CPSC 350 Data Structures Assignment 6, Sorting Algorithms

Jaewon Park
Chapman University
Orange, CA
jaepark@chapman.edu

Abstract—This includes different types of Sorting Algorithms and shortcomings of this empirical analysis

I. Introduction

There are different types of sorting algorithms: Bubble Sort, Insertion Sort, Selection Sort, and Merge Sort. These sorting algorithms are all used in different situations.

II. Sorting Algorithms Differences

A. What are some differences among these sorting algorithms?

Bubble Sort

- It is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.
- The bubble sort repeatedly compares adjacent elements of an array. The first and second elements are compared and swapped if out of order.
- Using the bubble sort technique, sorting is done in passes or iteration. Thus at the end of each iteration, the heaviest element is placed at its proper place in the list. In other words, the largest element in the list bubbles up.
- For run-time error, best case would be O(n) and worst case would be O(n2).

Insertion Sort

- It is a simple sorting algorithm that works the way we sort playing cards in our hands.
- General idea is that at any time we have a marker in the array.
- Everything to the left of the marker is partially sorted.
- We then take the item at the marker, find where it belongs in the sorted values to the left, and insert it.
- This of course involves shifting down values to make room.
- For random data, it will run 2x as fast as bubble sort, and still faster than selection sort.
- For pre-sorted data, runs much better than others – O(n) because nothing happens in the while loop.
- For run-time error, best case would be O(n) and worst case would be O(n2).

Selection Sort

- A lot less swaps than bubble sort, however still slow
- It is a combination of searching and sorting.
- During each pass, the unsorted element with the smallest (or largest) value is moved to its proper position in the array.
- The number of times the sort passes through the array is one less than the number of items in the array.
- It has O(n2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort.

Merge Sort

- Most effective for larger data sets.
- It uses the "divide and conquer" strategy wherein we divide the problem into subproblems and solve them individually.
- For run-time error, it would be o(nlogn)

Quick Sort

- Divide and conquer algorithm
- Users pick pivot and partitions the given array around the picked pivot.
- For run-time error, best case would be O(nlogn) and worst case would be O(n2).

III. Conclusion

There are different types of Sorting Algorithms and they are all used in different situations. After trying the time complexity of each of them, we can notice that those are effective, but sometimes not. After trying this assignment, I would choose Quick sort that shows the best performance in the average case for most inputs and it is also considered the "fastest" sorting algorithm.