**Exercise 4**

**Depth First Search (DFS) and Breadth First Search (BFS)**

**Title:** Implementation and Analysis of DFS and BFS for an application

**Problem Description:** A web crawler bot is like something to search the World Wide Web automatically for Web indexing. The problem here is to show how the DFS and BFS traverse through a simple web page.

**Solution:**   
The idea is to start from source page and follow all links from source and keep doing same using DFS and BFS.

**Python Code using BFS:**

from time import time

graph = {

  'Homepage' : ['AboutAuthor','RecipesIndex'],

  'AboutAuthor': ['Summary','Contact'],

  'Summary': [],

  'Contact': [],

  'RecipesIndex' : ['Veg'],

  'Veg' : ['BreakfastIndex','LunchIndex','DinnerIndex'],

  'BreakfastIndex' : ['Idli','Dosa'],

  'LunchIndex' : ['RiceVariety','sambar','Curd'],

  'DinnerIndex' : ['Chappathi','Naan','Phulka','AlooMutterMasala'],

  'Idli':[],

  'Dosa':[],

  'RiceVariety':[],

  'sambar':[],

  'Curd':[],

  'Chappathi':[],

  'Naan':[],

  'Phulka':[],

  'AlooMutterMasala':[]

}

visited = [] # List to keep track of visited nodes.

queue = []     #Initialize a queue

def bfs(visited, graph, node):

  visited.append(node)

  queue.append(node)

  while queue:

    s = queue.pop(0)

    print (s, end = "\n")

    for neighbour in graph[s]:

      if neighbour not in visited:

        visited.append(neighbour)

        queue.append(neighbour)

# Driver Code

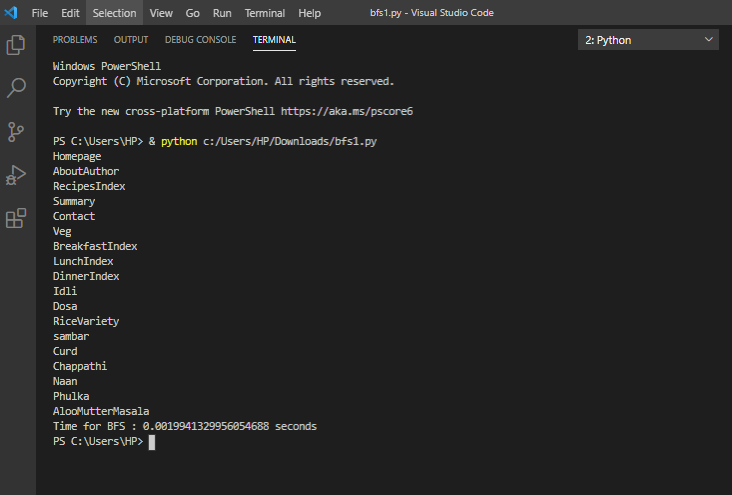
t0 = time()

bfs(visited, graph, 'Homepage')

t1 = time() - t0

print('Time for BFS :', t1, 'seconds')

**Output for BFS:**



**Python Code using DFS:**

from time import time

graph = {

  'Homepage' : ['AboutAuthor','RecipesIndex'],

  'AboutAuthor': ['Summary','Contact'],

  'Summary': [],

  'Contact': [],

  'RecipesIndex' : ['Veg'],

  'Veg' : ['BreakfastIndex','LunchIndex','DinnerIndex'],

  'BreakfastIndex' : ['Idli','Dosa'],

  'LunchIndex' : ['RiceVariety','sambar','Curd'],

  'DinnerIndex' : ['Chappathi','Naan','Phulka','AlooMutterMasala'],

  'Idli':[],

  'Dosa':[],

  'RiceVariety':[],

  'sambar':[],

  'Curd':[],

  'Chappathi':[],

  'Naan':[],

  'Phulka':[],

  'AlooMutterMasala':[]

}

visited = set() # Set to keep track of visited nodes.

def dfs(visited, graph, node):

    if node not in visited:

        print (node)

        visited.add(node)

        for neighbour in graph[node]:

            dfs(visited, graph, neighbour)

# Driver Code

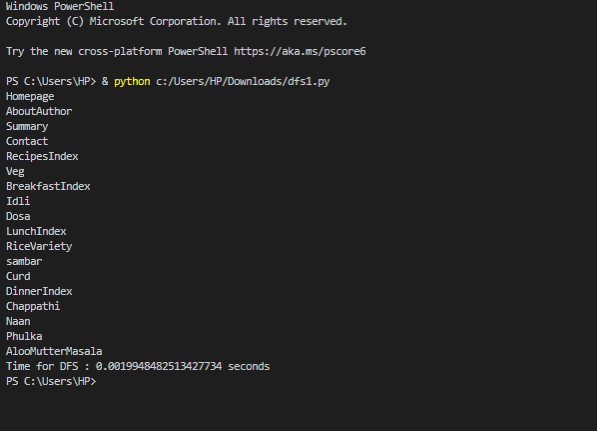
t0 = time()

dfs(visited, graph, 'Homepage')

t1 = time() - t0

print('Time for DFS :', t1, 'seconds')

**Output for DFS:**



**Analysis:**

The time complexity for DFS and BFS on a graph is O(V + E); where V is the number of vertices and E is the number of edges. From the output, it is clear that time taken to traverse the nodes using **DFS (0.0019secs)** is better than **BFS (0.0019secs).**