# Lab 5 Performing depth-first search on graphs

C S 372 Data Structures and Algorithms

October 1, 2019

In this lab, you will design and implement depth-first search (DFS) algorithms using recursive and iterative strategies.

## 1 Requirement for the DFS algorithm

We will design linear-time DFS algorithms with the following input and output:

**Input:** graph $G$ with the Graph class defined in Lab 4.

**Output:** All nodes in graph $G$

- are marked as visited
- have a pre clock value
- have a post clock value

Please update the Graph and Node classes to include the above information. There are multiple ways of enhancing the data structures from the previous lab. You are free to design your strategy. They cannot be saved as global variables.

## 2 Recursive solution

Implement the recursive solution we discussed in class using the following C++ function prototype:

```
void DFS_recursive(Graph & G)
{
  // Your recursive code for DFS
}
```

## 3 Iterative solution

Implement the iterative solution using the strategy similar to the iterative solution of the breadth-first-search (BFS). Instead of using a queue, you can use a stack to save the discovered but not yet processed nodes. Use the following C++ function prototype:

```
void DFS_iterative(Graph & G)
{
  // Your iterative code for DFS
}
```

You are encouraged to use the `std::stack` class from C++.

## 4 Test the classes

Generate five example graphs to test your code. The graphs do not have to be very large but must represent the following variety: cyclic/acyclic, directed/undirected, having one or more connected components.

Your C++ program must include a main() function that calls the `testall()` function. When the program is compiled by a C++ compiler, it generates a binary executable file that will run when invoked from the command line.

## 5 Plot the runtime as a function of number of nodes and edges

After testing the correctness of the the program, you will study how the runtime scale with the size of graph for the two methods. Use the random graph function you developed in Lab 4 to generate random graphs of increasing sizes reaching 10,000 nodes and 1,000,000 edges or more.

You will produce four plots in R:

1. DFS-recursive runtime as a function of number of nodes
2. DFS-iterative runtime as a function of number of nodes
3. DFS-recursive runtime as a function of number of edges
4. DFS-iterative runtime as a function of number of edges

## 6 Submission

Write a lab report to describe your lab work done in the following sections:

1. Introduction (define the background, motivation, and the problem),
2. Methods (provide the solutions),
3. Results
   (a) visualize your five test graphs using R package `igraph`
   (b) show four figures for empirical runtime on large graphs. Discuss if they appear to be linear. If not, explain the possible reasons.
4. Discussions (general implications and issues),
5. Conclusions (summarize the lab and point to a future direction).

Submit the source code files and your lab report online.