**Department of Computer Science and Engineering**

|  |  |
| --- | --- |
| **Course Code: CSE221** | **Credits: 1.5** |
| **Course Name: Algorithms** | **Semester: Spring’19** |

**Lab 04  
Depth First Search (DFS)**

1. **Topic Overview:**

DFS is a traversal or search algorithm used for tree or graph data structures. DFS starts at the root node and explores as far as possible before backtracking when it reaches a leaf node. This algorithm has many applications, such as topologically sorting a graph or solving a maze. In this lab, students will first learn how to traverse a graph using DFS, and then they will apply the DFS algorithm to solve various problems.

1. **Lesson Fit:**

There is prerequisite to this lab. Students must have completed Lab 01 and should have a basic idea on the following concepts:

* 1. Graph Representation
  2. Graph Traversal
  3. Recursion

1. **Learning Outcome:**

After this lecture, the students will be able to:

* 1. Implement the DFS algorithm
  2. Apply DFS to solve a Maze
  3. Topologically sort using DFS

1. **Anticipated Challenges and Possible Solutions**
   1. **Task 3a:** Students might face problem to visualize or represent the maze as a graph

**Solution:** The instructor will explain with a small example before starting the task.

1. **Acceptance and Evaluation**

Students will show their progress as they complete each problem. They will be marked according to their class performance. There may be students who might not be able to finish all the tasks, they will submit them later and give a viva to get their performance mark. The mark distribution for the lab will be as follows:

Code: 05

Viva: 05

1. **Activity Detail**
   1. **Hour: 1  
      Discussion:**Explain how DFS works and its various applications. **Problem Task:**
      1. Task 1
   2. **Hour: 2**

**Discussion:**

Check DFS is implemented correctly. Explain topological sorting and the maze problem

**Problem Task:**

* + 1. Task 2-3
  1. **Hour: 3**

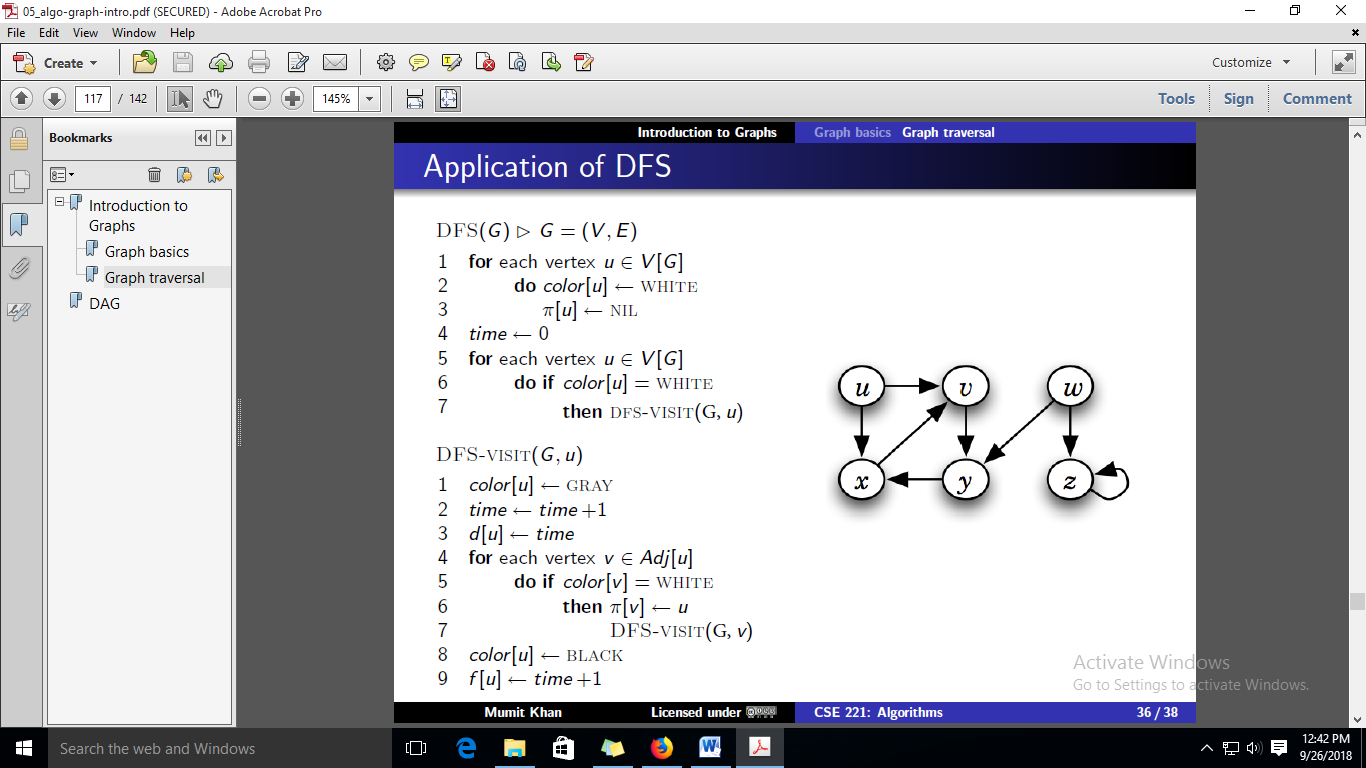
**Discussion:**

Check that individual students have completed at least tasks 1 and 2. Discuss any issues that the students have faced. Discuss Task 3.

1. **Home tasks:** All unfinished tasks.

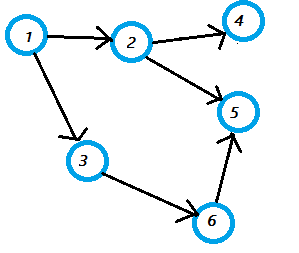
**Lab 4 Activity List**

**Task 1**



Read the graph.txt file to create an adjacency matrix **or** list. Reuse your code from Lab01. **Graph.txt represents a directed acyclic graph.** After creating the matrix/list traverse the graph by implementing the DFS algorithm given above. You should print the order of both visited/discovered (grey) nodes and processed/finished (black) nodes.

**Sample graph (your graph.txt may represent a different graph):**

****

**Sample Output:**

|  |  |
| --- | --- |
| Adjacency matric:  0 1 2 3 4 5 6  0 0 0 0 0 0 0 0  1 0 0 1 1 0 0 0  2 0 0 0 0 1 1 0  3 0 0 0 0 0 0 1  4 0 0 0 0 0 0 0  5 0 0 0 0 0 0 0  6 0 0 0 0 0 1 0 | Discovered Nodes:  1, 2, 4, 5, 3, 6  Processed Nodes:  4, 5, 2, 6, 3, 1 |

**Task 2**

Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge uv, vertex u comes before v in the ordering. A graph can be topologically sorted using DFS by sorting the vertices in descending order by their finishing time. Write an algorithm (make small modification to your algorithm in task 1) to topologically sort the graph in graph.txt.

**Sample Output:**

|  |
| --- |
| After Topological Sort: Nodes: 1 🡪 3 🡪 6 🡪 2 🡪 5 🡪 4  Finish Times: 12 11 10 7 6 4 |

**Task 3**

1. Create maze.txt file to represent the input for the maze given below. Each tile in the maze represents a node and is labelled with a number. If you can go from one tile to another in one move (left, right, up or down) then there is an edge between the nodes/tiles.
2. Apply the DFS algorithm to solve the maze below. You have to start at node 1 and end at node 9. Once you reach your destination you should terminate the algorithm, thus slight modification to the algorithm in task 1 is required. Print the entire path traveled to reach the end. For example, one possible solution is

1🡪2🡪5🡪6🡪10🡪11🡪10🡪6🡪7🡪8🡪9

