Lab 11: Sorting

CSC 2054

100 points

Part II:

Do the same thing from part I for shell sort, heap sort, merge sort, and quicksort.

**Pitfall**: you must make sure the vector you sort is in the original randomized list.

Run the program for list sizes of 100, 1000,10000, and 20000

Report the runtime results of your program below (10 points): You must have completed the part II to earn the points for these table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Heap Sort | Quick Sort | Shell Sort | Merge Sort | Insertion Sort |
| 100 | 102 | 40 | 78 | 99 | 212 |
| 1000 | 1658 | 638 | 1610 | 1428 | 17581 |
| 10000 | 23011 | 8257 | 25944 | 21820 | 1724075 |
| 20000 | 44715 | 16640 | 55643 | 48814 | 6940525 |

Now create another list, which is in reverse order (i.e., the worst case for many algorithms).

Try running the program on the worst case list with the following sizes and write the times in the following table: (10 points)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Heap Sort | Quick Sort | Shell Sort | Merge Sort | Insertion Sort |
| 100 | 95 | 38 | 72 | 93 | 361 |
| 1000 | 2132 | 567 | 1735 | 1646 | 37569 |
| 10000 | 20618 | 5398 | 17256 | 17398 | 3564255 |
| 20000 | 43278 | 11086 | 37407 | 36536 | 13783275 |

Now create another list, which is already sorted.

Try running the program on the sorted list with the following sizes and write the times in following table: (10 points)

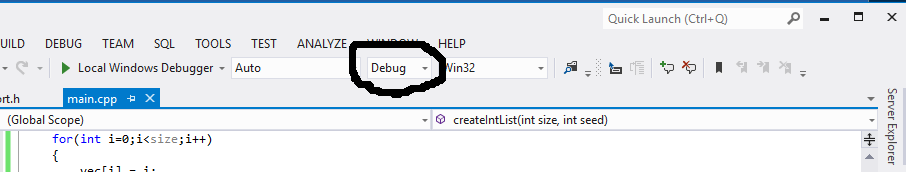
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Heap Sort | Quick Sort | Shell Sort | Merge Sort | Insertion Sort |
| 100 | 39 | 40 | 54 | 111 | 12 |
| 1000 | 519 | 903 | 1240 | 1406 | 123 |
| 10000 | 5434 | 8034 | 13123 | 19432 | 1250 |
| 20000 | 8756 | 17805 | 27713 | 37336 | 2601 |

Part II of your program works. (30 points)

Part III:

Run your program in release mode (or the Mac equivalent)

See the dropdown menu set of debug in the screenshot below? Change it to release and note how the performance changes. Release mode removes all the debug information (meaning, you don’t want to write a program in release mode – only use release mode after your program successfully runs).



Run the program for list sizes of 10,000, 50,000, and 100,000 (using the random list)

Report the time results of your program below: (10 points) [if it runs super-fast then increase the sizes)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Heap Sort | Quick Sort | Shell Sort | Merge Sort | Insertion Sort |
| 10000 | 754 | 508 | 1769 | 1143 | 18009 |
| 50000 | 4994 | 3691 | 7176 | 6388 | 428402 |
| 100000 | 9741 | 6284 | 14122 | 13322 | 1692230 |
| 200000 | 23186 | 13366 | 29872 | 26302 | 472474 |

Describe how the performance changes between debug and release. (6 points)

**The performance is extremely fast in release mode compared to debug mode. Even with the random vector it runs faster than debug runs through a sorted vector of the same size.**

Why is the number of inversions a negative number when you use seed 5 with 100,000 items? [use release mode or it might take a very long time to run] (4 points)

**Because the number of inversions at that point is too absurdly huge for the computer to comprehend so it represents it as a negative value.**