Indian Institute of Information Technology Surat

Lab Report on
Artificial Intelligence (CS 701) Practical

Submitted by

[RAHUL KUMAR SINGH] (UI21CS44)

Course Faculty

Dr. Ritesh Kumar

Mrs. Archana Balmik

Department of Computer Science and Engineering
Indian Institute of Information Technology Surat
Gujarat-394190, India

Aug-2024

Lab No: 6

Aim:

To implement DFS and BFS using PROLOG code.

Description:

Depth-First Search (DFS)

- Explores as far as possible along each branch before backtracking.
- Uses a stack or recursion for traversal.

Breadth-First Search (BFS)

- Explores all neighbors at the current depth level before moving deeper.
- Uses a queue for traversal.

Code:

A)DFS

```
% DFS
edge(a, b).
edge(a, c).
edge(b, d).
edge(b, e).
edge(c, f).
edge(c, g).

dfs(Node) :-
    dfs_util(Node, []).

dfs_util(Node, Visited) :-
    \+ member(Node, Visited),
    write(Node), write(' '),
    findall(Neighbor, edge(Node, Neighbor), Neighbors),
    dfs_neighbors(Neighbors, [Node | Visited]).

dfs_neighbors([], _).
dfs_neighbors([Neighbor | Rest], Visited) :-
    dfs_util(Neighbor, Visited),
    dfs_neighbors(Rest, Visited).
```

B)BFS

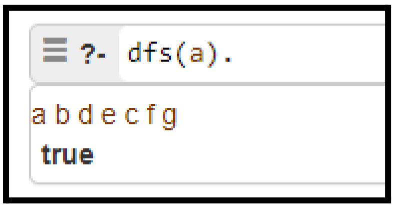
```
% BFS
edge(a, b).
edge(a, c).
edge(b, d).
edge(b, e).
edge(c, f).
edge(c, g).

bfs(Start) :-
    bfs([Start], []).

bfs([], _).
bfs([Node|Queue], Visited) :-
    \+ member(Node, Visited),
    write(Node), write(' '),
    findall(Neighbor, edge(Node, Neighbor), Neighbors),
    append(Queue, Neighbors, NewQueue),
    bfs(NewQueue, [Node|Visited]).
bfs([_|Queue], Visited) :-
    bfs(Queue, Visited).
```

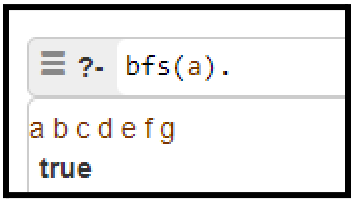
Output:

A)DFS



```
?- dfs(a).
a b d e c f g
true
```

B)BFS



```
?- bfs(a).
a b c d e f g
true
```

Conclusion:

- The functions are named `dfs/1` and `bfs/1`, initiating depth-first and breadth-first searches, respectively.
- DFS explores deeply along branches, while BFS processes nodes level by level.
- Temporary variables and lists are used to track visited nodes and manage the queue.
- These traversal algorithms are practical for graph-related problems and demonstrate Prolog's strengths in symbolic reasoning.