|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **AY -ODD 2024-25** | | | | | | | | | | | | | |
| **GUJARAT TECHNOLOGICAL UNIVERSITY** | | | | | | | | | | | | | |
| **SCHOOL OF ENGINEERING AND TECHNOLOGY** | | | | | | | | | | | | | |
| **PRACTICAL - 6** | | | | | | | | | | | | | |
| **Course Code & Name** | | | **ME01095021- Artificial Intelligence** | | | | | | | | | | |
| **Academic Term:** | | | **AY –ODD 2024-25** | | | | | **Semester** | | | | **I** | |
| **Student Enrollment No:** | | | **241370795004** | | | | | **Batch:** | | | |  | |
| **Student Name:** | | | **Dake Darsh Dhaneshkumar** | | | | | | | | | | |
|  | |  |  |  | |  | |  | | | |  |  |
| **AIM/Objective:** | | | | | | | | | | | | | |
| 1 | | To implement A\* algorithm for Block World Problem with local & global heuristic functions. | | | | | | | | | | | |
|  | |  |  |  | |  | |  | |  | | |  |
| **Expected Outcome:** | | | | | | | | | | **CO/PO/PSO** | | | |
| 1 | | The expected outcome is that the Block World Agent will solve the problem using both local and global heuristics. The local heuristic focuses on the immediate difference between the current state and the goal, while the global heuristic incorporates both the difference and the number of moves made so far, aiming for a more efficient solution. | | | | | | | | CO5 | | | |
|  | | **Experiment Result and Analysis**  **Resources and Software used:**   1. Python 3.12.6 2. Jupyter Notebook   **Code:**  import copy  import time  class State:  def \_\_init\_\_(self, first\_stack, second\_stack, total\_num, moves=None):  if moves is None:  moves = []  self.first\_stack = first\_stack  self.second\_stack = second\_stack  self.total\_num = total\_num  self.moves = moves  def \_\_eq\_\_(self, other):  return (  self.first\_stack == other.first\_stack  and self.second\_stack == other.second\_stack  and self.total\_num == other.total\_num  and self.moves == other.moves  )  def goal\_state\_move(self, visited\_states):  while self.difference() != 0:  visited\_states.append(copy.deepcopy(self.first\_stack))  self = self.select\_move()  visited\_states.append(copy.deepcopy(self.first\_stack)) # Add final state  return self.moves  def select\_move(self):  # Generate valid moves to minimize the difference  for index, stack in enumerate(self.first\_stack):  for index2, stack2 in enumerate(self.first\_stack):  if index != index2: # Avoid moving to itself  curr\_table, move = self.valid\_state\_move(self.first\_stack, index, index2)  new\_state = State(curr\_table, self.second\_stack, self.total\_num, copy.copy(self.moves))  new\_state.moves.append(move)  if new\_state.difference() < self.difference():  return new\_state  # Move top block to the temporary table if no improvement  for index, stack in enumerate(self.first\_stack):  if len(stack) > 1: # If not alone  curr\_table, move = self.valid\_state\_move(self.first\_stack, index, -1)  new\_state = State(curr\_table, self.second\_stack, self.total\_num, copy.copy(self.moves))  new\_state.moves.append(move)  if new\_state.difference() <= self.difference():  return new\_state  def valid\_state\_move(self, table, start\_index, end\_index):  temp\_table = copy.deepcopy(table)  left = temp\_table[start\_index]  top\_block = left.pop()  if end\_index < 0: # Move to temporary table  temp\_table.append([top\_block])  move = (top\_block, "Table")  else: # Move to another stack  right = temp\_table[end\_index]  move = (top\_block, right[-1] if right else "Empty Stack")  right.append(top\_block)  if len(left) == 0:  temp\_table.remove(left)  return temp\_table, move  def difference(self):  same\_num = 0  for left in self.first\_stack:  for right in self.second\_stack:  index = 0  while index < len(left) and index < len(right):  if left[index] == right[index]:  same\_num += 1  index += 1  else:  break  diff = self.total\_num - same\_num  return diff  class BlockWorldAgent:  def \_\_init\_\_(self):  self.visited\_states = []  def solve(self, initial\_arrangement, goal\_arrangement):  start\_time = time.time()  total\_num = sum(len(stack) for stack in initial\_arrangement)  state = State(initial\_arrangement, goal\_arrangement, total\_num)  solution = state.goal\_state\_move(self.visited\_states)  end\_time = time.time()  runtime = str((end\_time - start\_time) \* 1000)  print("\nSolution Moves:", solution)  print("Running Time:", runtime, "ms")  print("\nVisited States:")  for step, state in enumerate(self.visited\_states):  print(f"Step {step + 1}: {state}")  return solution  def test():  test\_agent = BlockWorldAgent()  initial\_arrangement\_1 = [["A", "B", "C"], ["D", "E"]]  goal\_arrangement\_1 = [["A", "C"], ["D", "E", "B"]]  print("Initial Arrangement:", initial\_arrangement\_1)  print("Goal Arrangement:", goal\_arrangement\_1)  test\_agent.solve(initial\_arrangement\_1, goal\_arrangement\_1)  if \_\_name\_\_ == "\_\_main\_\_":  test()  **Output:** | | | | | | | |  | | |  |
|  | | |  |  | | --- | --- | | **Conclusion** | | | 1 | The global heuristic is likely to be more effective than the local heuristic, as it considers the path cost, potentially resulting in fewer moves and a faster solution. The local heuristic, on the other hand, may take more steps since it only looks at the current state difference. | | | | | | | | | | | | |
|  | |  |  |  |  | |  | |  | |  | | |
| **Evaluation Rubrics** | | | | | **Marks** | | **Inadequate** | | **Good** | | **Excellent** | | |
| **0%** | | **50%** | | **100%** | | |
| 1 | The understanding of the Student regarding the objective of the given practical | | | | **2** | |  | |  | |  | | |
| 2 | Installation of Software or Hardware Setup level | | | | **2** | |  | |  | |  | | |
| 3 | Quality of the Analysis done | | | | **2** | |  | |  | |  | | |
| 4 | Quality of the report including concluding remarks and Findings | | | | **2** | |  | |  | |  | | |
| 5 | Question & Answer related to given practical & timely submission | | | | **2** | |  | |  | |  | | |
|  | | | | | **10** | |  | |  | |  | | |
| **Total Marks Obtained Out of 10** | | | | | | |  | | | | | | |
|  | |  |  |  | |  |  | | | | |  |  |
|  | |  |  |  | |  |  | | | | |  |  |
|  | |  |  |  | |  |  | | | | |  |  |
|  | | **Date of Completion:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | |  | | **Course**  **Coordinator Sign:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | | | | |