***EXPERIMENT 03***

***Best First Search To Find Goal Nodel***

tree = {  
    'A': [('B', 12), ('C', 4)],  
    'B': [('D', 7), ('E', 3)],  
    'C': [('F', 8), ('G', 2)],  
    'E': [('H', 0)],  
    'F': [('H', 0)],  
    'G': [('H', 0)],  
    'H': []  
}

start, goal = input("Enter the start and goal state: ").split(" ")

def Best\_First\_Search(start, goal, tree, openl=[], closel=[]):  
    if start is goal:        # => START IS GOAL NODE  
        closel.append(start)  
        print("F O U N D: ", closel)  
        return  
    if start not in closel:  # => explore node with lowest heuristic value  
        print(start, " Not in Close List,  e x p l o r e")  
        closel.append(start)  
        neighbors = tree[start]  
        for nbors in neighbors:  # => Find neighbors and apppend to open  
            print(nbors, " is neighbor to ", start)  
            if nbors[0][0] not in openl:  
                openl.append(nbors)  
        # sort to get node with lowest heuristic value  
        openl.sort(key=lambda x: x[1])  
        print("U P D A T E D open\_list: ", openl)  
        if len(openl) == 0:  # open is empty, node not found  
            print("N O D E   N O T   F O U N D")  
            return  
        if openl[0][0] is goal:  # If found  
            closel.append(openl[0][0])  
            print("F O U N D: ", closel)  
        else:  # Otherwsie explore node with best heuristic value  
            node = openl[0]  
            openl.remove(node)  
            Best\_First\_Search(node[0], goal, tree, openl, closel)

Best\_First\_Search(start, goal, tree, [], [])

***Output:***

*Enter the start and goal state: A H*

*A Not in Close List, e x p l o r e*

*('B', 12) is neighbor to A*

*('C', 4) is neighbor to A*

*U P D A T E D open\_list: [('C', 4), ('B', 12)]*

*C Not in Close List, e x p l o r e*

*('F', 8) is neighbor to C*

*('G', 2) is neighbor to C*

*U P D A T E D open\_list: [('G', 2), ('F', 8), ('B', 12)]*

*G Not in Close List, e x p l o r e*

*('H', 0) is neighbor to G*

*U P D A T E D open\_list: [('H', 0), ('F', 8), ('B', 12)]*

*F O U N D: ['A', 'C', 'G', 'H']*