****

**Green University of Bangladesh**

**Department of Computer Science and Engineering(CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Spring, Year:2024), B.Sc. in CSE (Day)**

**LAB REPORT NO #03**

**Course Title: Artificial Intelligence Lab**

**Course Code: CSE 316 Section: 213D3**

**Lab Experiment Name: Iterative Deepening depth-first search(IDDFS).**

**Student Details**

|  |  |  |
| --- | --- | --- |
| **Name** | | **ID** |
| **1.** | Shoaib Ahmed | 212902019 |

**Lab Date : 16-03-24**

**Submission Date : 11-05-24**

**Course Teacher’s Name : Md. Zahidul Hasan**

**[For Teachers use only: Don’t Write Anything inside this box]**

|  |
| --- |
| **Lab Report Status**  **Marks: ………………………………… Signature:.....................**  **Comments:.............................................. Date:..............................** |

**1. TITLE OF THE LAB EXPERIMENT**

Iterative Deepening depth-first search(IDDFS).

**2. OBJECTIVES/AIM**

* To take as input the number of vertices in the graph, the adjacency matrix

representing the graph, the starting vertex, and the maximum depth for the IDDFS.

* To perform IDDFS from the starting vertex to the specified maximum depth. The

IDDFS is implemented as a series of Depth depth-limited search (DLS) calls with

increasing depth limits.

* To maintain a list of visited vertices during the IDDFS. Suppose a vertex is

encountered that has already been visited and is earlier in the visited list than the

current vertex. In that case, assumes a topological sort is possible and adjusts the

order of vertices in the visited list.

* To print the visited vertices after each DLS call, showing the order in which

vertices are visited at each depth limit.

**3. PROCEDURE / ANALYSIS / DESIGN**

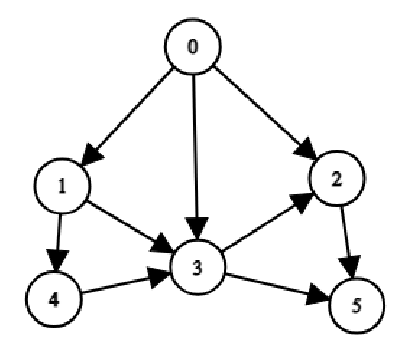


Fig-3.1: Demo Graph

Iterative Deepening Depth-First Search (IDDFS) is a strategy used to explore nodes in a graph or tree up to a specified depth, incrementally increasing the depth limit with each iteration until a solution is found. This approach combines the completeness of breadth-first search (BFS) with the memory efficiency of depth-first search (DFS). IDDFS ensures that all nodes within the depth limit are explored, making it suitable for scenarios where the exact depth of the solution is unknown or where memory constraints prevent full breadth-first exploration. However, IDDFS may revisit nodes and incur redundant work at deeper levels compared to more sophisticated algorithms like A\* search or iterative deepening A\* (IDA\*). Overall, IDDFS strikes a balance between completeness and efficiency, particularly in scenarios where limited memory or the need for a simple yet effective search strategy is paramount.

**4. IMPLEMENTATION**

Code:

|  |
| --- |
| def dls(graph, start, depth, maxDepth, m, visited):  if depth > maxDepth or start in visited:  return  visited.append(start)  for i in range(m):  if graph[start][i] == 1:  if i not in visited:  dls(graph, i, depth + 1, maxDepth, m, visited)  else:  if visited.index(i) < visited.index(start):  if depth + 1 <= maxDepth:  visited.remove(i)  visited.remove(start)  visited.insert(visited.index(i), start)  visited.insert(visited.index(start), i)  dls(graph, i, depth + 1, maxDepth, m, visited)  def iddfs(graph, start, depth, m):  visited = []  for i in range(depth):  dls(graph, start, 0, i, m, visited)  if i == depth - 1:  break  visited.clear()  print("Topological sort: ", end="")  print(' -> '.join(map(str, visited)))  def main():  m = int(input("Enter the number of vertices: "))  graph = []  print("Enter the adjacency matrix: ")  for \_ in range(m):  graph.append(list(map(int, input().split())))  start = int(input("Enter the start vertex: "))  depth = int(input("Enter the maximum depth: "))  iddfs(graph, start, depth + 1, m)  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**5. TEST RESULT / OUTPUT**

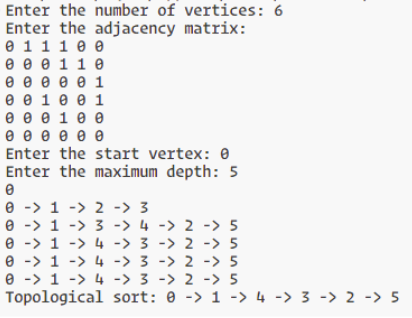


Fig-3.1: Output of IDDFS

**6. ANALYSIS AND DISCUSSION**

Here I implement a Depth-Limited Search (DLS) and Iterative Deepening Depth-First Search (IDDFS) algorithm to explore a graph represented by an adjacency matrix. The dls function recursively traverses nodes up to a specified depth limit, handling cycles and performing a topological sort if necessary. The iddfs function iteratively applies dls with increasing depth limits until a solution is found or the maximum depth is reached. The main function facilitates user interaction by taking input for graph details and initiating the IDDFS search. However, the implementation lacks clarity in handling the topological sort aspect, as the sorting operations within dls may not correctly maintain the order. Overall, while the code demonstrates basic graph traversal and iterative deepening techniques, it requires refinement for robustness and clarity in handling topological sorting and ensuring correctness in edge cases.