CSCI 2720

Data Structures

Project 3

Sorting Algorithms

Fall 2016

**Team Members**

1. Name UGA ID CRN
2. Name UGA ID CRN
3. Name UGA ID CRN
4. **Introduction**

In this section, you should describe the objective of this project, and briefly introduce the experiment and this report contents.

1. **Theoretical Analysis of Sorting algorithm**

In this section, you will provide an analysis of the complexity of each of the sorting algorithm in terms of time and number of comparisons on the different input types. At the end of this section you will also provide your expectations about the complexity for each algorithm and input type.

1. **Experimental Setup**

In this section, you should provide a description of your experimental setup, which includes but is not limited to:

* 1. Machine specification
  2. How did you generate the test inputs? Why did you use different input files and What input sizes did you test?
  3. How do you measure time? Do you take an average of many execution? How many?
  4. Did you use extra memory space or other data structures other than an array? If so, explain when and why?

1. **Experimental Results**

In this section, for each algorithm, you should compare its theoretical performance to the actual performance (Execution time). Then you will compare them to one another in terms of actual performance and the number of comparisons done.

**4.1 O(n2) Sorting Algorithms**

Compare O(n2) sorting algorithms (Selection, insertion and Bubble) in terms of execution time then number of comparisons on the three different data sets. Discuss based on the results of your experiment.

Support your discussion with “*at least*” two plots:

1. A plot of Running time (y-axis) and number of elements (x-axis).
2. A plot of number of comparisons (y-axis) and number of elements (x-axis).

You might want to add extra notes on one of the sorting algorithms by inserting its specific details (see insertion chart on sheet2 in the excel file)

*Note: see the excel file plots.xlsx (The data in the file is unreal, just any numbers, SO, replace with your experiment results) you can create any chart by selecting the desired data.*

**4.2 O (n log n) Sorting Algorithms**

Compare O (n log n) sorting algorithms (Heap, merge and quick) in terms of execution time and number of comparisons. Show a table of your results and 2 plots similar to previous subsection 4.1.

1. **Concluding Remarks**

Provide a discussion of your results, which includes but is not limited to:

* + - 1. Compare the theoretical analysis with the experimental analysis. To what extent does the theoretical analysis agree with the experimental results? Attempt to understand and explain any inconsistencies you note.
      2. Comparing algorithms to each other which one is the fastest or the best? Does it depend on the order of the data to be sorted or number of elements? Which algorithms was most impressive with respect to amount of work done (number of comparisons)

Show a table and a plot of the running time for each algorithm on the three data sets, using different number of elements.

See table 1 below and the excel file.

Table 1: Running time vs. number of elements

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Number of elements** | | |  |  |  |  |  |
| **Sort Type** | **data type** | **10** | **100** | **500** | **1000** | **10000** | **20000** | **100000** | **200000** |
| Insertion | Inorder |  |  |  |  |  |  |  |  |
| Insertion | Random |  |  |  |  |  |  |  |  |
| Insertion | Reverse |  |  |  |  |  |  |  |  |
| Selection | Inorder |  |  |  |  |  |  |  |  |
| Selection | Random |  |  |  |  |  |  |  |  |
| Selection | Reverse |  |  |  |  |  |  |  |  |
| Merge | Inorder |  |  |  |  |  |  |  |  |
| Merge | Random |  |  |  |  |  |  |  |  |
| Merge | Reverse |  |  |  |  |  |  |  |  |
| Quick | Inorder |  |  |  |  |  |  |  |  |
| Quick | Random |  |  |  |  |  |  |  |  |
| Quick | Reverse |  |  |  |  |  |  |  |  |

**References:**

[1] Please insert references: books, papers, and web references

[2]