MRI Programación III – Cursada 2022
Trabajo Práctico Especial

DFS y BFS

Integrantes:
Borsoi, Walter 1078362
Smith, Brenda 1079984

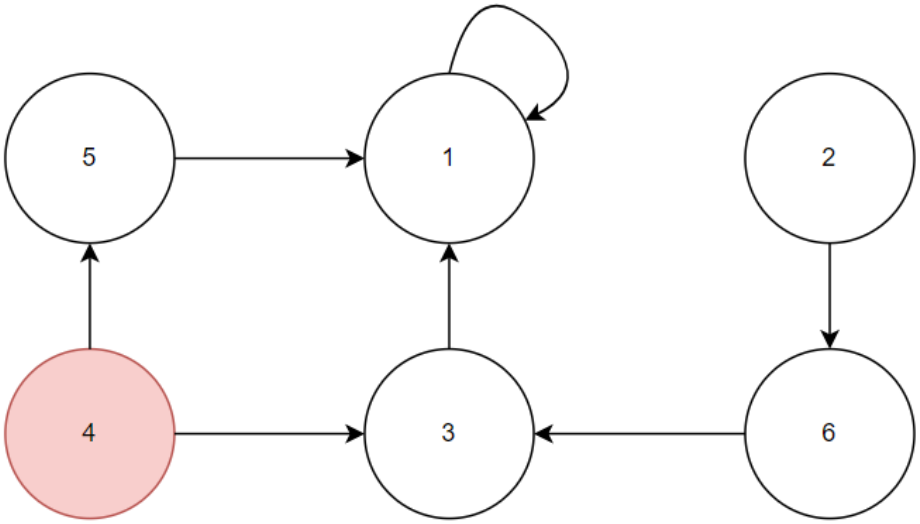
Profesor:
Rodriguez, Guillermo Horacio

MRI Pinamar

Elegimos dos de los seis algoritmos candidatos los cuales son:

- DFS: **Depth-First-Search (DFS) – Búsqueda en profundidad:**
- BFS: **Algoritmo Breath-First- Search (BFS) - Búsqueda en anchura**

  Ejemplo de grafo para ambos algoritmos:



Se detallan a continuación:

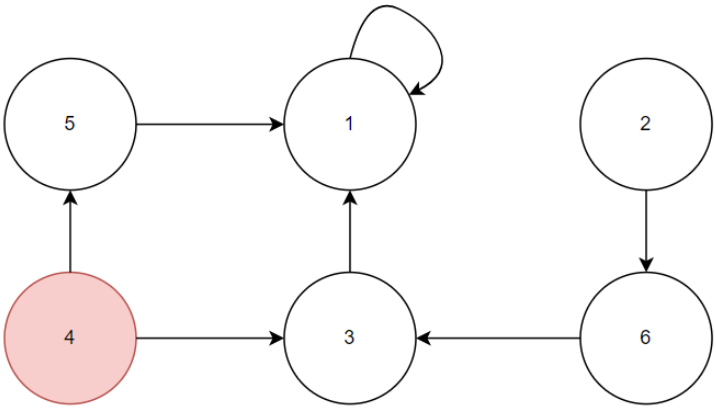
1. Algoritmo Depth-First-Search (DFS) – Búsqueda en profundidad:

DFS.java

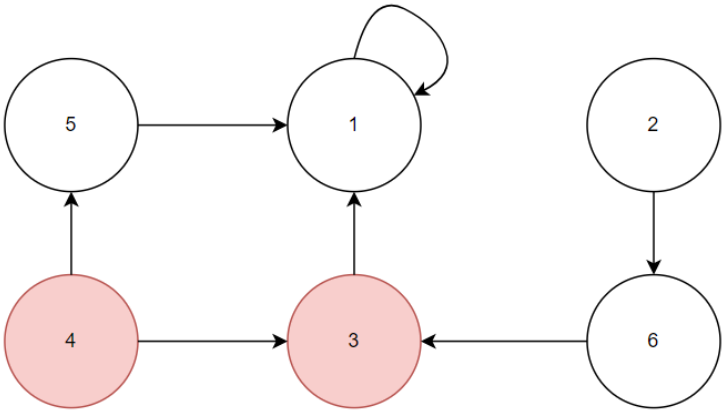
```java
package algoritmos;
import java.util.*;

public class DFS {
    static void DFSRecursivo(Grafo grafo, int nodo, boolean visitado[]){
        visitado[nodo] = true;
        System.out.print("|" + nodo + "| ");;

        Iterator<Integer> aux = grafo.adyacentes[nodo].listIterator();
        while (aux.hasNext()) {
            int n = aux.next();
            if (!visitado[n])
                DFSRecursivo(grafo, n, visitado);
        }
    }

    public static void BusquedaProfundidad(Grafo grafo, int v){
        boolean visited[] = new boolean[grafo.V];

        DFSRecursivo(grafo, v, visited);
    }
}

```

Entradas del método: grafo y nodo
Salidas del método: nodos alcanzados desde el origen.

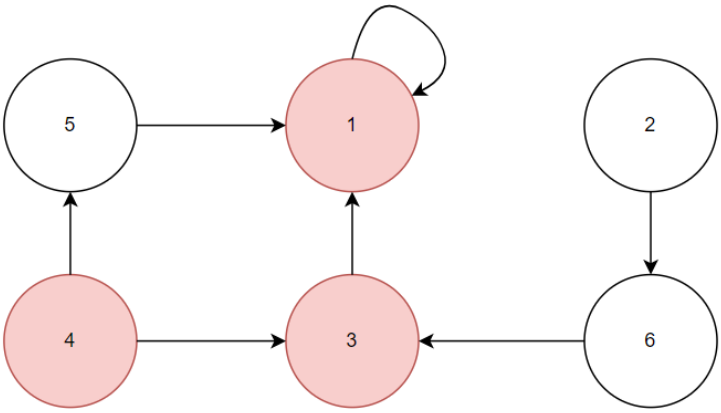
Solución:
Paso 1:



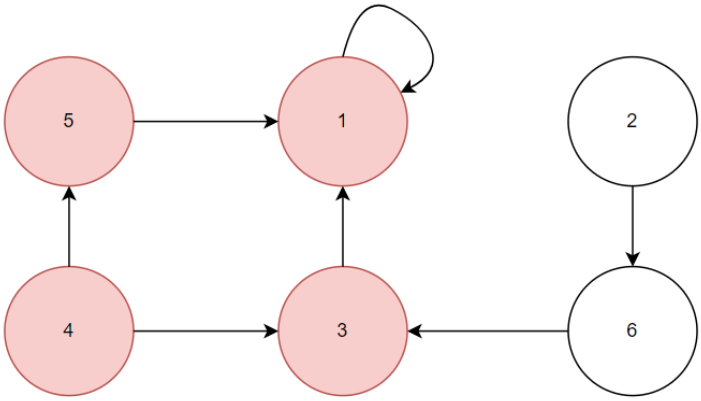
Paso 2:



Paso 3:



Paso 4:



Salida por pantalla:

```
DFS partiendo del nodo 1
|4| |3| |1| |5|
```

2. Algoritmo Breath-First- Search (BFS) - Búsqueda en anchura

Desde cada vértice consideramos que todos los vértices son accesibles desde él.

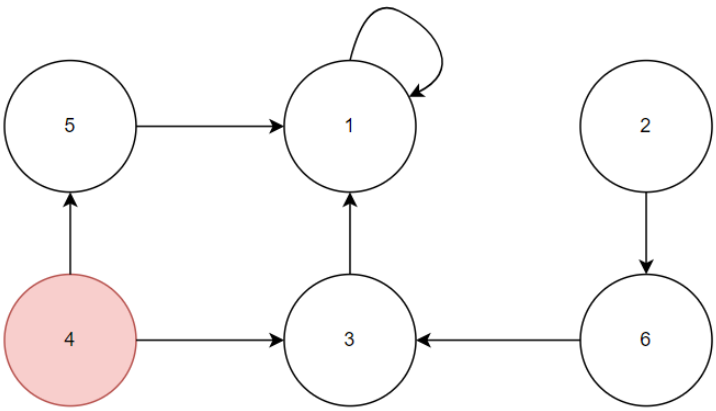
*BFS.java

```java
package algoritmos;
import grafos.*;

public class BFS {
    public static void BusquedaAnchura(Grafo grafo, int origen) {
        boolean visitado[] = new boolean[grafo.V];

        LinkedList<Integer> cola = new LinkedList<Integer>();

        visitado[origen] = true;
        cola.add(origen);
        while (cola.size() != 0){
            origen = cola.poll();
            System.out.print("|" + origen + "| ");

            Iterator<Integer> aux = grafo.adyacentes[origen].listIterator();
            while (aux.hasNext()){
                int nodo = aux.next();
                 if (!visitado[nodo]){
                    visitado[nodo] = true;
                    cola.add(nodo);
                }
            }
        }
    }
}
```

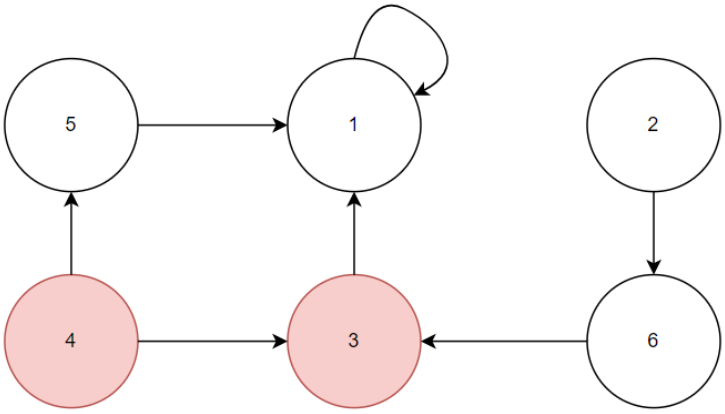
Entradas del método: grafo y origen.
Salidas del método: nodos alcanzados desde el origen.

Solución:

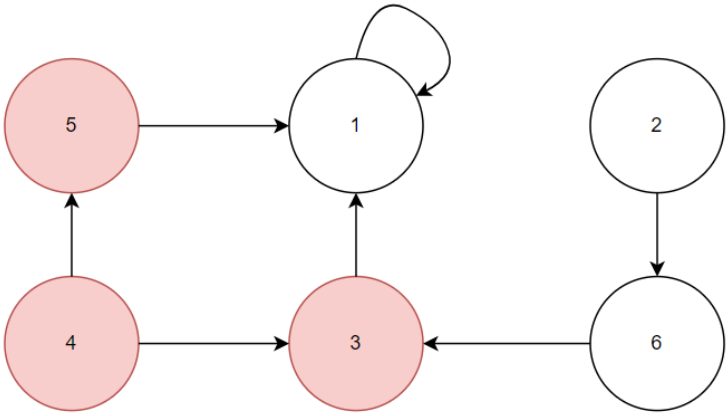
Paso 1:



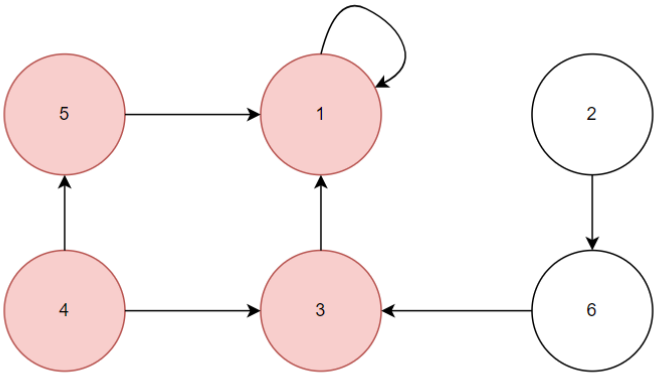
Paso 2:



Paso 3:



Paso 4:



Salida por pantalla:

```
BFS partiendo del nodo 2
|4| |3| |5| |1|
```