Informe N° 4: BFS

Gianfranco Pupiales

2024-05-25

Tabla de Contenidos

1. Objetivos

- Distinguir los conceptos fundamentales del algoritmo de búsqueda en amplitud (BFS).
- Implementar el algoritmo de búsqueda en amplitud (BFS) utilizando Python.

2. Introducción

El algoritmo de búsqueda en amplitud (BFS) es una técnica de recorrido de grafos que explora todos los vértices en un grafo en el nivel de profundidad actual antes de pasar a los vértices en el siguiente nivel de profundidad. Comienza en un vértice especificado y visita todos sus vecinos antes de pasar al siguiente nivel de vecinos. BFS se utiliza comúnmente en algoritmos para encontrar caminos, componentes conectados y problemas de camino más corto en grafos («Breadth First Search or BFS for a Graph - GeeksforGeeks — geeksforgeeks.org»).

3. Desarrollo

3.1 Enunciado ejercicio 1

Dado el siguiente grafo, implemente el algortimo BFS donde se responda a las preguntas:

1. ¿Existe un path desde S hasta F?

   Sí.

2. ¿Cuál es esa ruta?

   La ruta utilizanddo BFS es CAR -> CAT -> BAT.

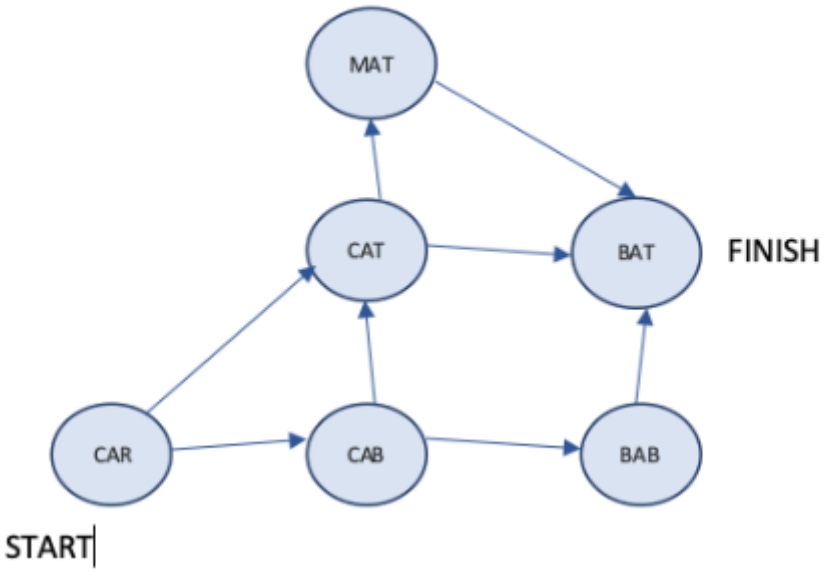

Figura 1: Grafo

```
from collections import deque

graph = {
    "CAR": ["CAT", "CAB"],
    "CAT": ["MAT", "BAT"],
    "CAB": ["CAT", "BAB"],
    "MAT": ["BAT"],
    "BAT": [],
    "BAB": ["BAT"]
```

```
}


def is_that(person):
    return person == "BAT"


def search(name):
    search_queue = deque()
    # Tupla que almacena el nodo actual y la ruta hasta este nodo
    search_queue += [(name, [name])]
    searched = []
    while search_queue:
        person, path = search_queue.popleft()
        if person not in searched:
            if is_that(person):
                print(person + " is that person!")
                print("Path:", ' -> '.join(path))
                return True
            else:
                for neighbor in graph[person]:
                    # Agregar el vecino y la nueva ruta
                    search_queue.append((neighbor, path + [neighbor]))
                searched.append(person)
    return False


search("CAR")
```

```
BAT is that person!
Path: CAR -> CAT -> BAT

True
```

3.2 Enunciado ejercicio 2

Genere el árbol BFS (BFS Tree) para el siguiente grafo G, sabiendo que el nodo de inicio es 0. Ese sería el componente conectado que contiene al nodo 0. Implemente el algoritmo BFS y use las estructuras auxiliares necesarias.

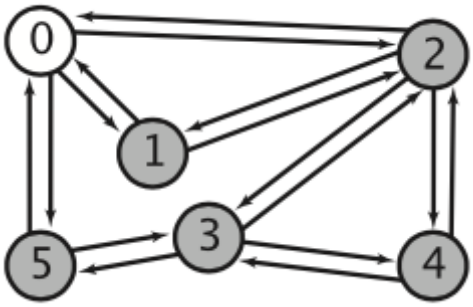

Figura 2: Grafo G

```python
from collections import deque

graph_G = {
    0: [2, 1, 5],
    1: [0, 2],
    2: [0, 1, 3, 4],
    3: [5, 4, 2],
    4: [3, 2],
    5: [3, 0]
}


def bfs_tree(graph, start):
    # Estructura para almacenar el árbol BFS
    tree = {}
    # Conjunto para mantener un registro de nodos visitados
    visited = set()
    # Iniciar la cola de búsqueda con el nodo de inicio y su padre (None para el nodo raíz)
    queue = deque([(start, None)])

    while queue:
        node, parent = queue.popleft()
        if node not in visited:
            visited.add(node)
            if parent is not None:
                # Agregar el nodo como hijo del padre en el árbol
                tree.setdefault(parent, []).append(node)
            for neighbor in graph[node]:
                if neighbor not in visited:
```

```
                    # Agregar vecinos a la cola con el nodo actual como padre
                    queue.append((neighbor, node))
    return tree


start_node = 0
bfs_tree_result = bfs_tree(graph_G, start_node)
print("BFS Tree from node", start_node, ":", bfs_tree_result)
```

```
BFS Tree from node 0 : {0: [2, 1, 5], 2: [3, 4]}
```

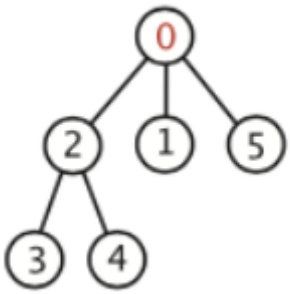

Figura 3: Árbol BFS con nodo de inicio 0

4. Conclusiones

- El método de búsqueda en amplitud resulta eficaz para hallar una ruta desde un nodo inicial hasta uno objetivo en un grafo. Emplea una cola para explorar todos los nodos adyacentes antes de avanzar hacia los adyacentes de esos nodos. En el caso de que exista el camino, este enfoque asegura la identificación del más corto.
- El árbol BFS creado a partir de un nodo inicial en un grafo ofrece una visualización clara de cómo los nodos están conectados en una jerarquía desde el nodo de partida.

5. Declaración del Uso de IA

En este informe se utilizó IA para mejorar la coherencia, cohesión y sintaxis de las conclusiones.

6. Referencias Bibliográficas

«Breadth First Search or BFS for a Graph - GeeksforGeeks — geeksforgeeks.org». https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/.