Samkit Shah Bhavya Shah

2019130060, 2019130056

TE Comps

Batch C

**Experiment – 2**

**Aim:** To implement the fire extinguisher system using BFS and DFS.

# Theory:

* **Depth-First-Search (DFS)**:
  1. DFS always expands DEPTH-FIRST the deepest node in the current frontier of the search tree.
  2. The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors.
  3. DFS uses a LIFO queue.
  4. Visits children before siblings.

# Breadth-First-Search (BFS):

1. BFS is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on.
2. All the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.
3. BFS uses a FIFO queue.
4. Visits siblings before children

# Code:

top\_dfs=[]

vis=[]

graph=[]

def bfs(V):

    values = [0]\*(V)

    for i in range(len(graph)):

        for j in graph[i]:

            values[j] += 1

    queue = []

    for i in range(V):

        if values[i] == 0:

            queue.append(i)

    cnt = 0

    order = []

    while queue:

        u = queue.pop(0)

        order.append(u)

        for i in graph[u]:

            values[i] -= 1

            if values[i] == 0:

                queue.append(i)

        cnt += 1

    if cnt != V:

        print("They cannot be studied because of interdependancy")

        return []

    else :

        return order

def dfs(p):

    vis[p] = True

    for i in graph[p]:

        if vis[i] == True:

            return False

        if not dfs(i):

            return False

    top\_dfs.append(p)

    return True

def main():

    x = int(input("Enter the number of subects in your course: "))

    y = dict()

    mapp=[]

    for i in range(x):

        z = input("Enter the name of your course: ")

        y[z] = i;

        vis.append(False)

        graph.append([])

        mapp.append(z)

    num = int(input("Enter the number of prerequisites pair: "))

    print("Enter the prerequisite in the form a is a prerequisite to b")

    for i in range(num):

        a = input("subject1: ")

        b = input("subject2: ")

        c = y[a]

        d = y[b]

        graph[c].append(d)

    ans = bfs(x)

    for i in ans:

        print(mapp[i],end=" ")

    print("\n")

    for i in  range(x):

        if not vis[i]:

            #print(i)

            result = dfs(i)

            if not result:

                print("They cannot be studied because of interdependancy")

                return

    top\_dfs.reverse()

    for i in top\_dfs:

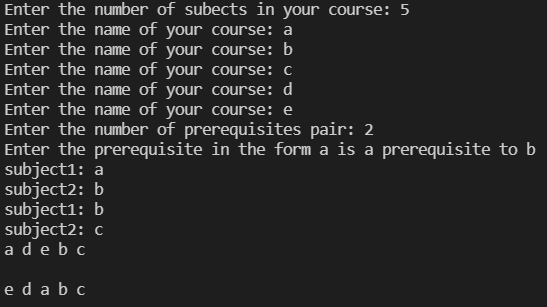
        print(mapp[i],end=" ")

    print("\n")

# if \_name\_ == '\_main\_' :

main()

**Output**:



# Conclusion:

DFS searches until it reaches the leaves of the tree or the last node of the graph that has no further successors, whereas BFS investigates all the nodes of a specific frontier before moving on to the next frontier, as seen in the previous experiment. BFS is complete, whereas DFS isn't. I can also deduce from the previous experiment that BFS is better for nodes that are closer to the source and DFS is better for nodes that are further away from the source.