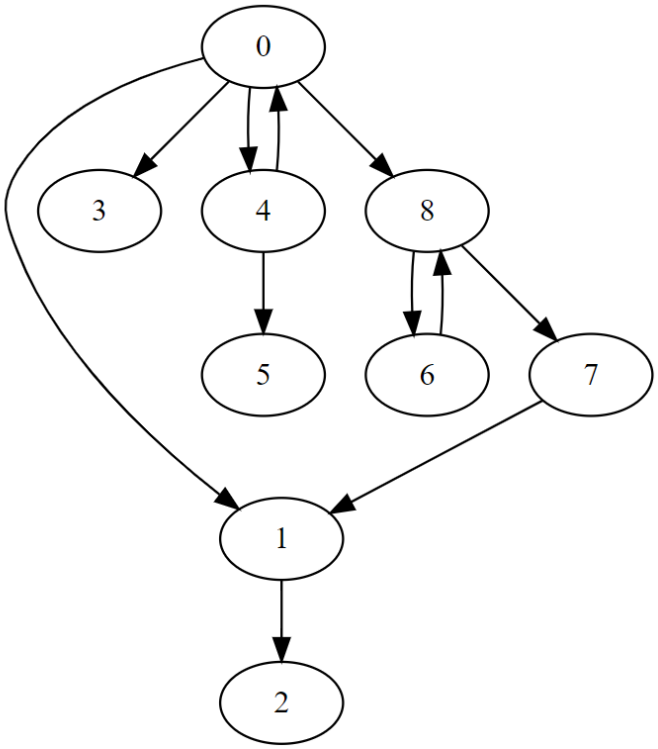
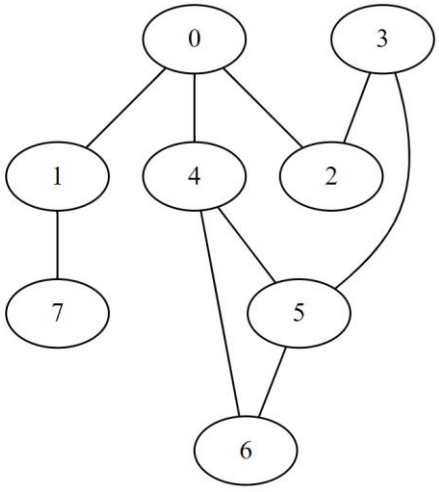
1) Observe o Grafo abaixo.



   a) Quantos vértices e arestas esse grafo apresenta?
   b) É um grafo direcionado ou não direcionado?
   c) Represente esse grafo através de uma matriz de adjacência.

2) Um grafo consiste num conjunto de nós (ou vértices) e num conjunto de arcos (ou arestas). É correto afirmar que o grau de um nó é:
   a) o número de arcos incidentes nesse nó.
   b) um número associado ao arco, também chamado de peso.
   c) a distância entre este nó e um outro nó qualquer do grafo.
   d) a posição deste nó em relação ao nó raiz do grafo
   e) o número de pares ordenados que formam o arco.

3) Observe o Grafo abaixo:



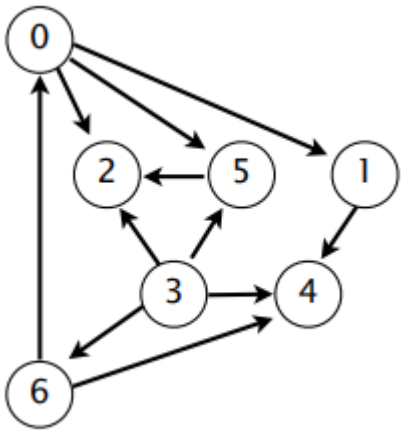
a. Utilizando o algoritmo visto em aula faça uma busca em PROFUNDIDADE a partir do vértice 0, preenchendo a tabela v, visitados[], antecessor[].

V	visitados[]	antecessor[]
0		
1		
2		
3		
4		
5		
6		
7		

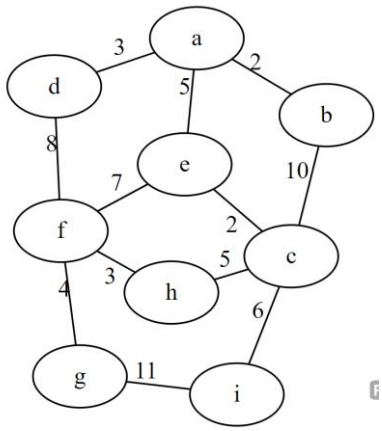
b. Utilizando o algoritmo visto em aula faça uma busca em LARGURA a partir do vértice 0, preenchendo a tabela v, visitados[], antecessor[], distancia[].

V	visitados[]	antecessor[]	distancia[]
0			
1			
2			
3			
4			
5			
6			
7			

4) Mostre o resultado da ordenação topológica do grafo abaixo:
(Dica: Realizar um caminhamento em pré-ordem com busca em profundidade e retornar o caminho invertido.)

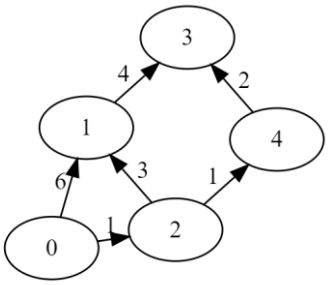


5) Dado o grafo abaixo apresente a árvore geradora mínima utilizando o algoritmo de KRUSKAL, iniciando pelo vértice a. (slides aula 24)




```
graph G {
        a -- b [ label="2" ]
        b -- c [ label="10"]
        a -- e [ label="5"]
        a -- d [ label="3"]
        d -- f [ label="8"]
        f -- g [ label="4"]
        c -- h [ label="5"]
        e -- f [ label="7"]
        c -- i [ label="6"]
        i -- g [ label="11"]
        c -- e [ label="2"]
        h -- f [ label="3"]
}
```

6) Utilizando o algoritmo de Dijkstra e tendo como ponto de partida o vértice 0 apresente o menor caminho para todos os vértices. Utilize a tabela auxiliar visto em aula para execução passo a passo.
(slides aula 26)



7) Dada a classe BuscaEmProfundidade abaixo, implemente os métodos construtor e dfs.

```
public class BuscaEmProfundidade {

    private boolean[] visitado;
    private int[] antecessor;
    private int verticeOrigem;

    public BuscaEmProfundidade (Grafo g, int verticeOrigem) {
              //….implementar
        dfs(g, verticeOrigem);
    }

    private void dfs(Grafo g, int vertice) {
        this.visitado[vertice] = true;
          //implementar
    }

}
```

8) Complete a implementação do método realizarBusca da classe BuscaEmLargura abaixo:

```
import java.util.LinkedList;
import java.util.Queue;

public class BuscaEmLargura {

    private boolean[] marked;
    private int[] edgeTo;
    private int[] distTo;

    public BuscaEmLargura(Graph G, int s) {
        marked = new boolean[G.V()];
        edgeTo = new int[G.V()];
        distTo = new int[G.V()];
        for(int i=0;i<G.V();i++) distTo[i] = Integer.MAX_VALUE;

        realizarBusca(G, s);
    }
    private void realizarBusca(Graph G, int s) {
        Queue<Integer> q = new LinkedList<>();
            //complementar
    }
}
```