CS210 : ARTIFICIAL INTELLIGENCE LAB

Practical Test

Submitted By:

Name: AKSHAT SAHU

Roll No: U22CS034

Branch: CSE

Semester: 4th Sem

Division : A

Submitted To: Dr. Chandra Prakash

Department of Computer Science and Engineering

SV NATIONAL INSTITUTE OF TECHNOLOGY

SURAT

2024

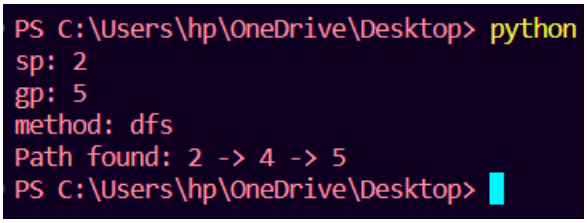
(1) Design a Python program to use the A* algorithm for robot navigation in a maze. Given a maze represented as a grid with walls, the start position, and the goal position, your task is to find the shortest path from the start position to the goal position while navigating through the maze.
maze = [[0, 1, 0, 0, 0],
[0, 1, 0, 1, 0],
[0, 0, 0, 1, 0],
[0, 1, 0, 1, 0],
[0, 0, 0, 0, 0]
]
start = (0, 0)
goal = (4, 4)
(2) Choose a classic problem-solving task (e.g., the Eight Queens Puzzle, the Traveling Salesman Problem) and formulate it as a search problem. Implement and compare different search algorithms (e.g., DFS, BFS, A*, etc.) to find solutions to the problem. Analyze the efficiency and effectiveness of each algorithm in solving the problem.
(3) You are given a set of items, each with a weight and a value, and a knapsack with a maximum weight capacity. Your task is to maximize the total value of items in the knapsack without exceeding its weight capacity. Design a Python program to solve this optimization problem using a Genetic Algorithm (GA).
eg. Example items (weight, value) items = [Item(2, 10), Item(3, 20), Item(4, 30), Item(5, 40)]
#knapsack_capacity = 7
population_size = 50
num_generations = 100
mutation_rate = 0.1

```
#  Choose a classic problem-solving task and formulate it as a search problem.
Implement and compare different search
# algorithms (e.g., DFS, BFS, A*, etc.) to find solutions to the problem. Analyze
the efficiency and
# effectiveness of each algorithm in solving the problem.

ROOMS = {0: [3],1: [4, 5],2: [4],3: [1, 2, 3],4: [0,3, 5],5: [1, 3]}
def bfs(start, goal):
    queue = [(start, [])]
    visited = set()
    parents = {start: None}
    while queue:
        current_node, path = queue.pop(0)
        if current_node == goal:
            final_path = []
            while current_node is not None:
```

```
                final_path.insert(0, current_node)
                current_node = parents[current_node]
            return final_path
        if current_node not in visited:
            visited.add(current_node)
            for neighbor in ROOMS[current_node]:
                if neighbor not in visited and neighbor not in parents:
                    parents[neighbor] = current_node
                    queue.append((neighbor, path + [current_node]))
    return None
def dfs(start, goal):
    stack = [(start, [])]
    visited = set()
    parents = {start: None}
    while stack:
        current_node, path = stack.pop()
        if current_node == goal:
            final_path = []
            while current_node is not None:
                final_path.insert(0, current_node)
                current_node = parents[current_node]
            return final_path
        if current_node not in visited:
            visited.add(current_node)
            for neighbor in ROOMS[current_node]:
                if neighbor not in visited and neighbor not in parents:
                    parents[neighbor] = current_node
                    stack.append((neighbor, path + [current_node]))
    return None
startpoint = int(input("sp: "))
goalpoint = int(input("gp: "))
searchapproach = input("method: ").upper()
if startpoint not in ROOMS or goalpoint not in ROOMS or searchapproach not in
("BFS", "DFS"):
    print("Invalid input. Please check and try again.")
    exit()
path = bfs(startpoint, goalpoint) if searchapproach == "BFS" else dfs(startpoint,
goalpoint)
if path:
    print(f"Path found: {' -> '.join(map(str, path))}")
else:
    print("No path found from", startpoint, "to", goalpoint)
```

OUTPUT-


PS C:\Users\hp\OneDrive\Desktop> python
sp: 2
gp: 5
method: dfs
Path found: 2 -> 4 -> 5
PS C:\Users\hp\OneDrive\Desktop>