Name: Sang Hun Kim

PUID: 0025808690

2/3/2017

Programming Assignment 1 Report

From this assignment, it was asking to create a sorted array from unsorted array given as binary file using either insertion sort or selection sort within shell sort. For generating the sequence of the Pratt sequence (2p\*3q), it was concluded that the best-case time complexity to generate the sequence was O(1), and the worst-case time complexity was O(n2). The reason behind that best-case time complexity being O(1), which is said to be constant time complexity, is because when the size of the given input binary file has either no values or a single integer value, then there is no need to be sorted, in which the input binary file has already been sorted. For this being said, it will run through the *if* statement in shell\_sort\_seq(int Size) function written in sorting.c file only once. However, whenever the input binary file has more than one integer value, then it will go through *else* statement. From the first two *while* statement, it is shown that it has O(n) time complexity. As program goes on, it counters nested for loops. Each nested for loops have time complexities of O(n2), because each outer loops executes their statement n times (in linear), and each inner loops also executes their statements n times since the number in the while statement argument increases linearly. Thus, it was concluded that the worst-case time complexity was O(n2) and best-case time complexity was O(1) for generating Pratt sequence. Furthermore, the space complexity for generating Pratt sequence is also O(n2) because the long integer array of number sequence is created in heap memory depending on the values in nested loops.

As shown below from the table, they are the outputs after sorting unsorted integer arrays. From the sorting functions, it can be concluded that the time complexity and space complexity for both Shell\_Insertion\_Sort and Shell\_Selection\_Sort are O(n4) because of the following reasons. For time complexity, each sorting functions go through three while nested loop with inner for loop, each having linear time complexity. However, since they are all combined as nested loop, it can be concluded that the worst case time complexity for both sorting methods are O(n4). Same as space complexity, because there is a memory allocation for temporary subarrays in the very most inner nested loop, in which values in subarrays are apart from each other in original unsorted array by value corresponding to shell sort sequence, space complexity is also O(n4).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Insertion Sort / Selection Sort | I/O time | Sorting Time | # Moves | #Comparison |
| 15 int | 0.000000e+00  0.000000e+00 | 0.000000e+00  0.000000e+00 | 1.050000e+02  2.250000e+02 | 8.300000e+01  2.330000e+02 |
| 1,000 int | 0.000000e+00  0.000000e+00 | 0.000000e+00  0.000000e+00 | 4.004100e+04  9.286500e+04 | 3.477000e+04  1.474965e+06 |
| 10,000 int | 0.000000e+00  2.500000e-01 | 0.000000e+00  2.500000e-01 | 6.929290e+05  1.652133e+06 | 6.144430e+05  1.496031e+08 |
| 100,000 int | 9.000000e-02  2.511000e+01 | 9.000000e-02  2.511000e+01 | 1.061606e+07  2.581623e+07 | 9.536590e+06  1.499419e+10 |
| 1,000,000 int | 1.720000e+00  (Too slow to generate) | 1.710000e+00  (Too slow to generate) | 1.509460e+08  (Too slow to generate) | 1.367297e+08  (Too slow to generate) |