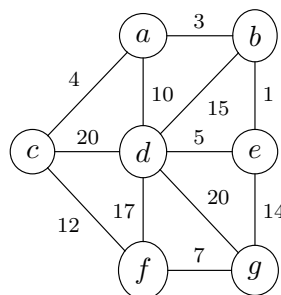


Práctica No. 6 (Grafos)

1. Dado el siguiente grafo dirigido con costos implementado con una matriz de adyacencias:

$$\begin{bmatrix} 1 & 3 & nil & 7 & 8 & nil \\ nil & nil & nil & 18 & nil & nil \\ 7 & nil & 1 & 3 & nil & 15 \\ 1 & 4 & 2 & nil & 7 & 8 \\ nil & 5 & nil & nil & 6 & 9 \end{bmatrix}$$

- Obtenga la representación con listas de adyacencias.
 - Aplique Depth-First Search al grafo
 - Aplique Breadth-First Search al grafo.
2. Implementar la clase **Grafo**, con las siguientes operaciones:
- Crear un grafo vacío.
 - Decir si un grafo es vacío.
 - Dar el número de vértices de un grafo.
 - Dar el número de arcos de un grafo.
 - Determinar si hay un arco entre dos nodos dados.
 - Insertar un nodo en un grafo.
 - Insertar un arco entre dos nodos.
 - Borrar un vértice del grafo.
 - Borrar un arco del grafo.
 - Decir si un nodo pertenece al grafo.
3. Implementar el Depth-First Search y el Breadth-First Search en la clase **Graph**.
4. Modifique el algoritmo DFS para hacer un algoritmo que cuente la cantidad de **caminos simples** entre dos nodos dados en un grafo dirigido. Cuál es el tiempo de ejecución de su algoritmo.
5. Implementar el Algoritmo de Warshall y Dijkstra en la clase **Graph**
6. Haga un algoritmo que, dado un grafo no dirigido, diga si el grafo es conexo. Cuál es el tiempo de ejecución de su algoritmo?
7. Dado un grafo dirigido con costos. Proponga un algoritmo que detecte si hay ciclos con costos negativos. Cuál es el tiempo de ejecución del algoritmo?
8. Considere el siguiente grafo no dirigido y conexo:



- Utilice el algoritmo de Prims para obtener un árbol abarcador mínimo.
- Utilice el algoritmo de Kruskal para obtener un árbol abarcador mínimo.