Original List: 7 8 6 15 12 1 17 2 4 14 5 13 3 7

14 elements

CS 372 HW Sorts

3/30/21

1.) Bubble Sort (greatest to least)

14 elements 14 passes Compare the two elements and place larger first

After 1st pass: 8 7 15 12 6 17 2 4 14 5 13 3 7 1
2nd pass: 8 15 12 7 17 6 4 14 5 13 3 7 2 1
3rd pass: 15 12 8 17 7 6 14 5 13 4 7 3 2 1
4th pass: 15 12 17 8 7 14 6 13 5 7 4 3 2 1
5th pass: 15 17 12 8 14 7 13 6 7 5 4 3 2 1
6th pass: 17 15 12 14 8 13 7 7 6 5 4 3 2 1
7th pass: 17 15 14 12 13 8 7 7 6 5 4 3 2 1
8th pass: 17 15 14 13 12 8 7 7 6 5 4 3 2 1
9th-14th passes: no changes, but they run regardless

2.) Selection Sort (greatest to least)

Linearly move through list and find the smallest unsorted element then swap it. As before we will need 14 passes or 13 since the last is sorted.

After 1st pass: 7 8 6 15 12 7 17 2 4 14 5 13 3 1
2nd pass: 7 8 6 15 12 7 17 3 4 14 5 13 2 1
3rd pass: 7 8 6 15 12 7 17 13 4 14 5 3 2 1
4th pass: 7 8 6 15 12 7 17 13 5 14 4 3 2 1
5th pass: 7 8 6 15 12 7 17 13 14 5 4 3 2 1
6th pass: 7 8 14 15 12 7 17 13 6 5 4 3 2 1
7th pass: 13 8 14 15 12 7 17 7 6 5 4 3 2 1
8th pass: 13 8 14 15 12 17 7 7 6 5 4 3 2 1
9th pass: 13 17 14 15 12 8 7 7 6 5 4 3 2 1
10th pass: 13 17 14 15 12 8 7 7 6 5 4 3 2 1
11th pass: 15 17 14 13 12 8 7 7 6 5 4 3 2 1
12th pass: 15 17 14 13 12 8 7 7 6 5 4 3 2 1
13th pass: 17 15 14 13 12 8 7 7 6 5 4 3 2 1
14th pass: No change.

7 8 6 15 12 1 17 2 4 14 5 13 3 7

3.) Insertion (greatest to least) Create a seperate list and take elements from one list and then bubble sort as you go. Still need 14 passes.

After 1st pass: 7

2nd pass: 8 7

3rd pass: 8 7 6

4th pass: 15 8 7 6

5th pass: 15 12 8 7 6

6th pass: 15 12 8 7 6 1

7th pass: 17 15 12 8 7 6 1

8th pass: 17 15 12 8 7 6 2 1

9th pass: 17 15 12 8 7 6 4 2 1

10th pass: 17 15 14 12 8 7 6 4 2 1

11th pass: 17 15 14 12 8 7 6 5 4 2 1

12th pass: 17 15 14 13 12 8 7 6 5 4 2 1

13th pass: 17 15 14 13 12 8 7 6 5 4 3 2 1

14th pass: 17 15 14 13 12 8 7 7 6 5 4 3 2 1

4.) Shell sort (least to greatest) also insertion sort

Select some k, this could be any integer number that is less than 1/2 the number of elements. Inspect the first element and compare it to the k^{th} element after it and sort how needed. This process is faster than the other 3 sorts listed thus far, but needs other sorts to finish.

k = 5 decrease after each pass

k = 4 1st pass 7 1 6 2 4 8 5 13 3 17 15 12 14 7

k = 3 2nd pass 2 1 6 5 4 3 7 13 8 7 14 15 12 17

k = 2 3rd pass 2 1 4 3 6 5 7 7 8 13 12 15 14 17

Now after 3 passes, we can use a simpler sort to swap the slightly grouped numbers

1 2 3 4 5 6 7 7 8 12 13 14 15 17

Original: 7 8 6 15 12 1 17 2 4 14 5 13 3 7

Dustin Dorsey

CS 372 HW Sorts

3/30/21

5.) Quick Sort (smallest to greatest)

8 6 15 12 1 17 2 4 14 5 13 3 7
i j

3 6 15 12 1 17 2 4 14 5 13 8 7
i j

3 6 5 12 1 17 2 4 14 15 13 8 7
i j

3 6 5 4 1 17 2 12 14 15 13 8 7
i j i

2 3 6 5 4 1 7 17 12 14 15 13 8 7

Now 7 is sorted and two lists (left and right) are created. Use the same sort for each side

1 2 6 5 4 3 7 17 12 14 15 13 8 7
1 2 6 5 4 3 7 17 12 14 15 13 8 7
1 2 3 5 4 6 7 17 12 14 15 13 8 7
1 2 3 5 4 6 7 17 12 14 15 13 8 7
1 2 3 4 5 6 7 17 12 14 15 13 8 7
1 2 3 4 5 6 7 17 12 14 15 13 8 7
1 2 3 4 5 6 7 7 12 14 15 13 8 17
1 2 3 4 5 6 7 7 8 12 15 13 14 17
1 2 3 4 5 6 7 7 8 12 15 13 14 17
1 2 3 4 5 6 7 7 8 12 14 13 15 17
1 2 3 4 5 6 7 7 8 12 13 14 15 17
1 2 3 4 5 6 7 7 8 12 13 14 15 17

(a) Heap Sort (least to greatest)

We create a heap and delete the top element and reheap each time

7 8 6 15 12 1 17 2 4 14 5 13 3 7 n2 parent → left child 2n; right child 2n+1

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Building Heap Array style (saves space)

7, 8 7, 8 7 6, 15 8 6 7, 15 12 6 7 8, 15 12 6 7 8 1, 17 12 15 7 8 1 6

17 12 15 7 8 1 6 2, 17 12 15 7 8 1 6 2 4, 17 14 15 7 12 1 6 2 4 8,

17 14 15 7 12 1 6 2 4 8 5, 17 14 15 7 12 13 6 2 4 8 5 1,

17 14 15 7 12 13 6 2 4 8 5 1 3, 17 14 15 7 12 13 7 2 4 8 5 1 3 6

Heap: 17

14 15

7 12 13 7

2 4 8 5 1 3 6

Heap Sort | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 8 | 12 | 13 | 14 | 15 | 17 |

Delete the top and reheap add top node to end

15 14 13 7 12 6 7 2 4 8 5 1 3 Rebuild Heap each time

14 12 13 7 8 6 7 2 4 3 5 1 Always take deleted root

13 12 7 7 8 6 1 2 4 3 5

12 7 7 5 8 6 1 2 4 3

left 7 8 7 7 5 3 6 1 2 4

right 7 7 7 6 5 3 4 1 2

7 5 6 2 3 4 1

6 5 4 2 3 1

5 3 4 2 1

4 3 1 2

3 2 1

2 1

1

□