المحاضرة الخامسة

Coins & Travelling Salesperson Problems

Coins problem

```
from queue import Queue
def flip(coins, index):
    coins[index] = 't' if coins[index] == 'h'
    else 'h'
 def bfs(start, goal):
    visited = set()
    q = Queue()
    q.put(start)
    visited.add(tuple(start))
    while not q.empty():
        state = q.get()
        if state == goal:
            return state
```

```
 for i in range(len(state)):
     temp = state.copy()
     flip(temp, i)
     if tuple(temp) not in visited:
         print(temp)
         q.put(temp)
         visited.add(tuple(temp))
return None
```

```
 # implementation
start = ['h', 'h', 'h']
goal = ['t', 't', 't']
result = bfs(start, goal)
if result is None:
    print("No solution found.")
else:
    print("Solution:", result)
```

Travelling Salesperson Problem(TSP)

Q/**Complete the following code for the TSP :**

```
# Define the graph for the cities and their connections
graph = {
    'A': [('B', 10), ('C', 20), ('D', 15), ('E', 50)],
    'B': [('A', 10), ('C', 25), ('D', 20), ('E', 30)],
    'C': [('A', 20), ('B', 25), ('D', 35), ('E', 30)],
    'D': [('A', 15), ('B', 20), ('C', 35), ('E', 40)],
    'E': [('A', 50), ('B', 30), ('C', 30), ('D', 40)]
}
# Define the function to solve the TSP using the BFS algorithm
def solve_tsp_bfs(start_city):
    queue = [(start_city, [start_city], 0)]
    shortest_path = None
```

```
while queue:
    current_city, visited_cities, total_distance = queue.pop(0)
    if set(visited_cities) == set(graph.keys()):
# If all cities have been visited, update the shortest path
        if shortest_path is None or total_distance < shortest_path[1]:
            shortest_path = (visited_cities, total_distance)
    else:
_____
_____
_____
```