

INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG



Lab report-3

Course code: CSE-3636

Course Title : Artificial Intelligence Lab

Submitted To:

Md Safayet Hossen

Department of CSE

International Islamic University Chittagong

Submitted By:

Md.Riaz Ahmed

Id: C201060

Section: 6BM

Semester: 6th

Date of submission : 25/03/23

BFS and DFS:

```
DFS.py x
1 from collections import deque, # dictionary
2 def bfs(graph, start):
3     visited = set() # Keep track of visited nodes
4     queue = deque([start]) # We create a deque data structure to use as our queue for the BFS traversal.
5     # We add the starting node to the queue.
6
7     while queue: # We enter a loop that continues until the queue is empty.
8         vertex = queue.popleft()
9         """ In each iteration of the loop, we take the next node
10            from the left end of the queue (using the popleft() method) We mark this node as visited."""
11         if vertex not in visited: # Add the node to the queue and mark it as visited
12             visited.add(vertex)
13             queue.extend(graph[vertex] - visited)
14     return visited # Return the visited nodes
15
16 def dfs(graph, start, visited=None):
17     if visited is None: # We create a set to keep track of visited nodes
18         visited = set() # If the visited argument is not provided, we create a new set.
19     visited.add(start) # We mark the current node as visited.
20     """ We loop through all the adjacent nodes of the current node.
21         If a neighbor has not been visited yet we recursively call the dfs function
22         with the neighbor as the new starting node and the visited set as a parameter."""
23
24     for next in graph[start] - visited:
25         dfs(graph, next, visited)
26     return visited
```

```
graph = {
    'A': set(['B', 'C']),
    'B': set(['A', 'D', 'E']),
    'C': set(['A', 'F']),
    'D': set(['B']),
    'E': set(['B', 'F']),
    'F': set(['C', 'E'])
}

print(bfs(graph, 'A')) # {'A', 'C', 'B', 'E', 'D', 'F'}
print(dfs(graph, 'A')) # {'A', 'C', 'F', 'E', 'B', 'D'}
```

Code:

```
from collections import deque # dictionary
```

```
def bfs(graph, start):
```

```

visited = set() # Keep track of visited nodes

queue = deque([start]) #We create a deque data structure to use as our queue for the
BFS traversal.

# We add the starting node to the queue.

while queue: #We enter a loop that continues until the queue is empty.
    vertex = queue.popleft()
    """ In each iteration of the loop, we take the next node
    from the left end of the queue (using the popleft() method) We mark this node as
    visited."""
    if vertex not in visited: # Add the node to the queue and mark it as visited
        visited.add(vertex)
        queue.extend(graph[vertex] - visited)
return visited # Return the visited nodes

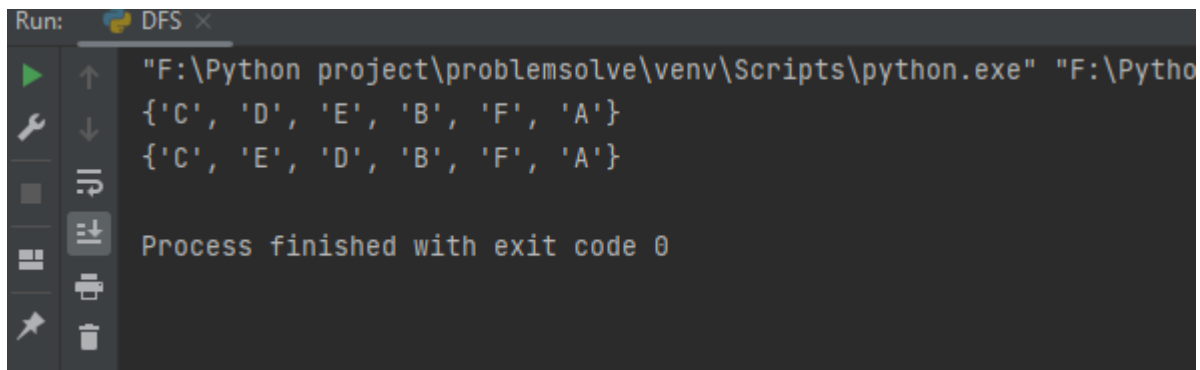
def dfs(graph, start, visited=None):
    if visited is None: # We create a set to keep track of visited nodes
        visited = set() # If the visited argument is not provided, we create a new set.
    visited.add(start) # We mark the current node as visited.
    """ We loop through all the adjacent nodes of the current node.
    If a neighbor has not been visited yet we recursively call the dfs function
    with the neighbor as the new starting node and the visited set as a parameter."""

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited

```

```
graph = {  
    'A': set(['B', 'C']),  
    'B': set(['A', 'D', 'E']),  
    'C': set(['A', 'F']),  
    'D': set(['B']),  
    'E': set(['B', 'F']),  
    'F': set(['C', 'E'])  
}  
  
print(bfs(graph, 'A')) # {'A', 'C', 'B', 'E', 'D', 'F'}  
print(dfs(graph, 'A')) # {'A', 'C', 'F', 'E', 'B', 'D'}
```

Output:



The screenshot shows a terminal window titled "Run: DFS x". The command prompt is "F:\Python project\problemsolve\venv\Scripts\python.exe". The output of the BFS algorithm is shown as a set: {'C', 'D', 'E', 'B', 'F', 'A'}. The output of the DFS algorithm is shown as a set: {'C', 'E', 'D', 'B', 'F', 'A'}. The terminal also shows "Process finished with exit code 0".

```
Run: DFS x  
"F:\Python project\problemsolve\venv\Scripts\python.exe" "F:\Python project\problemsolve\venv\Scripts\python.exe"  
{'C', 'D', 'E', 'B', 'F', 'A'}  
{'C', 'E', 'D', 'B', 'F', 'A'}  
Process finished with exit code 0
```