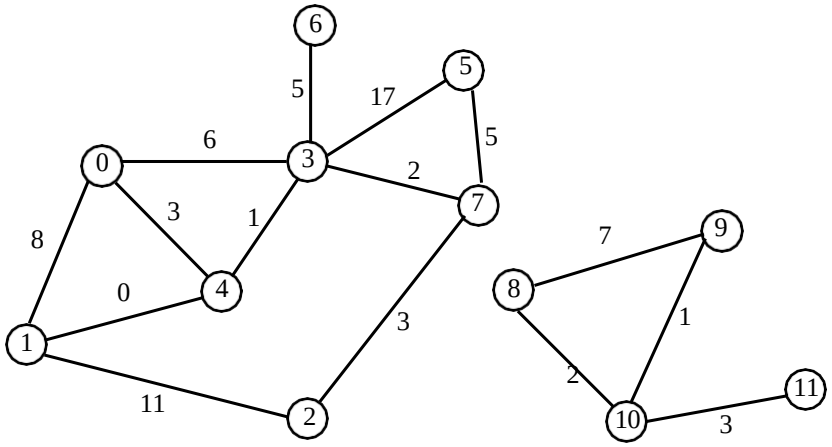

Figura 3: Grafo não-orientado (sem pesos negativos)

Conteúdo do arquivo para o grafo da Figura 3:

orientado=nao
V=12

(0,1):8
(0,3):6
(0,4):3
(1,4):0
(1,2):11
(2,7):3
(3,4):1
(3,5):17
(3,6):5
(3,7):2
(5,7):5
(8,9):7
(8,10):2
(9,10):1
(10,11):3

Todos os algoritmos imprimem seu resultado na saída padrão, 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, con- siderando 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)
- **Componentes Conexas**: vértices que formam cada componente conexa. Por exemplo, considerando o grafo da Figura 3, a saída é:
  Componente 1: 0, 1, 2, 7, 3, 4, 5, 6
  Componente 2: 8, 9, 10, 11
- **Componentes fortemente Conexas**: vértices que formam cada componente fortemente conexa. Por exemplo, considerando o grafo da Figura 1, a saída é:
  Componente 1: 0, 1, 3
  Componente 2: 2, 4