Luc Dang

CPSC 335-02

Project 2: Sort Race

Team: Poggy

Complexity Order Paper

Insertion Sort:

Insertion sort loops through i=0 to the length of the string = N, and a nested loop that loops from j=i to 0 at the worst. There is a comparison each time which makes T(N) = N *N = N^2. Therefore the complexity for Insertion Sort is O(N^2).

Poresort:

Pore sort is implemented to compare all even-indexed characters with the next indexed character, and if all are in order then a 'oddsorted' flag is put up. Then the odd-indexed characters are compared with their next indexed character and if all are in order then a 'evensorted' flag is put up. If any character is out of order then the sort will continue to loop until both flags are true. For each loop there is N/2 comparisons, and since there is an even and odd loop, T(N) = (N/2) * (N/2) which is N^2. Therefore the complexity for Poresort is O(N^2).

Mergesort:

Mergesort was implemented with recursion which divides the N sized string in half until sublist size N is 1 character long. This division process is T(LogN), then the merging process is T(N) for the N=1 character sublists. So T(N) = N*(LogN), and the complexity of Mergesort is O(N*LogN).

Quicksort:

Quicksort was also implemented with recursion using partitioning. A pivot value is chosen and the characters on both side of the pivot value are compared which T(N). Quicksort is called on each half so T(N) = LogN since the list is continually cut in half. Therefore the complexity of Quicksort is O(N*LogN)