MP3

To Generate Test Data

./geninput x y z 1 | ./lab3 a

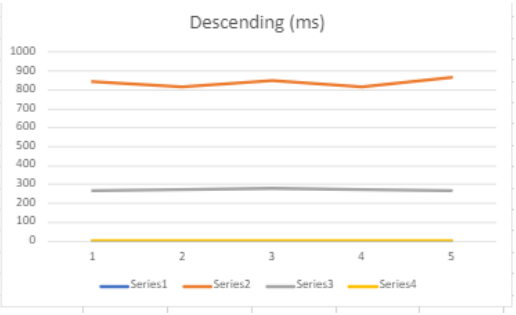
- X = list size. Use 12000 for all except merge. Use 25000 for merge
- Y = what order to generate the list
    - 1: ascending
    - 2: random
    - 3: descending
- Z = the type of sort
    - 1: bubble
    - 2: insertion
    - 3: recursive selection
    - 4: iterative selection
    - 5: merge
- A = which way to sort the new list
    - 1: ascending
    - 2: descending

1. For lists that are initially random, explain the differences in running time for the sorting algorithms. Do your iterative and recursive selection sort algorithms show dramatic differences in running times or are they similar? Why does the mergesort algorithm show a dramatic improvement in run time? If the runtime for merge sort is not dramatically faster than the other algorithms you have a bug.
    a. All the algorithms, except for merge, all significantly increased in run time due to having to sort the entire list into an ascending order. As the item list size increased, the run time slowly increased.
    b. Both iterative and recursive selection sort had a lower run time when the list was sorted into descending order. As for ascending and random order, both had relatively similar run times
    c. Merge sort runs on an O(nlogn) and is the most efficient of the sorts because instead of constantly iterating and moving nodes, it just splits the list, compares, then moves nodes around till all nodes are separated. Then, it reverses the order and moves nodes around as the nodes are being put back together.
2. If a list is already in ascending or descending order, some sort algorithms are very fast while others still have to perform a similar number of comparisons as when the list is not sorted. Describe which algorithm(s) show extremely fast performance if the list is already sorted and explain why.
    a. Bubble sort is faster when sorted because it's essentially just has to reverse the list if in descending or just iterate through the list once if in ascending
    b. Insertion sort is faster when sorted because, similar to bubble sort, it either just iterates through the list once if in ascending or moves a couple nodes around if in descending

c. Recursive and merge had relatively similar run times for all 3 sort types.

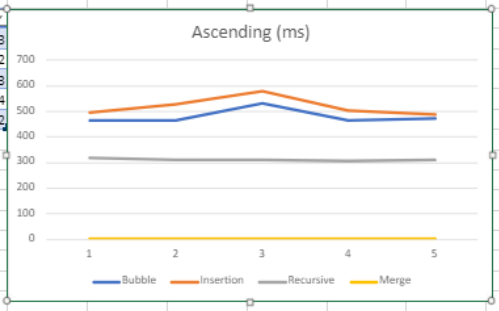
Ascending (ms)

List Size	Bubble	Insertion	Recursive	Merge
100	0.14	845.8	268.4	2.3
500	0.14	814.2	274.6	3.2
1000	0.16	850.7	278.6	2.4
10000	0.14	816.4	274.5	2.3
50000	0.15	867.3	271.1	2.3



Descending (ms)

List Size	Bubble	Insertion	Recursive	Merge
100	465	494.9	317	2.3
500	463.6	527.3	308.8	2.2
1000	532.5	578.5	311.9	2.3
10000	465.2	504.8	308.5	2.4
50000	472.6	486.9	310.9	2.2



Random (ms)

List Size	Bubble	Insertion	Recursive	Merge
100	658.4	1301.8	304.2	2.3
500	643.8	1283.6	286.4	2.5
1000	639.7	1307.9	298.1	2.5
10000	658.2	1320	289.5	2.2
50000	624.2	1380.4	316.2	2.2