Laboratório de grafos
Prof. João B. Oliveira

1. Use os arquivos fornecidos na página da disciplina, que implementam uma classe Grafo e um pequeno programa que a usa para testes. Perceba que a classe `GrafoDir` oferece os métodos abaixo:

```
public GrafoDir( int n ) // n é o número máximo de nodos no grafo
public int NumNodos()    // retorna o máximo de nodos
public int NumArestas()  // retorna o número de arestas
public int CapNodos()    // retorna o num. de nodos em uso
-------------------------------------------
public void InsereAresta( int orig, int dest )
public void RemoveAresta( int orig, int dest )
public boolean ExisteAresta( int orig, int dest )
-------------------------------------------
public void DesmarcaNodos()
public void Marca( int n )
public boolean isMarked( int n )
-------------------------------------------
public List<Integer> Nodos() // retorna a lista dos nodos
public List<Integer> Vizinhos( int n )
```

2. Para este exercício supõe-se que os grafos são armazenados como no exemplo abaixo:

```
3 5   // nodo 3 liga com nodo 5
7 8   // nodo 7 liga com nodo 8
0 0   // final de entrada
```

   (a) Leia um grafo G de um arquivo e depois liste os nodos de G e seus graus.

   (b) Dado um grafo G e um de seus nodos v, escreva um algoritmo que lista todos os nodos de G que podem ser alcançados a partir de v.

   (c) Dado um grafo G e um de seus nodos v, escreva um algoritmo que lista todos os outros nodos de G, informando a que distância eles estão do nodo v.

3. Para um grafo dirigido, escreva o algoritmo que descobre quais os vértices onde não chega nenhuma aresta.

4. Escreva um método `Componentes`(G) que determina quantos componentes o grafo G possui.

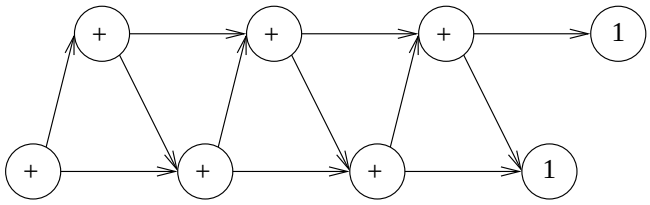
5. Para um grafo dirigido G, apresente um algoritmo `Não_posso_alcançar`(G,v) que imprime os vértices que **não** podem ser alcançados a partir do vértice v.

6. Para um grafo dirigido G e vértices u e v pertencentes a G, apresente um algoritmo `Caminhos`(G,u,v) que conta quantos são os caminhos de u até v.

7. Para um grafo dirigido e **sem ciclos** G apresente um algoritmo `Ordena`(G,u,v) que imprime os vértices de G com a seguinte condição: se há um caminho de u até v no grafo, então u deve ser impresso antes de v na saída. (Isso se chama **ordenação topológica**).

8. Dado um grafo não-ponderado e não-dirigido G e dois vértices u e v pertencentes a G, apresente o algoritmo que determina a menor distância (em número de arestas) de u a v.

9. Para o grafo ao lado, que representa uma coleção de valores inteiros, escreva um algoritmo que percorre os vértices do grafo calculando o valor que cada vértice representa. Certifique-se de que um vértice tem os seus operandos já disponíveis quando vai ser calculado!



10. Dado um grafo ponderado G e dois vértices u e v pertencentes a G, apresente o algoritmo que determina a menor distância (em soma dos pesos) de u até v.

11. Para um grafo dirigido e acíclico G, a *atingibilidade* de um nodo v é o número de nodos que tem um caminho até v. Apresente um algoritmo que recebe um grafo G e apresenta a atingibilidade de cada nodo do grafo.

12. Para um grafo não-dirigido G, apresente um algoritmo `Posso_alcançar`(G,u,v) que imprime os vértices que podem ser alcançados a partir do vértice u e também a partir do vértice v. Não repita vértices na saída.

13. Escreva um algoritmo que lista todas as arestas de um grafo que são pontes.

14. Leia um arquivo contendo uma lista de palavras. Em seguida construa um grafo G onde os nodos representam as palavras e existem arestas entre dois nodos se as duas palavras tiverem apenas uma letra de diferença, pelas seguintes regras:

    (a) Uma letra a mais ou a menos

        i. Uma letra trocada por outra letra.

    (b) Usando o grafo construído em 14, escreva um algoritmo que recebe uma palavra P e lista toda as suas vizinhas imediatas, ou seja, todas as palavras que são originadas a partir de P mudando-se apenas uma letra.

    (c) Usando o grafo construído em 14, escreva um algoritmo que acha o caminho mais curto para transformar uma palavra P_1 em outra palavra P_2.