|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **AY -ODD 2024-25** | | | | | | | | | | | | | |
| **GUJARAT TECHNOLOGICAL UNIVERSITY** | | | | | | | | | | | | | |
| **SCHOOL OF ENGINEERING AND TECHNOLOGY** | | | | | | | | | | | | | |
| **PRACTICAL - 3** | | | | | | | | | | | | | |
| **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 Best First Search for 8-puzzle Problem with local & global heuristic functions. | | | | | | | | | | | |
|  | |  |  |  | |  | |  | |  | | |  |
| **Expected Outcome:** | | | | | | | | | | **CO/PO/PSO** | | | |
| 1 | | The implementation of Best-First Search for the 8-puzzle problem using local and global heuristic functions is expected to demonstrate how the choice of heuristic affects the efficiency of finding the solution, with global heuristics typically leading to faster convergence by considering the overall goal. | | | | | | | | CO5 | | | |
|  | | **Experiment Result and Analysis**  **Resources and Software used:**   1. Python 3.12.6 2. Jupyter Notebook   **Code:**  import heapq  class Puzzle8:  def \_\_init\_\_(self, initial\_state):  self.goal\_state = [1,2,3,8,0,4,7,6,5]  self.initial\_state = initial\_state  def local\_heuristic(self, state):  # local heuristic: Misplaced tiles  count = 0  for i in range(9):  if state[i] != 0 and state[i] != self.goal\_state[i]:  count += 1  return count  def global\_heuristic(self, state):  # Global heuristic: Sum of Manhattan distances  distance = 0  for i in range(9):  if state[i] != 0:  goal\_idx = self.goal\_state.index(state[i])  x1, y1 = i // 3, i % 3  x2, y2 = goal\_idx // 3, goal\_idx % 3  distance += abs(x1 - x2) + abs(y1 - y2)  return distance  def is\_goal\_state(self, state):  return state == self.goal\_state  def generate\_children(self, state):  children = []  zero\_idx = state.index(0)  moves = [(0, 1), (0, -1), (1, 0), (-1, 0)] # Right, Left, Down, Up  for move in moves:  new\_x = zero\_idx // 3 + move[0]  new\_y = zero\_idx % 3 + move[1]  if 0 <= new\_x < 3 and 0 <= new\_y < 3:  new\_state = state[:]  new\_zero\_idx = new\_x \* 3 + new\_y  new\_state[zero\_idx], new\_state[new\_zero\_idx] = (  new\_state[new\_zero\_idx],  new\_state[zero\_idx],  )  children.append(new\_state)  return children  def best\_first\_search(self, heuristic):  open\_list = [(heuristic(self.initial\_state), self.initial\_state, [])]  closed\_set = set()  while open\_list:  \_, current\_state, path = heapq.heappop(open\_list)  if self.is\_goal\_state(current\_state):  return path + [current\_state]  if tuple(current\_state) in closed\_set:  continue  closed\_set.add(tuple(current\_state))  for child\_state in self.generate\_children(current\_state):  if tuple(child\_state) not in closed\_set:  heapq.heappush(  open\_list,  (heuristic(child\_state), child\_state, path + [current\_state]),  )  return None  # Example usage  if \_\_name\_\_ == "\_\_main\_\_":  initial\_state = [0,8,3,2,6,4,1,7,5]  puzzle = Puzzle8(initial\_state)  # Using global heuristic (h1)  moves\_h1 = puzzle.best\_first\_search(puzzle.global\_heuristic)  print("Solution using global heuristic (h1):")  for step in moves\_h1:  print(step)  print()    # Using local heuristic (h2)  moves\_h2 = puzzle.best\_first\_search(puzzle.local\_heuristic)  print("Solution using local heuristic (h2):")  for step in moves\_h2:  print(step)  **Output:** | | | | | | | |  | | |  |
|  | | |  |  | | --- | --- | | **Conclusion** | | | 1 | The Best First Search implementation for the 8-puzzle problem shows how different heuristics guide the search. The global heuristic (misplaced tiles) offers a basic estimate, while the local heuristic (Manhattan distance) provides a more accurate path to the goal, often leading to faster solutions with fewer steps. | | | | | | | | | | | | |
|  | |  |  |  |  | |  | |  | |  | | |
| **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:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | | | | |