Lab 4:

BFS:

import collections

class Node:

    def \_\_init\_\_(self,val):

        self.val = val

        self.edges = []

    def \_\_eq\_\_(self,other):

        return self.val == other

    def \_\_hash\_\_(self):

        return self.val

class Graph:

    def \_\_init\_\_(self,nodes=[]):

        self.nodes=nodes

    def add\_node(self,val):

        new\_node=Node(val)

        self.nodes.append(new\_node)

    def add\_edge(self, node1, node2):

        node1.edges.append(node2)

        node2.edges.append(node1)

    goal = int(input('enter goal'))

    def bfs(self):

        if not self.nodes:

            return[]

        start = self.nodes[0]

        visited, queue,result = set([start]), deque([start]), []

        while queue:

            node= queue.popleft()

            result.append(node)

            if node in visited and node == self.goal:

                break

            for nd in node.edges:

                if nd not in visited:

                    queue.append(nd)

                    visited.add(nd)

        return result

    def dfs(self):

        if not self.nodes:

            return[]

        start = self.nodes[0]

        visited, stack, result = set([start]), [start], []

        while stack:

            node=stack.pop()

            result.append(node)

            if node in visited and node == self.goal:

                break

            for nd in node.edge:

                if nd not in visted:

                    stack.append(nd)

                    visited.add(nd)

        return result

graph=Graph()

g1.add\_edge('0','1')

g1.add\_edge('0','7')

g1.add\_edge('1','2')

g1.add\_edge('1','7')

g1.add\_edge('2','3')

g1.add\_edge('2','5')

g1.add\_edge('2','8')

g1.add\_edge('7','8')

g1.add\_edge('7','6')

g1.add\_edge('7','1')

g1.add\_edge('8','6')

g1.add\_edge('6','5')

g1.add\_edge('3','5')

g1.add\_edge('5','4')

dfs\_result = graph.dfs()

bfs\_result = graph.bfs()

print("BFS")

for i in range(len(bfs\_result)):

    print(bfs\_result[i].val)

print("DFS")

for i in range(len(dfs\_result)):

    print(dfs\_result[i].val)