Artificial Intelligence

Nehal Jhajharia (U20CS093)
Lab Assignment 4

Ques. Implement Traveling Salesman problem using below algorithms in prolog. Compare the complexity of both algorithms. Which algorithm is best suited for implementing the Traveling Salesman problem and why?

1) BFS
2) DFS

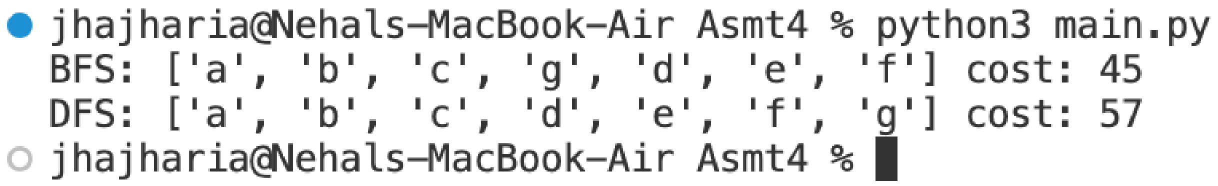
```
class TSP:
   def __init__(self):
       self.graph = []
       self.init_graph()
       self.bfs()
       self.dfs()

   def init_graph(self): #abcdefg
   # let's push weights
       self.graph.append([0, 12, 10, -1, -1, -1, 12]) # a
       self.graph.append([12, 0, 8, 12, -1, -1, -1]) # b
       self.graph.append([10, 8, 0, 11, 3, -1, 9]) #c
       self.graph.append([-1, 12, 11, 0, 11, 10, -1]) # d
       self.graph.append([-1, -1, 3, 11, 0, 6, 7]) #e
       self.graph.append([-1, -1, -1, 10, 6, 0, 9]) #f
       self.graph.append([12, -1, 9, -1, 7, 9, 0]) #g

   def bfs(self):
       res = []
       visited = [0, 0, 0, 0, 0, 0, 0]
       start = 0
       queue = []
       queue.append(start)
       while len(queue) != 0:
           top = queue[0]
           if visited[top] == 0:
               res.append(top)
               visited[top] = 1
               for idx, j in enumerate(self.graph[top]):
                   if j > 0 and visited[idx] == 0: queue.append(idx)
           queue.pop(0)
       cost = 0
       for i in range(len(self.graph) - 1):
           cost += self.graph[res[i]][res[i + 1]]
       res = [chr(i+97) for i in res]
       print("BFS:", res, "cost:", cost)

   def dfs(self):
       res = []
       visited = [0, 0, 0, 0, 0, 0, 0]
       stack = []
       stack.insert(0, 0)
       while len(stack) != 0:
           top = stack[0]
           if visited[top] == 0:
               res.append(top)
               visited[top] = 1
               for idx, j in enumerate(self.graph[top]):
                   if j > 0 and visited[idx] == 0:
                       stack.insert(0, idx)
                       break
           else:
               stack.pop(0)
       cost = 0
       for i in range(len(self.graph) - 1):
           cost += self.graph[res[i]][res[i+1]]
       res = [chr(i+97) for i in res]
       print("DFS:", res, "cost:", cost)


if __name__ == '__main__':
  TSP()
```



```
● jhajharia@Nehals-MacBook-Air Asmt4 % python3 main.py
  BFS: ['a', 'b', 'c', 'g', 'd', 'e', 'f'] cost: 45
  DFS: ['a', 'b', 'c', 'd', 'e', 'f', 'g'] cost: 57
○ jhajharia@Nehals-MacBook-Air Asmt4 % 
```