Siddharth Prabhu

ECE 368

PA02

1)

|  |  |  |
| --- | --- | --- |
| File Size (long integers) | I/O Run Time(seconds) | I/O (Sorting Time)  (seconds) |
| 15 | 0 | 0 |
| 1000 | 0 | 0 |
| 10000 | 2.06 | 2.06 |
| 100000 | 587 | 588 |
| 1000000 | N/A | N/A |

I used a shell insertion algorithm to sort my linked lists. The run times as seen from the table above increased as the size of the file increased. The time complexity of shell sort using insertion sort was worst case O(n^2) and O (nlogn) is the time complexity if the list is sorted.

The space complexity includes, a linked list for the sequences and a linked list of linked lists. The space complexity for the sequence generation of linked lists is O ((log n ^2)). This is because if the sequence can be assumed to be in a triangular form both the height and base will be (log n) elements assuming the size of the list is n. I made a linked list of linked lists so taking into account the space complexity for that the overall space complexity should be O (n).

Compared to assignment 1, the I/O run times is pretty similar as one element was read at a time just as assignment 1 .The I/O sorting time got above are much slower as linked lists were used. The values were not swapped instead pointers were updated and set to corresponding elements. It is very easy to jump to points on an array as opposed to a linked list. The traversal function in my code was one of the main reasons why my program efficiency declined as the size of the files increased. Traversing to find the previous node for larger lists is just not efficient.