Course Project
Analysis of Algorithms OPER 628

Due: 6 December 2024

The course project counts as 25% of your final grade and consists of you implementing various algorithms on a variety of test instances for a computational comparison.

1. Selection Sort, Quick Sort, Heap Sort, and Bucket Sort were all implemented in the Sorting Algorithms Lab.

   a. Implement Merge Sort in the programming language of your choice (but ensure all sorting algorithms are in the same language).

   b. Perform a series of benchmarking tests to observe performance of the various algorithms. Your tests should include input sequences that are very "random" looking, along with sequences that are "almost sorted" or "almost reverse sorted." The length/size of the input sequences should also span a range of test instances from only a few numbers, to a much larger input. Write a short report describing your code and results of your testing. (Note: You will have to modify Bucket Sort from the Lab to accept an arbitrary range of values).

2. In the Network Algorithms Lab, Dijkstra's Algorithm was implemented using both a naive approach to find a temporary node with the smallest distance label, and an improved implementation using binary heap.

   a. Perform a series of benchmarking tests to observe performance of the naive and heap implementation. Your tests should include a variety of network sizes (e.g., range in number of nodes) and densities (e.g., range in number of arcs in the network). Write a short report describing your code and results of your testing.

3. In the Network Algorithms Lab, Kruskal's Algorithm was implemented using both a naive and improved approach to detect cycles. Similarly, Prim's Algorithm was implemented using both a naive approach to find the smallest cost arc to build the tree, and an improved implementation using a binary heap.

   a. Perform a series of benchmarking tests to observe performance of the naive and improved cycle detection approach to Kruskal's, and a naive approach to find the smallest cost arc to build the tree and improved implementation using a binary heap for Prim's. Your tests should include a variety of network sizes (e.g., range in number of nodes) and densities (e.g., range in number of arcs in the network). Write a short report describing your code and results of your testing.

The grading rubric includes:

1. Your delivery of an informative, well presented report of your analysis (around 6000 - 7000 words).
2. An oral presentation covering the major results (i.e., ~15 minutes)
3. Sufficiency of generated test instances
4. Coding and implementing algorithms to the desired $O(\cdot)$ runtime.
5. Quality of the written report
6. Organization of the final presentation

Both the written report and slides are due NLT 0900 on 6 December 2024.