

# Assignment 11: Depth first search

**Allocated time: 2 hours**

## Implementation

You are required to correctly and efficiently implement the depth first search algorithm (DFS) (Chapter 22.3 from Cormen). For graph representation, you should use adjacency lists. You also have to:

- Implement the Tarjan algorithm for detecting strongly connected components
- Implement topological sorting (Chapter 22.4 from Cormen)

## Thresholds

Threshold	Requirements
5	Correct and efficient implementation for DFS + demo
6	Correct and efficient implementation for topological sort + demo
8	Correct and efficient implementation for Tarjan + demo
10	Analysis of the DFS performance + graphics

## Evaluation

**!** Before you start to work on the algorithm evaluation code, make sure you have a correct algorithm! Exemplify the correctness of your algorithm/implementation by running it on a smaller graph:

- Printing the initial graph (the adjacency lists)
- Printing the tree resulted from dfs
- Printing all strongly connected components of the graph
- A list of nodes sorted topologically (should this list be nonempty/if it is why so?)

Since, for a graph, both  $|V|$  and  $|E|$  may vary, and the running time of DFS depends on both, we will make each analysis in turn:

1. Set  $|V| = 100$  and vary  $|E|$  between 1000 and 4500, using a 100 increment. Generate the input graphs randomly – make sure you don't generate the same edge twice for the same graph. Run the DFS algorithm for each graph and count the number of operations performed; generate the corresponding chart (i.e. the variation of the number of operations with  $|E|$ ).
2. Set  $|E| = 4500$  and vary  $|V|$  between 100 and 200, using an increment equal to 10. Repeat the procedure above to generate the chart which gives the variation of the number of operations with  $|V|$ .
3. Interpret your charts.