CS 5800 Final Project Proposal:
Comparison and evaluation of Heap-Sort and Quick-Sort Algorithms

Name: Yuanzhi Xu, Kejian Tong, Ting Ji

Final Project: Comparison and evaluation of Heap-Sort and Quick-Sort Algorithms

Introduction

I. Context

Sorting is a fundamental operation in computer science (many programs use it as an intermediate step). A large number of sorting algorithms have been made in order to have best performance in terms of computational complexity (best, average and worst), memory usage, stability and method. One sorting algorithm can perform much better than another. The common sorting algorithms include quick sort, merge sort, heap sort, selection sort, insertion sort, bubble sort, etc. These sorting algorithms mentioned above are all comparison based, which are lower bounded by O(nlogn). Quick-Sort and Heap-Sort are two representative sorting algorithms which are widely used in computer science since they guarantee O(nlogn) complexity if implemented properly, which is the best performance we can achieve with comparison-based sorting. However, we got to know that most built-in sort algorithms these days are usually Quick-Sort.

II. Why we choose this topic

Since sorting can often reduce the complexity of a problem, it is an important algorithm and fundamental operation in Computer Science. Sorting algorithms have direct applications in searching algorithms, database algorithms, divide and conquer methods, data structure algorithms, and many more.A large number of sorting algorithms have been made in order to have the best performance in terms of computational complexity (best, average, and worst), memory usage, stability and method. Quick-Sort and Heap-Sort are two representative sorting algorithms that are widely used in computer science since they guarantee O(nlogn) complexity if implemented properly, which is the best performance we can achieve with comparison-based sorting. However, we got to know that most built-in sorting algorithms these days are usually Quick-Sort. Why is Quick-Sort more popular than Heap-Sort, given that Quick-Sort's performance can downgrade to O(n^2) while Heap-Sort's complexity is always O(nlogn). We hope to look into this and see if we can find out the reason.

III. Defined questions

1. There are differences between Quicksort and heap sort. How much are they different?
2. Does one of them beat the other in some factors?
3. How do we evaluate the two algorithms according to the used method, stability, memory usage, and complexity?
4. Are there any enhancements to these algorithms?

IV. Why are each of you personally interested in those questions?

1. Yuanzhi Xu:

Learning arithmetics and mathematics since I was a child, I find that numbers are an indispensable part of our daily life. And people are never tired of inventing new algorithms for numbers, such as Fibonnaci numbers, prime numbers, and so on. As we can collect a larger number of data, it is important to understand the connection and relations of the data. And sorting is one of the well-applicable algorithms in our daily life. It is easier and faster to locate items in a sorted list than unsorted. Sorting algorithms can be used in a program to sort an array for later searching or writing out to an ordered file or report. Since sorting can often reduce the complexity of a problem, it is an important algorithm in Computer Science. These algorithms have direct applications in searching algorithms, database algorithms, divide and conquer methods, data structure algorithms, and more in the data-driven world. This concludes why I'm personally interested in sorting and prompts me to come up with these questions.

2. Kejian Tong

Before I started this course, algorithms were a black box to me and I didn't really realize how useful algorithms were in life. I was always curious about some things in daily life. For example, in real life, we tend to break things down along useful lines. If we were to sort the change, we would first divide the coins by denomination, then add each denomination, and then add them together. For another example, Commercial Computing is used in various government and private organizations for the purpose of sorting various data like sorting files by name/date/price, sorting of students by their roll no., sorting of account profile by given id, etc. Algorithms are also invisible codes that tell a computer how to accomplish a specific task. We can think of it as a recipe for a computer: an algorithm tells the computer what to do to produce a specific result. Every time we do a Google search or check the Facebook feed or navigate with GPS in the car, and we are interacting with the algorithm. With these questions and doubts, I am interested in understanding the algorithmic logic behind these life scenarios, why it is Quick Sort, and why it is sometimes Heap Sort.

3. Ting Ji

As a computer science major student, I have been introduced to many sorting algorithms such as selection sort, bubble sort, insertion sort, merge sort, quick sort and heap sort. However, I sometimes feel a bit confused about which sorting algorithm to use since there are so many sorting algorithms to choose from. Especially for Heap-Sort and Quick-Sort, although their average performances are both O(nlogn), why do commercial applications stick with Quick-Sort instead of Heap-Sort while Quick-Sort's worst-case performance is significantly worse than Heap-Sort? There must be some reason behind this. I think this is a very interesting problem to investigate. And I can learn a lot from this investigation, such as what Heap-Sort and Quick-Sort are, how they work, what their complexities are, how they are different from each other and when we favor one over the other.

Scope

This outline of the analysis defines the scope of the project given the short time frame of this final project

a. QuickSort
   - How it works
   - Stability
   - Memory Usage
   - Coding Algorithm
   - Complexity

b. HeapSort
   1. How it works
   2. Stability
   3. Memory Usage
   4. Coding Algorithm
   5. Complexity

c. Comparisons table

d. Enhancement
   1. QuickSort
   2. HeapSort

Description of the plan:

We already:

- Confirmed the topic of the project
- Defined the questions we wish to answer
- Set the scope of the project

We plan to:

- Learn all the sorting algorithms thoroughly again
- Looking for application of sorting algorithm in the industry
- Conduct research according to the outline of the scope
- Compare HeapSort and QuickSort from different aspects