

# **MRI Programación III – Cursada 2022**

## **Trabajo Práctico Especial**

### ***DFS y BFS***

Integrantes:

Borsoi, Walter 1078362

Smith, Brenda 1079984

Profesor:

Rodriguez, Guillermo Horacio

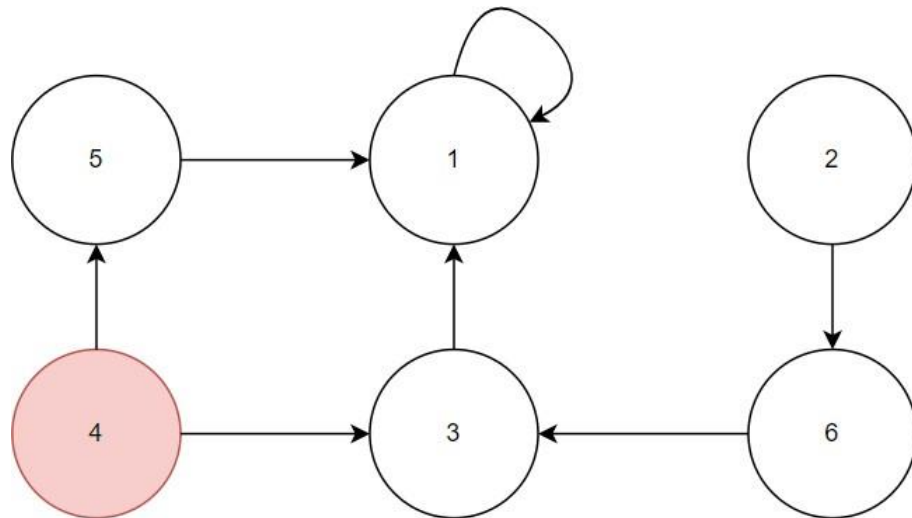
MRI Pinamar



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 ×
1 package algoritmos;
2 import java.util.*;
3
4
5
6 public class DFS {
7     static void DFSRecursivo(Grafo grafo, int nodo, boolean visitado[]){
8         visitado[nodo] = true;
9         System.out.print("|" + nodo + " | ");
10
11         Iterator<Integer> aux = grafo.adyacentes[nodo].listIterator();
12         while (aux.hasNext()) {
13             int n = aux.next();
14             if (!visitado[n])
15                 DFSRecursivo(grafo, n, visitado);
16         }
17     }
18
19     public static void BusquedaProfundidad(Grafo grafo, int v){
20         boolean visited[] = new boolean[grafo.V];
21
22         DFSRecursivo(grafo, v, visited);
23     }
24 }
25

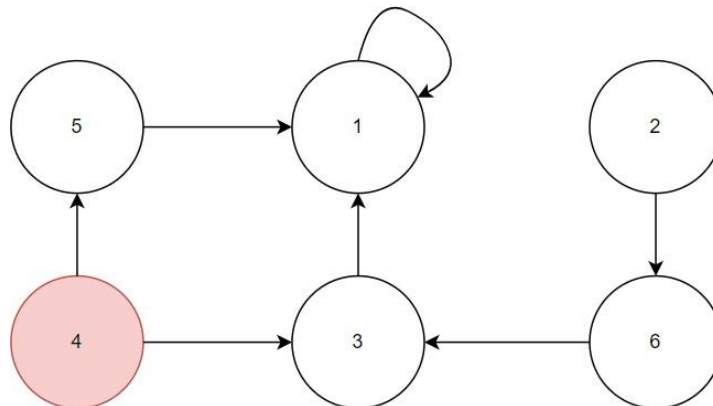
```

**Entradas del método:** grafo y nodo

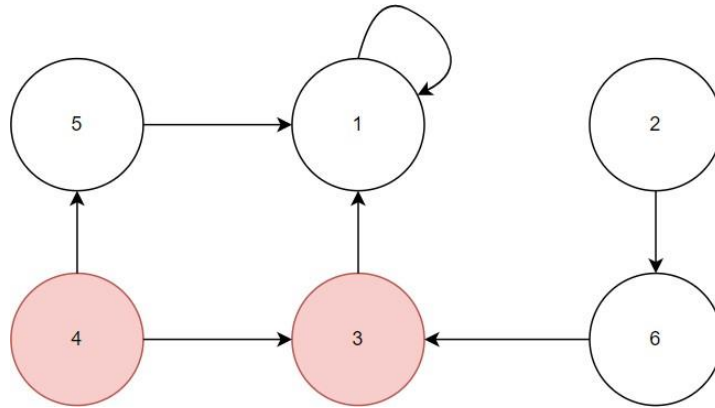
**Salidas del método:** nodos alcanzados desde el origen.

**Solución:**

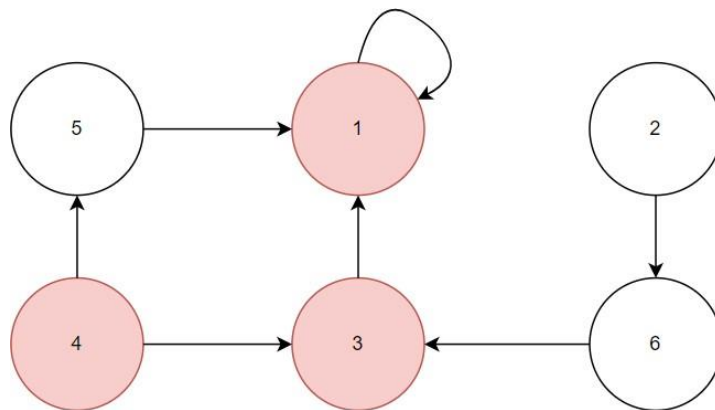
**Paso 1:**



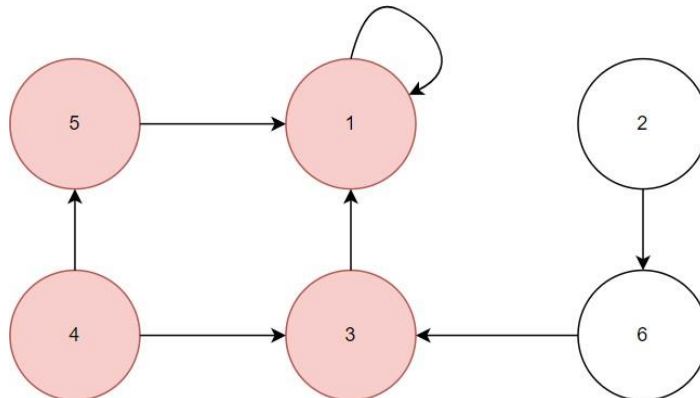
**Paso 2:**



**Paso 3:**



**Paso 4:**



**Salida por pantalla:**

```
DFS partiendo del nodo 4  
|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 ×
1 package algoritmos;
2 import grafos.*;
3
4
5 public class BFS {
6     public static void BusquedaAnchura(Grafo grafo, int origen) {
7         boolean visitado[] = new boolean[grafo.V];
8
9         LinkedList<Integer> cola = new LinkedList<Integer>();
10
11         visitado[origen] = true;
12         cola.add(origen);
13         while (cola.size() != 0){
14             origen = cola.poll();
15             System.out.print("|" + origen + "| ");
16
17             Iterator<Integer> aux = grafo.adyacentes[origen].listIterator();
18             while (aux.hasNext()){
19                 int nodo = aux.next();
20                 if (!visitado[nodo]){
21                     visitado[nodo] = true;
22                     cola.add(nodo);
23                 }
24             }
25         }
26     }
27 }
28
```

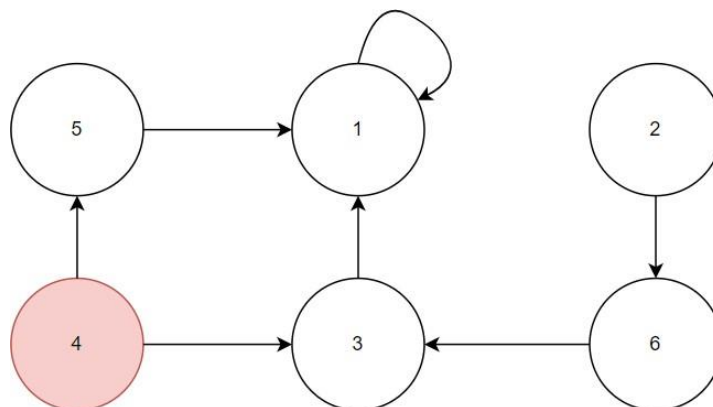
**Entradas del método:** grafo y origen.

**Estructura auxiliar:** cola.

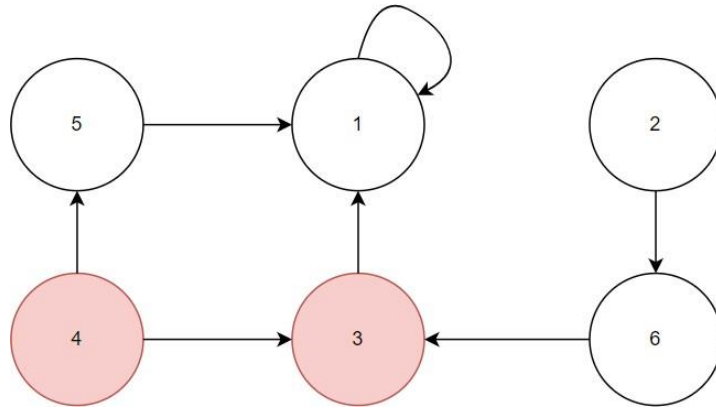
**Salidas del método:** nodos alcanzados desde el origen.

**Solución:**

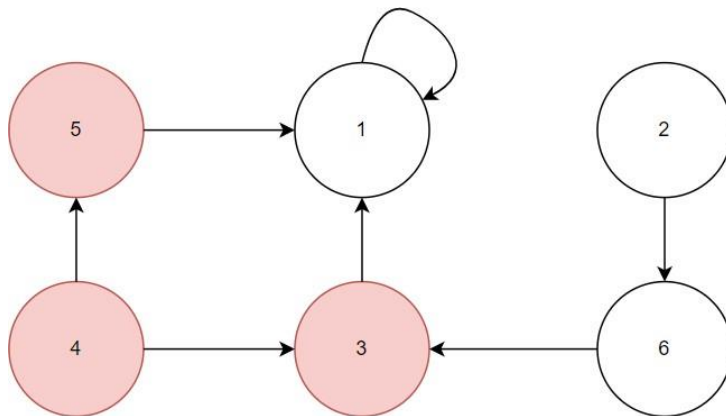
**Paso 1:**



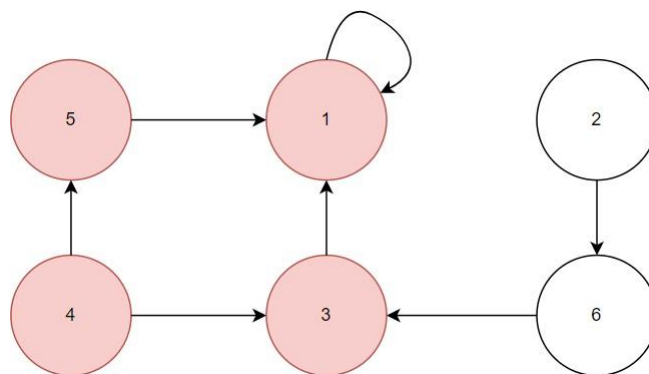
**Paso 2:**



**Paso 3:**



**Paso 4:**



**Salida por pantalla:**

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