Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Campus Sonora Norte

Act 4.1 - Grafo: sus representaciones y sus recorridos

Curso:

Programación de estructuras de datos y algoritmos fundamentales

(TC1031)

Estudiante:

Daniel Alfredo Barreras Meraz A01254805

Docente:

Baldomero Olvera Villanueva

Fecha de entrega:

11 de noviembre de 2023

Código fuente

- Adjunto en el archivo zip.

Casos de prueba

Caso de prueba	Entrada	Resultados		
		loadGraph	DFS	BFS
1	{{0, 1}, {3, 4}, {4, 5}, {6, 7}, {7, 8}, {0, 3}, {1, 4}, {2, 5}, {3, 6}, {4, 7}, {5, 8}}	Lista de adyacencia: 0: 1 3 1: 0 4 2: 5 3: 4 0 6 4: 3 5 1 7 5: 4 2 8 6: 7 3 7: 6 8 4 8: 7 5	0 1 4 3 6 7 8 5 2	0 1 3 4 6 5 7 2 8
2	{{0, 3}, {1, 2}, {1, 4}, {2, 5}, {4, 5}, {4, 6}, {5, 7}, {6, 7}}	Lista de adyacencia: 0: 3 1: 2 4 2: 1 5 3: 0 4: 1 5 6 5: 2 4 7 6: 4 7 7: 5 6	0 3	0 3

3	{{0, 1}, {1, 2}, {0, 3}, {1, 4}, {2, 5}, {3, 4}, {4, 5}}	Lista de adyacencia: 0: 1 3 1: 0 2 4 2: 1 5 3: 0 4 4: 1 3 5 5: 2 4	0 1 2 5 4 3	0 1 3 2 4 5
4	{{0, 1}, {1, 3}, {0, 2}, {2, 3}, {3, 4}, {3, 5}, {4, 6}, {5, 6}}	Lista de adyacencia: 0: 1 2 1: 0 3 2: 0 3 3: 1 2 4 5 4: 3 6 5: 3 6 6: 4 5	0 1 3 2 4 6 5	1 2 3 4 5 6

Comprobación de casos de prueba

```
Matriz de adyacencia:
0: 0 1 0 1 0 0 0 0 0
1: 1 0 0 0 1 0 0 0 0
2: 0 0 0 0 0 1 0 0 0
3: 1 0 0 0 1 0 1 0 0
4: 0 1 0 1 0 1 0 1 0
5: 0 0 1 0 1 0 0 0 1
6: 0 0 0 1 0 0 0 1 0
7: 0 0 0 0 1 0 1 0 1
8: 0 0 0 0 0 1 0 1 0

Lista de adyacencia:
0: 1 3
1: 0 4
2: 5
3: 4 0 6
4: 3 5 1 7
5: 4 2 8
6: 7 3
7: 6 8 4
8: 7 5
Recorrido DFS a partir del nodo 0: 0 1 4 3 6 7 8 5 2
Recorrido BFS a partir del nodo 0: 0 1 3 4 6 5 7 2 8
```

```
Matriz de adyacencia:
0: 0 0 0 1 0 0 0 0
1: 0 0 1 0 1 0 0 0
2: 0 1 0 0 0 1 0 0
3: 1 0 0 0 0 0 0 0
4: 0 1 0 0 0 1 1 0
5: 0 0 1 0 1 0 0 1
6: 0 0 0 0 1 0 0 1
7: 0 0 0 0 0 1 1 0

Lista de adyacencia:
0: 3
1: 2 4
2: 1 5
3: 0
4: 1 5 6
5: 2 4 7
6: 4 7
7: 5 6
Recorrido DFS a partir del nodo 0: 0 3
Recorrido BFS a partir del nodo 0: 0 3
```

```
Matriz de adyacencia:
0: 0 1 0 1 0 0
1: 1 0 1 0 1 0
2: 0 1 0 0 0 1
3: 1 0 0 0 1 0
4: 0 1 0 1 0 1
5: 0 0 1 0 1 0

Lista de adyacencia:
0: 1 3
1: 0 2 4
2: 1 5
3: 0 4
4: 1 3 5
5: 2 4
Recorrido DFS a partir del nodo 0: 0 1 2 5 4 3
Recorrido BFS a partir del nodo 0: 0 1 3 2 4 5
```

```
Matriz de adyacencia:
0: 0 1 1 0 0 0 0
1: 1 0 0 1 0 0 0
2: 1 0 0 1 0 0 0
3: 0 1 1 0 1 1 0
4: 0 0 0 1 0 0 1
5: 0 0 0 1 0 0 1
6: 0 0 0 0 1 1 0

Lista de adyacencia:
0: 1 2
1: 0 3
2: 0 3
3: 1 2 4 5
4: 3 6
5: 3 6
6: 4 5
Recorrido DFS a partir del nodo 0: 0 1 3 2 4 6 5
Recorrido BFS a partir del nodo 0: 0 1 2 3 4 5 6 %
```

Complejidad temporal

loadGraph: O(n^2) - Esta función carga los arcos del grafo y los almacena en una Matriz de Adyacencia y en una Lista de Adyacencia. Como cada vértice puede estar conectado con todos los demás, en el peor de los casos, la función debe recorrer todos los pares de vértices, lo que da una complejidad de tiempo de O(n^2).

DFS: O(n + m) - Esta función realiza un recorrido en profundidad (DFS) a partir de un nodo inicial. En el peor de los casos, la función debe visitar todos los vértices y todos los arcos del grafo, lo que da una complejidad de tiempo de O(n + m), donde n es el número de vértices y m es el número de arcos.

BFS: O(n + m) - Esta función realiza un recorrido en anchura (BFS) a partir de un nodo inicial. Al igual que con la función DFS, en el peor de los casos, la función debe visitar todos los vértices y todos los arcos del grafo, lo que da una complejidad de tiempo de O(n + m).