Mahlet Zemedie

03/11/2018

Lab4 Report

Introduction

A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph. Graphs are used to solve many real-life problems. Graphs are also used in social networks like linkedIn, Facebook. Depth First Traversal (or Search) for a graph is similar to Depth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again.

Method

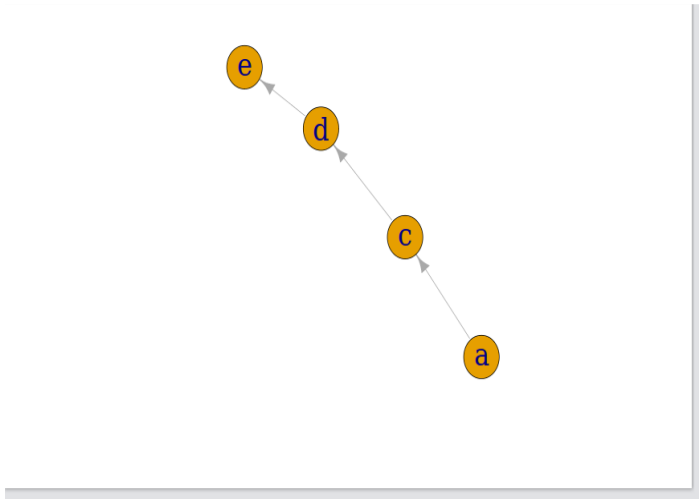
The purpose of this lab is to design data structures to represent graphs using adjacency lists. Two class is used to represent the graph; node and graph class. An input file must be scanned, and a graph needs to be created and then the generated graph needs to be written to another file. The input and output graphs are visualized using R. The file is read using the scan method and the graph is created using the graph data structure. The addition of edges is done in constant time by using sets and hash table. The other task is to design and implement Depth First Search using recursive and Iterative methods. The methods checked for a variety of graphs like cyclic/acyclic, directed/undirected, having one or more connected components. The plots for the DFS-recursive and DFS-iterative need to be plotted as a function of number of nodes and number of edges.

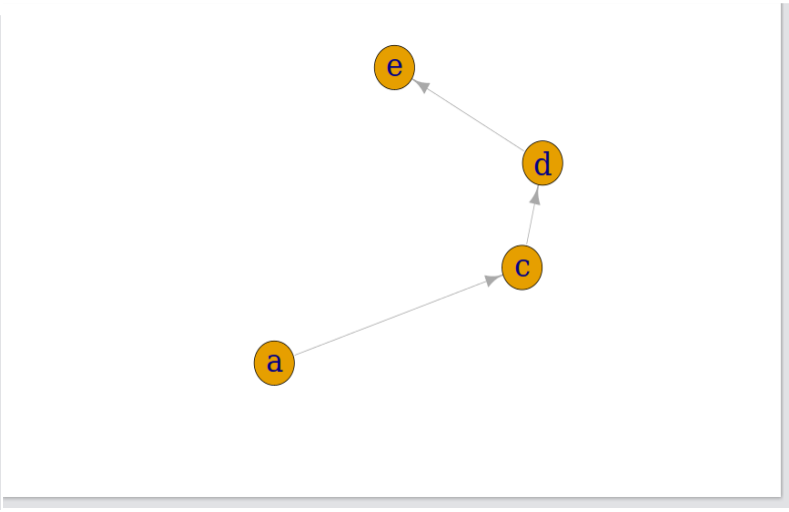
Result

Five cases are used for testing. The output for each case are plotted above. The time complexity of DFS is at least O(V). The graph is implemented using adjacency lists, wherein each node maintains a list of all its adjacent edges, then, for each node, we could discover all its neighbours by traversing its adjacency list just once in linear time. For a directed graph test cases, the sum of the sizes of the adjacency lists of all the nodes is E (total number of edges). So, the complexity of DFS is O(V) + O(E) = O(V + E). For an undirected graph, each edge will appear twice. Once in the adjacency list of either end of the edge. So, the overall complexity will be O(V) + O (2E) ~ O(V + E).

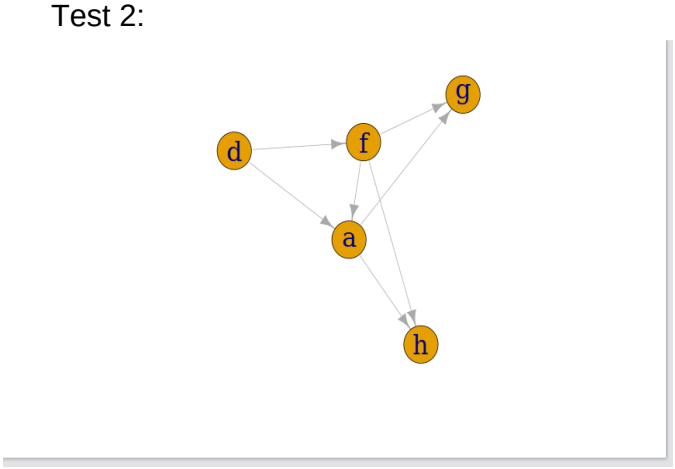
a. Figures for basic graph:

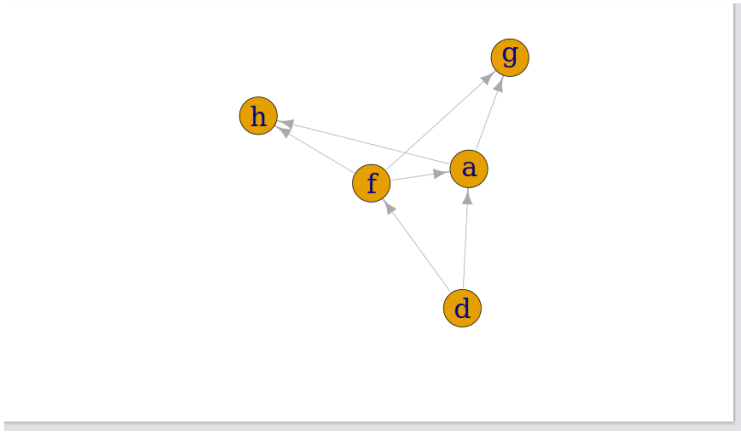
Test 1:



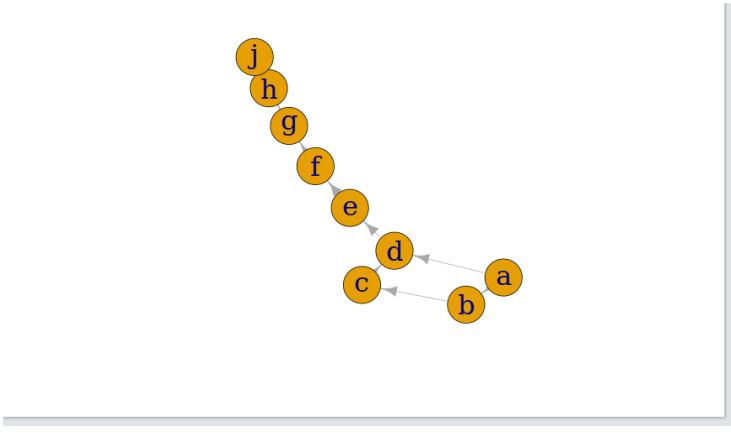


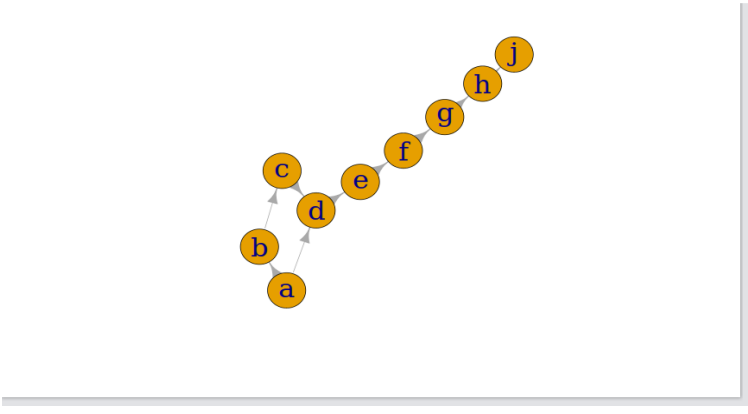
Test 2:



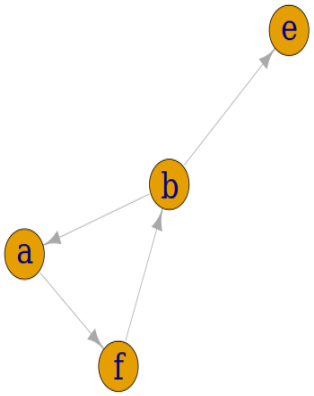


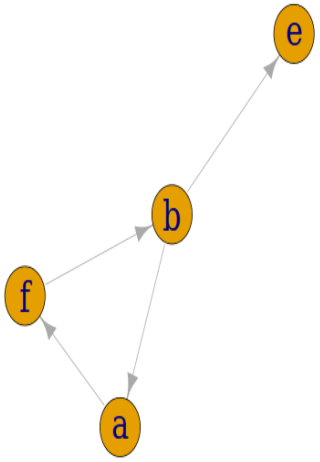
Test 3:



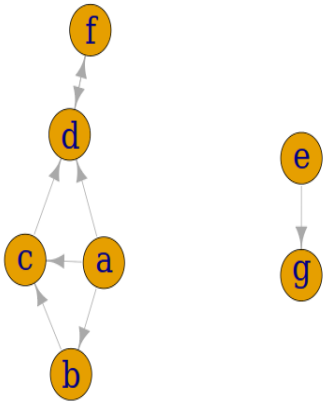


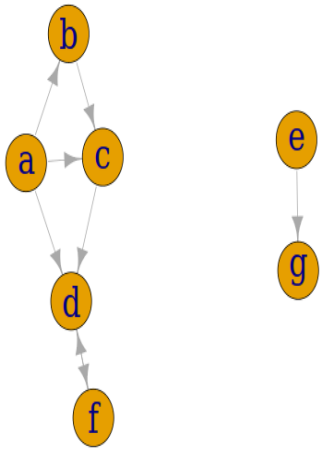
Test 4:





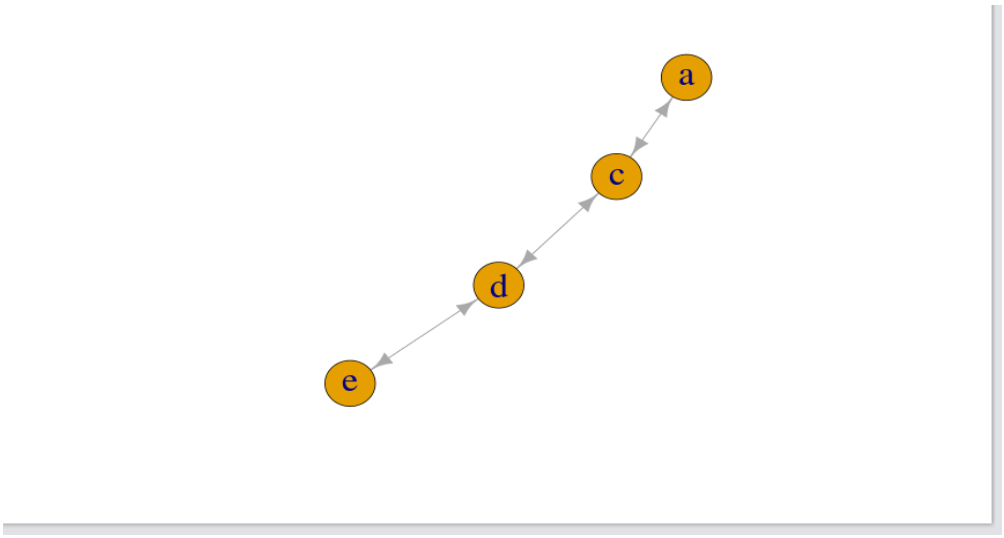
Test 5:



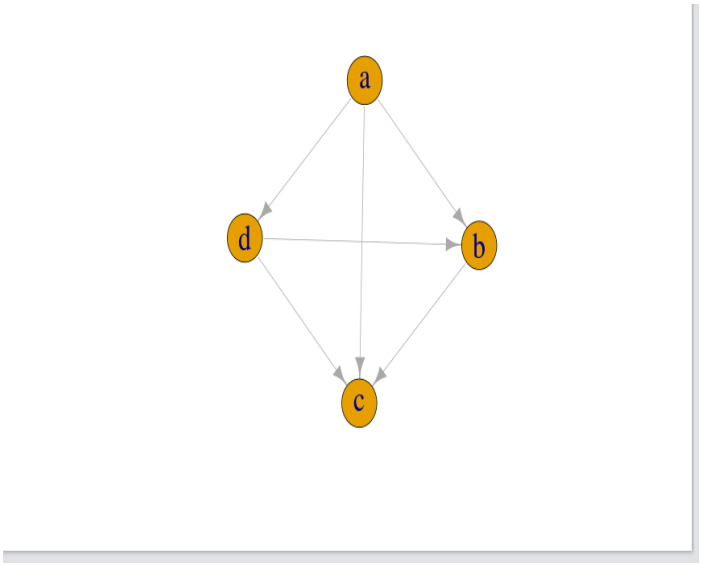


(b) Test DFS Graphs used:

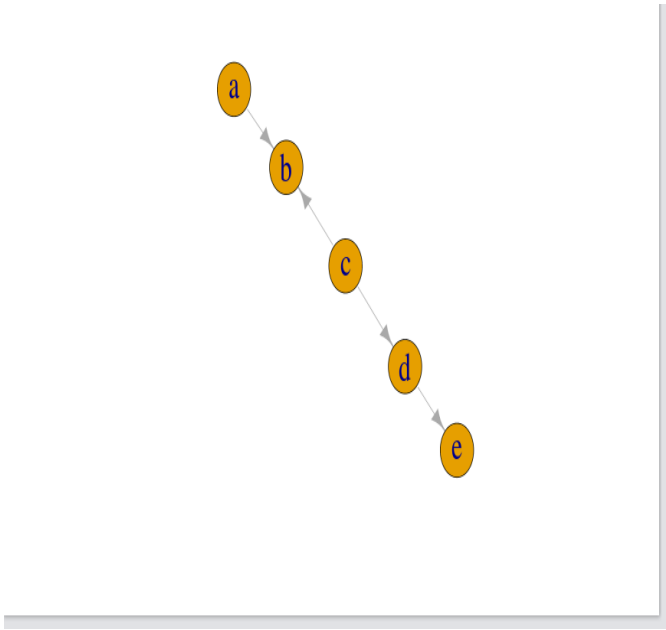
Test 1.



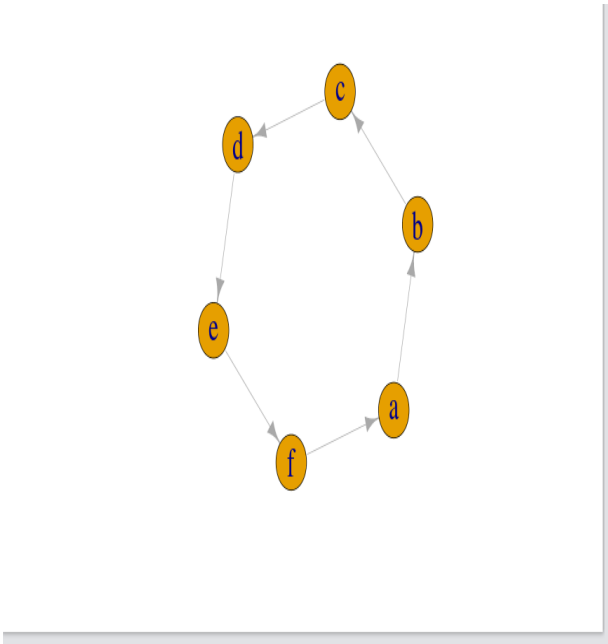
Test 2



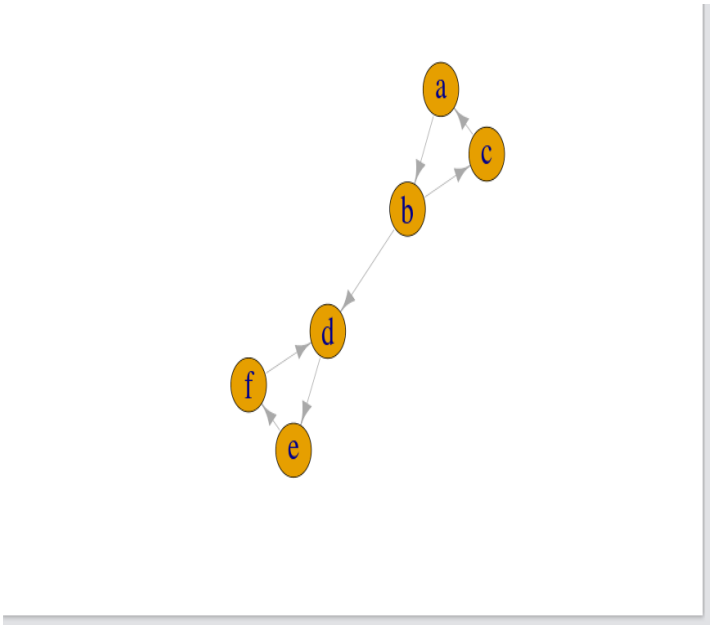
Test 3



4. Test Graph 4:

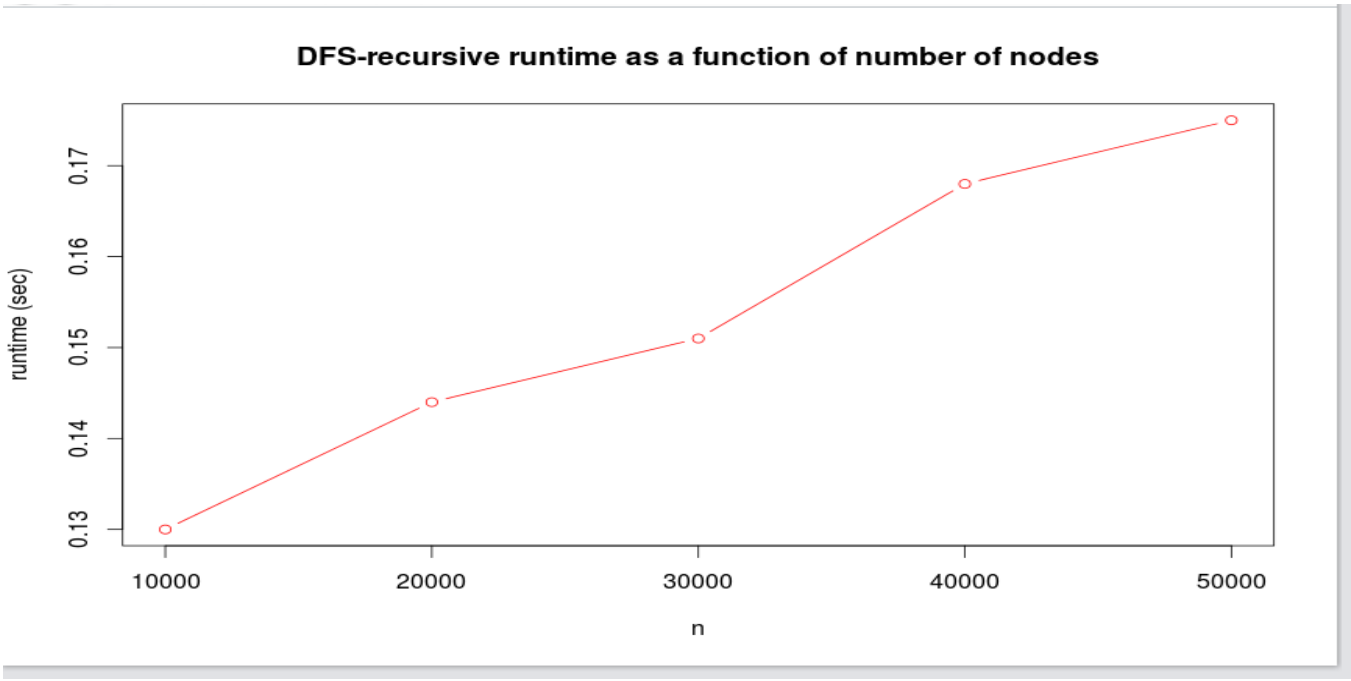


5. Test Graph 5:

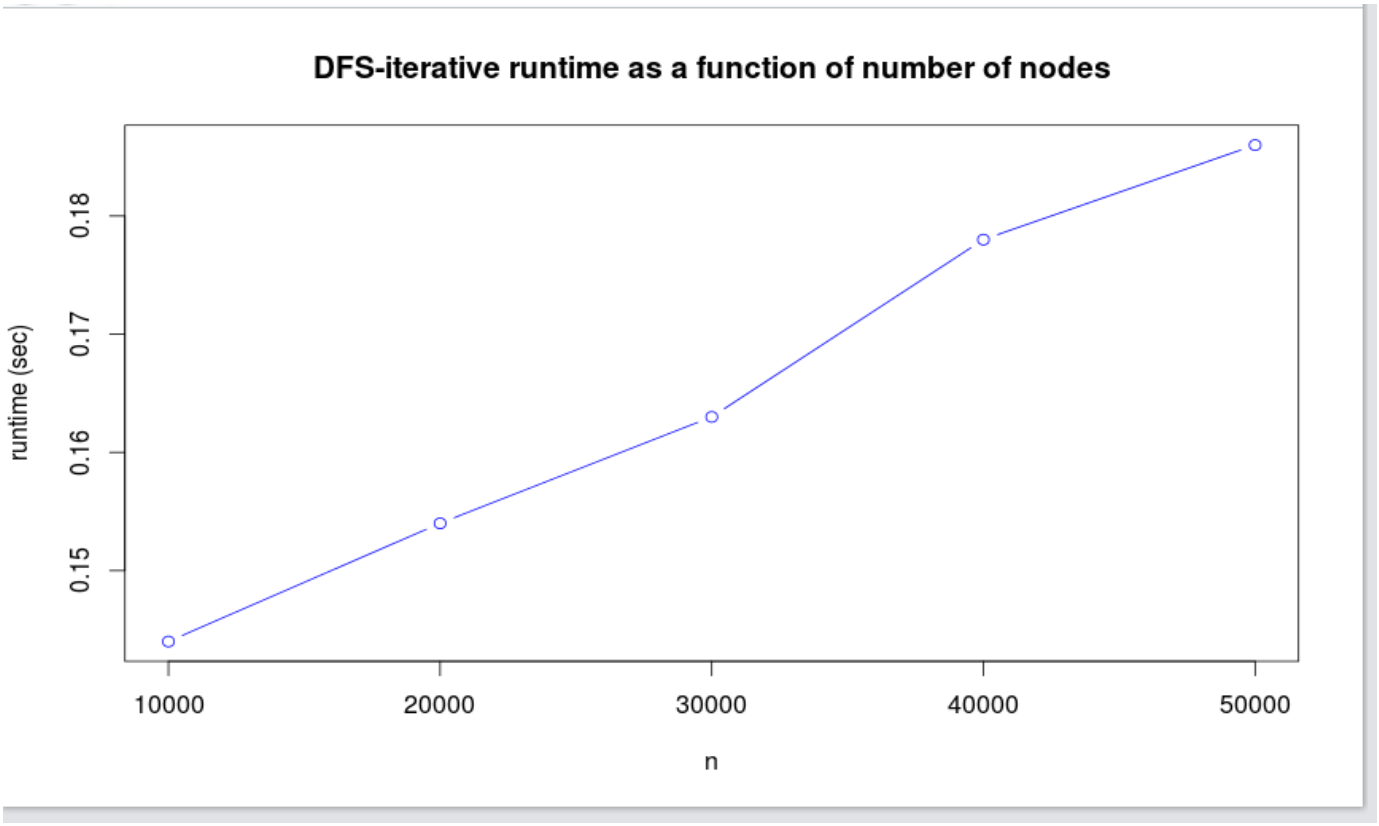


(b) Figures for empirical run-time on large graphs:

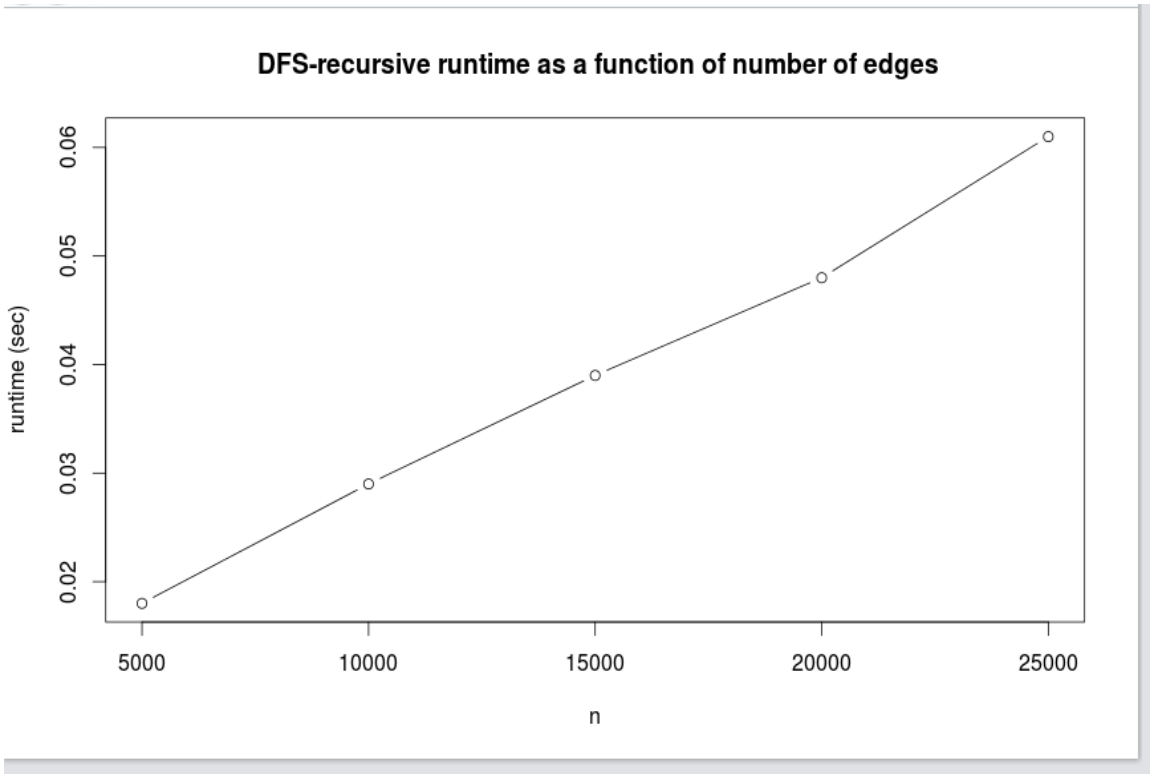
1. DFS-recursive runtime as a function of number of nodes



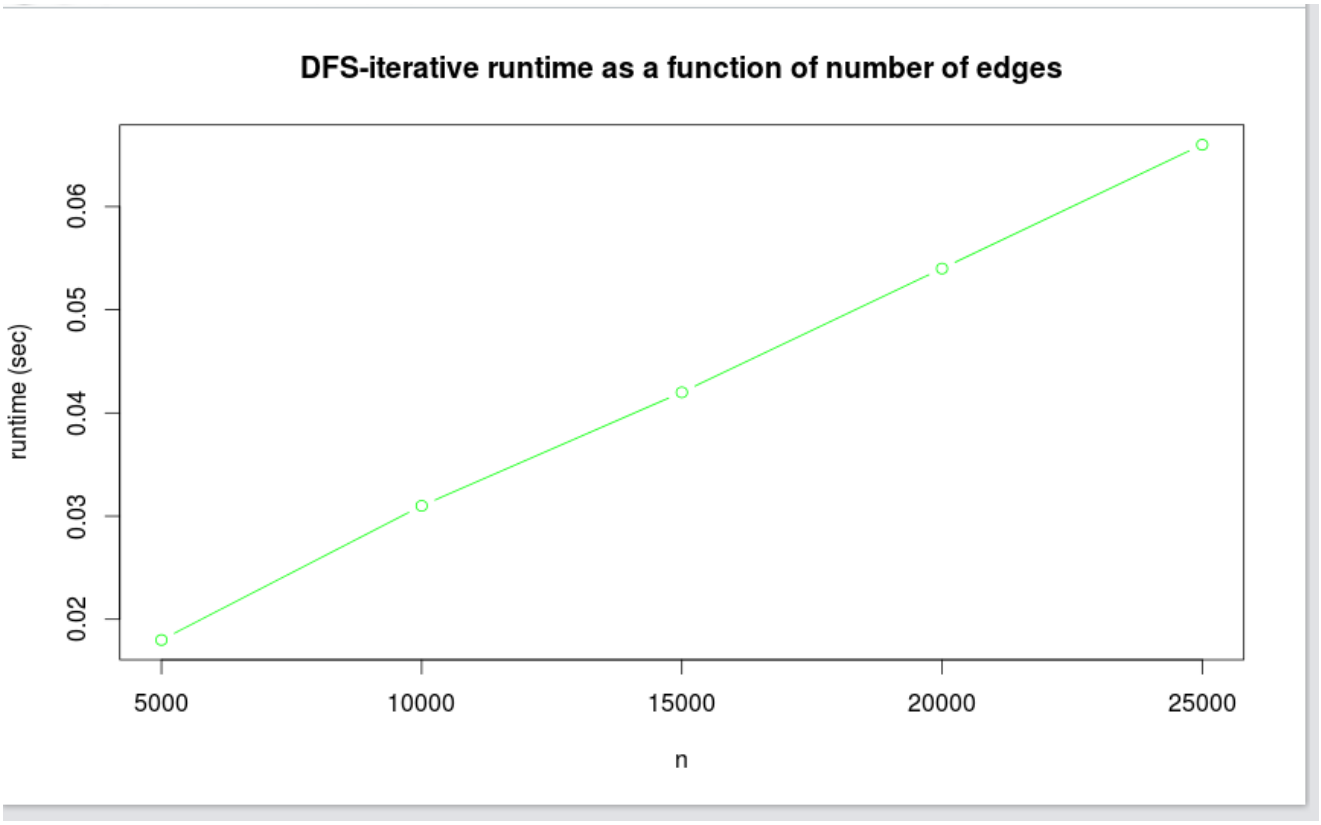
2. DFS-iterative runtime as a function of number of nodes



3. DFS-recursive runtime as a function of number of edges



4. DFS-iterative runtime as a function of number of edges



Discussions

– The graph for the Recursive function where the number of nodes is increasing for a constant value of number of edges is linear

– The graph for the Iterative function where the number of nodes is increasing for a constant value of number of edges is linear

– The graph for the Recursive function where the number of edges is increasing for a constant value of number of nodes is linear

– The graph for the Recursive function where the number of edges is increasing for a constant value of number of nodes is linear.