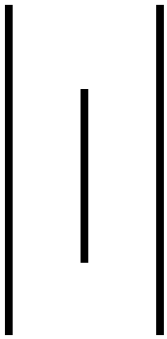
TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

DHARAN



ARTIFICIAL INTELLIGENCE

Lab Report III

Submitted by:

Gokarna Baskota
PUR075BEI013

Submitted To:

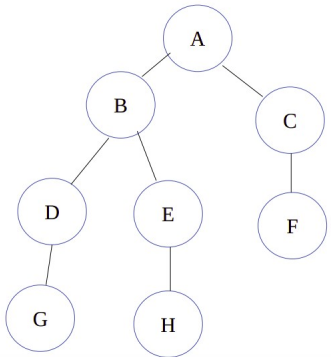
Department of Electronics and Computer Engineering

Lab 3: Problem Solving by Searching: DFS and BFS

An uninformed search algorithm generates the search tree without using any domain specific knowledge. Uninformed search algorithms are also called blind search algorithms. The search algorithm produces the search tree without using any domain knowledge, which is a brute force in nature. They don't have any background information on how to approach the goal or whatsoever. But these are the basics of search algorithms in AI. In this lab we will implement the following searching strategies in the graphs shown in figure below.

Depth First Search (DFS): It is a search algorithm where the search tree will be traversed from the root node. It will be traversing, searching for a key at the leaf of a particular branch. If the key is not found the searching retraces its steps back to the point from where the other branch was left unexplored and the same procedure is repeated for that other branch.

Breadth First Search(BFS): It is another search algorithm in AI which traverses breadth-wise to search the goal in a tree. It begins searching from the root node and expands the successor node. It further expands along breadth-wise and traverses those nodes rather than searching depth-wise.



Our goal is to find a path from node A to node H

Program:

```
def DFS(graph, start, target):
    visited = []
    stack = []
    prev = dict()
    stack.append(start)

    while len(stack) > 0:
        popped = stack.pop()
        if popped not in visited:
            if popped == target:
                return (True, prev)
            visited.append(popped)

            for vertex in graph[popped]:
                if vertex not in visited and vertex not in stack:
                    stack.append(vertex)
                    prev[vertex] = popped
    return (False, prev)

def BFS(graph, start, target):
    visited = []
    queue = []
```

```
    prev = dict()
    queue.append(start)

    while len(queue) > 0:
        popped = queue.pop(0)
        if popped not in visited:
            if popped == target:
                return (True, prev)
            visited.append(popped)

            for vertex in graph[popped]:
                if vertex not in visited and vertex not in queue:
                    queue.append(vertex)
                    prev[vertex] = popped
    return (False, prev)

def draw_path(path, start, goal):
    map_ = goal
    temp = goal

    while path[temp] != start:
        map_ = path[temp] + "-->" + map_
        temp = path[temp]
    return start + "-->" + map_

graph = {
        'A': ['B', 'C'],
        'B': ['D', 'E', 'A'],
        'C': ['F', 'A'],
        'D': ['G', 'B'],
        'E': ['H', 'B'],
        'F': ['C'],
        'G': ['D'],
        'H': ['E']
        }

start = 'A'
goal = 'H'

done, path = DFS(graph, start, goal)
print(done)
print(draw_path(path, start, goal))

done, path = BFS(graph, start, goal)
print(done)
print(draw_path(path, start, goal))
```

Output: