**KING SAUD UNIVERSITY**
**COLLEGE OF COMPUTER & INFORMATION SCIENCES**
**DEPARTMENT OF COMPUTER SCIENCE**

**Course:** Algorithms Design and Analysis - CSC311
**Semester:** Second Semester 2019/2020
**Instructors:** Dr. Mohamed Maher Ben Ismail & Dr. Fahad A Al-Dhelaan

# Theoretical and Empirical Analysis of Algorithms

*DEADLINE:* ***April 16th 2019, 23:59****.*

## 1. Introduction

Three projects are briefly described below. You are expected to select one of these topics for your project. However, in exceptional circumstances we are willing to consider other proposals that you may have on your own, provided they have a very strong relation to the contents of the course. You can discuss the basic architecture of your system, such as main data structures, main components of the algorithm, design of the user interface for input/output, etc. with me so as to get started on the right track.

## 2. Project Topics:

a) BST and Heap: Huffman coding and decoding:

Huffman Encoding is one of the simplest algorithms to compress data. Even though it is very old and simple , it is still widely used (eg : in few stages of JPEG, MPEG etc). In this project you will implement Huffman encoding and decoding. You can read up in Wikipedia or any other tutorial.

Your system must accept a file and (Huffman) tree for the same.

During you need to form a binary the construction of Huffman tree, use the priority queue to select nodes with smallest frequencies. Once you have constructed the tree, traverse the tree and create a dictionary of codewords (letter to code). Given any new sentences, your system must show how the sentence is converted to Huffman code and then decoded back to original sentence.

Note that you must implement BST and Heap yourself and must not rely on any language libraries. You can use external libraries like GraphViz to display your Huffman tree.

b) Red-Black Trees: Shared Memory de-duplication:

Red black trees are one of the most important data structures used extensively in the Linux Kernel. For eg, recent kernels use Completely Fair Scheduler for process scheduling which depends on Red Black trees!

In this project, you will use Red black trees to simulate another of their popular applications- Shared memory de-dupulication. You can read up the algorithm used at http://www.ibm.com/developerworks/linux/library/l-kernel-shared-memory/index.html .

Your system must have the following operations:

- ✓ Load: This will accept a list of <page id, hash of page content> records.
- ✓ Update: This will accept another list of <page id , hash of page content>. If page id already loaded, then just update its hash. This is equivalent to page content changing. If page id is not yet loaded, then add it to the system.
- ✓ De-dupulicate: In this operation, you must run the de-duplication algorithm explained in the website and display if any memory is freed.

c) Minimum Spanning Tree: Solving TSP for Metric Graphs using MST Heuristic:

Given an arbitrary metric graph, construct its Minimum spanning tree using Kruskal's algorithm. You can assume adjacency matrix representation of graphs. If you wish, you can reuse external libraries for heaps. Now use the constructed MST to find an approximate estimate for the TSP problem. You can choose to implement any of the two approximation algorithms specified in Wikipedia's entry on TSP – One with approximation factor of 1.5 (Christofides) or 2. Compare it with the optimal answer. You can use some external library to find the optimal solution to the TSP problem.

## 3. Experiments

Experiments have to be carried out according to the following directions.

1. Each algorithm has to be run on test data. Data will be generated in a random way.
2. If necessary, an algorithm may be run on test data a fixed number of times so that the running times obtained are meaningful

4. Running times will be plotted against input size.

## 4. Programming

Implementation of algorithms may be done in any language of student's choice. However, the language and its compiler should support certain features in order to be able to run the experiments properly. The choice of C, C++, Java, Maple, Matlab or the like should be enough. Source code has to be handed over.

## 5. Project Demonstration

Once the project is completed, the following is expected from you:

a) A demonstration of your project in which you show the various features of your system such as its correctness, efficiency, etc. You should be prepared to answer detailed questions on the system design and implementation during this demo. We will also examine your code to check for code quality, code documentation, etc.

b) You should also hand in a completed project report which contains details about your project, such as main data structures, main components of the algorithm, design of the user-interface for input/output, experimental results, e.g. charts of running time versus input size, etc.

c) You should also turn in your code and associated documentation (e.g. README files) so that everything can be backed up for future reference.

d) Email your code and all associated files with "CSC311-Project <Lastname>" as subject.

## 6. Written Report

A report describing the following points must be handed over.

• Brief explanation of the algorithms.

• Brief explanation of the implementations. It can be done by including sufficiently detailed comments in the code.

• Brief description of the experiment.

• Interpretation of experimental data. Comparison of experimental data with theoretical complexities.

• Conclusions. In this section students must draw his own conclusions (be creative).

The paper has to be written in correct English; it also has to possess clarity of thought.
Show me what you know; do not force to search for it through a poorly written paper.

## 7. Grading

We will take points off when:

• There is spelling mistakes

• It is plenty of irrelevant material. Down with the irrelevant!

• It lacks clarity of thought.

• It is lengthy, long-winded or poor in content.

• Code is not properly commented.

• Code is not properly structured.

• Variables have absurd names.

• There are run-time errors.

## 8. Questions and Office Hours

Mr. Rami, Dr. Fahad and Dr. Maher are willing to answer your questions about algorithms, complexity or the experiment. They will not answer questions about coding errors as it is our feeling that, at this point, writing error-free code is your responsibility.