|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **AY -ODD 2024-25** | | | | | | | | | | | | | |
| **GUJARAT TECHNOLOGICAL UNIVERSITY** | | | | | | | | | | | | | |
| **SCHOOL OF ENGINEERING AND TECHNOLOGY** | | | | | | | | | | | | | |
| **PRACTICAL - 1** | | | | | | | | | | | | | |
| **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 Breadth First Search and measure the storage & time performance by varying the number of levels & nodes. | | | | | | | | | | | |
|  | |  |  |  | |  | |  | |  | | |  |
| **Expected Outcome:** | | | | | | | | | | **CO/PO/PSO** | | | |
| 1 | | We expect to observe that **time complexity** increases linearly with the number of nodes (O(V + E), where V = vertices, E = edges), and **space complexity** depends on the maximum number of nodes at any level (O(W), where W is the width of the graph). As levels and nodes grow, memory usage will increase due to more nodes in the queue, and traversal time will rise proportionally to the graph's size. | | | | | | | | CO5 | | | |
|  | | **Experiment Result and Analysis**  **Resources and Software used:**   1. Python 3.12.6 2. Jupyter Notebook   **Code:**  from collections import deque  import time  # BFS from given source s with performance measurement  def bfs\_performance(adj, s):  q = deque()  visited = [False] \* len(adj)  visited[s] = True  q.append(s)    max\_queue\_size = 0 # Track maximum storage used by the queue  start\_time = time.time() # Start time  while q:  max\_queue\_size = max(max\_queue\_size, len(q))  curr = q.popleft()  for x in adj[curr]:  if not visited[x]:  visited[x] = True  q.append(x)  end\_time = time.time() # End time  execution\_time = end\_time - start\_time  return max\_queue\_size, execution\_time  # Function to add an edge to the graph  def add\_edge(adj, u, v):  adj[u].append(v)  adj[v].append(u)  # Generate a tree-like graph with given levels and branching factor  def generate\_tree\_graph(levels, branching\_factor):  adj = []  node = 0  for level in range(levels):  level\_nodes = branching\_factor \*\* level  for i in range(level\_nodes):  while len(adj) <= node:  adj.append([]) # Ensure adjacency list is large enough  parent = node  for \_ in range(branching\_factor):  child = len(adj)  adj.append([]) # Expand adjacency list for the child  add\_edge(adj, parent, child)  node += 1  return adj  # Measure performance by varying levels and nodes  if \_\_name\_\_ == "\_\_main\_\_":  levels\_to\_test = [8, 10] # Number of levels in the tree  branching\_factors = [2, 3, 4] # Branching factor for each node  for levels in levels\_to\_test:  for branching\_factor in branching\_factors:  print(f"Testing with Levels: {levels}, Branching Factor: {branching\_factor}")  adj = generate\_tree\_graph(levels, branching\_factor)  max\_queue\_size, execution\_time = bfs\_performance(adj, 0)  print(f"Max Queue Size (Storage): {max\_queue\_size}")  print(f"Execution Time: {execution\_time:.6f} seconds")  print("-" \* 50)  **Inputs:**    **Output:** | | | | | | | |  | | |  |
|  | | |  |  | | --- | --- | | **Conclusion** | | | 1 | The experiment demonstrates that BFS's execution time grows with the graph size, reflecting its O(V+E)O(V + E) complexity, while storage requirements depend on the graph's width. Higher levels and branching factors increase both time and memory usage, showing BFS is efficient for smaller or shallow graphs but resource-intensive for wide ones. | | | | | | | | | | | | |
|  | |  |  |  |  | |  | |  | |  | | |
| **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:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | | | | |