CS210 : ARTIFICIAL INTELLIGENCE LAB

LAB ASSIGNMENT 2: AI & Python

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

PART - 1

1)

Autonomous Robot -

1) Sensors - Sensors are the objects in robot that gather the information around its surrounding so that next action can be taken based on that information

- Sensors that we use in our robot

- Sonar Sensors - It is used to detect objects and distance between them robot and that object.
- Laser Sensor - It is used measure distance between the objects and robot.
- Visual Sensor - It is used to identify surrounding Environment

2) Actuators - Actuators are the components that perform the action on the basis of the information provided by the sensors.

1) Motors - It is used for the mechanical motion of the robot.

2) Brake - It is used to control the motion.

3) Steering - It helps in controlling the direction.

4) Hydraulic Actuators - It is used to deliver High power.

2:

```
import sys

def is_room_clean(dirt_status):
    return all(d == 0 for d in dirt_status)

def vacuum(room_number, dirt_status, algorithm):
    valid_actions = ['L', 'R', 'U', 'D', 'S', 'N']

    if algorithm == 'dfs':
        stack = [(room_number, [])]
    elif algorithm == 'bfs':
        queue = [(room_number, [])]
    else:
        print("Invalid algorithm specified.")
        return

    while stack or queue:
        if algorithm == 'dfs':
            current_room, actions = stack.pop()
        elif algorithm == 'bfs':
            current_room, actions = queue.pop(0)

        if is_room_clean(dirt_status):
            print_sequence(actions)
            return

        for action in valid_actions:
            new_room, new_dirt_status = simulate_action(current_room,
dirt_status, action)

            if algorithm == 'dfs':
                stack.append((new_room, actions + [(current_room, action)]))
            elif algorithm == 'bfs':
                queue.append((new_room, actions + [(current_room, action)]))

    print("Goal state not reached.")

def simulate_action(current_room, dirt_status, action):
    new_room = current_room
    new_dirt_status = dirt_status.copy()

    return new_room, new_dirt_status
```

```
def print_sequence(actions):
    for room, action in actions:
        print(f"{room},{action}")

def main():
    if len(sys.argv) != 3:
        print("Invalid number of arguments.")
        return

    input_file = sys.argv[1]
    output_file = sys.argv[2]

    with open(input_file, "r") as file:
        room_number = int(file.readline().strip())
        dirt_status = list(map(int, file.readline().strip().split(',')))
        algorithm = file.readline().strip()
        vacuum(room_number, dirt_status, algorithm)

if __name__ == "__main__":
    main()
```

3:

```
from collections import defaultdict

class Graph:
    def __init__(self):
        self.adjacency_list = defaultdict(list)

    def add_edge(self, u, v):
        self.adjacency_list[u].append(v)
        self.adjacency_list[v].append(u)

    def bfs(self, start, goal):
        visited = [False] * len(self.adjacency_list)
        queue = []
        path = []

        queue.append(start)
        visited[start] = True

        while queue:
            current = queue.pop(0)
            path.append(current)
```

```
            if current == goal:
                return path

            for neighbor in self.adjacency_list[current]:
                if not visited[neighbor]:
                    queue.append(neighbor)
                    visited[neighbor] = True

    def dfs(self, current, goal, visited, path):
        path.append(current)
        visited[current] = True

        if current == goal:
            return path

        for neighbor in self.adjacency_list[current]:
            if not visited[neighbor]:
                return self.dfs(neighbor, goal, visited, path)

def main():
    g = Graph()
    g.add_edge(1, 2)
    g.add_edge(2, 3)
    g.add_edge(3, 4)
    g.add_edge(3, 5)
    g.add_edge(5, 4)

    start_state = int(input("Enter starting state (1-5): "))
    goal_state = int(input("Enter state (1-5): "))
    approach = input("Enter approach (BFS/DFS): ")

    if start_state < 1 or start_state > 5 or goal_state < 1 or goal_state > 5:
        print("Invalid input.")
        return

    if approach.lower() == 'bfs':
        bfs_path = g.bfs(start_state, goal_state)
        if bfs_path:
            print("BFS Path:", "-".join(map(str, bfs_path)))
        else:
            print("Path not found.")
    elif approach.lower() == 'dfs':
        visited = [False] * len(g.adjacency_list)
        dfs_path = g.dfs(start_state, goal_state, visited, [])
```

```
        if dfs_path:
            print("DFS Path:", "-".join(map(str, dfs_path)))
        else:
            print("Path not found.")
    else:
        print("Invalid approach.")

if __name__ == "__main__":
    main()
```

PART B

1:

Part B

1 * Tonyii Hiwonder Robot.

1) Perceiving devices and Actuators
⇒ perceiving device
⇒ microphone
⇒ camera
⇒ Ultrasonic sensor
⇒ Touch sensor
⇒ motion sensor

2) Actuator
⇒ Electric motor
⇒ Servo motor
⇒ Gripper
⇒ Brake
⇒ Hydraulic Actuator.

→ Steps to build a Humanoid Robot ⇒

1 Planning ⇒
→ Making a note of functionality of Robot
→ Making a design based on the functionality you need

2) Assembly

3) Sensors

4) Programming

5) Power supply

6) Optimizing.

2)* TurtleBot 3

Perceiving devices

1) Cliff sensor
2) inertial measurement
3) Camera
4) lidar sensors

Actuator

1) Wheels
2) Castors
3) Dynamixel motors

→ Steps to build an Autonomous Robot.

1) Planning
2) Hardware Assembly
3) Power Supply
4) Install turtlebot 3 Packages
5) Sensors Calibration
6) Mapping setup
7) Launch mapping nodes
8) Map visualization
9) Save map
10) Optimization.

* Sensors in mobile devices. Which of them could be used in human motion analysis.

⇒ Accelerometer
⇒ Gyroscope
⇒ Magnetometer

⇒ GPS
⇒ Proximity
⇒ Light Sensor
⇒ Fingerprint
⇒ Barometer
⇒ camera.

Accelerometer, Gyroscope, GPS and camera can be used to analyse human motion.