1806ICT Programming Fundamentals

Searching and Sorting

1. Implement ternary search and compare its performance with binary search.

2. Using the provided C *qsort()* implement a sort that sorts all odd length strings before any even length strings in addition to the odd length strings being sorted descending and the even length strings being sorted ascending. The input file is *dictionary.txt* and your program should write its output to the file *sorted_dictionary.txt*

3. Implement binary insertion sort and compare its performance with insertion sort.

4. Implement a radix sort for 6 digit integers..

5. Shellsort (more accurately Shell's sort) is an important sorting algorithm which works by applying insertion sort to each of several interleaving sub-lists of a given list. On each pass through the list, the sublists in question are formed by stepping through the list with an increment h_i taken from some predefined decreasing sequence of step sizes, $h_1 > ... > h_i > ... > 1$, which must end with 1. (The algorithm works for any such sequence, though some sequences are known to yield a better efficiency than others. For example, the sequence 1, 4, 13, 40, 121, ... , used, of course, in reverse, is known to be among the best for this purpose.)

    a. Apply Shellsort to the list

        S, H, E, L, L, S, O, R, T, I, S, U, S, E, F, U, L

    b. Is Shellsort a stable sorting algorithm?

    c. Implement Shellsort in C and compare its performance on random arrays of sizes 10^2, 10^3, 10^4, and 10^5 as well as on increasing and decreasing arrays of these sizes with insertion sort.