Ben Guericke

CSCI 232

Program 1 write up

10/7/19

**Bubble Sort:**

Bubble sort works by comparing adjacent numbers and if they are out of order than it swaps them. Due to the simplistic nature of Bubble Sort I did not need to make any assumptions about the sort, however it is important to note that bubble sort is great for almost sorted data set and for slight error checking in a sorted data set.

**Cocktail Sort:**

Cocktail sort builds off the ideas of bubble sort with comparing adjacent elements and then swapping them. It first does a first loop of a regular bubble sort which ends up putting the largest element at the end of the array. Given that the largest element is going to be at the end of the array we do not need to sort that element so variable of the end of the array is decreased by one. Then we use a reverse bubble sort that goes from the end of the array to the front resulting in the smallest element being in the lowest position. Then like the first bubble sort that element is in the proper place, so it increases the starting element of the sort by 1. Cocktail sort is also stable because it builds off of the stable sorting algorithm bubble sort.

**Radix Sort:**

Radix sort was the most difficult sort to write because it requires the extra step of finding the number of places of the numbers being sorted. One assumption that I had to make while writing this class was that the elements in of the type comparable were integers. This is because there are places in other data types. When sorting the elements by the ith place I utilized insertion sort because it was stable and does not require additional memory.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Big O notation | Runtime N=100,000 | Predicted N=200,000 | Predicted N=300,000 | Correlates? |
| Bubble Sort | O(n^2) | 20.286s | 81.144s | 182.574s | Yes it does because as the big O increases the runtime increases at the same rate |
| Cocktail Sort | O(n^2) | 25.41s | 101.64s | 228.69s | Same as above |
| Radix Sort | O(W \* n^2 ) W= Number of places | 21.844 | 158.122 | 237.183s | Same as above |