Elijah Rourke Algo Hw 2

1) Step by step to merge two sorted lists (mergesort)

list 1: [1,2,3,6] list 2: [-3,0,6,7] Sol: [ ]

we compare the first element of both lists and add the smaller value to the sorted merged array.

L1: [1,2,3,6] L2: [-3,0,6,7] | Sol [ ] 1>-3

L1: [1,2,3,6] L2: [0,6,7] | Sol [-3] 1>0

L1: [1,2,3,6] L2: [6,7] | Sol [-3,0] 1<6

L1: [2,3,6] L2: [6,7] | Sol [-3,0,1] 2<6

L1: [3,6] L2: [6,7] | Sol [-3,0,1,2] 3<6

L1: [6] L2: [6,7] | Sol [-3,0,1,2,3] 6<6

L1: [ ] L2: [6,7] | Sol [-3,0,1,2,3,6]

append rest of L2

→ Sol [-3,0,1,2,3,6,6,7]

2) insertion sort step by step

list: [-21,5,7,-10,61,8,3,10]

i=1 -21<5 no shift ← [-21,-10,5,7,61,8,3,10]

i=2 5<7 no shift ←

i=3 7<-10 not true so swap. 5<-10 swap. -21<-10 stop.

i=4 7<61 no shift

i=5 61<8 not true so swap. 7<8 true so stop.
[-21,-10,5,7,8,61,3,10]

i=6 61<3 not true. it swaps all the way to index 2.

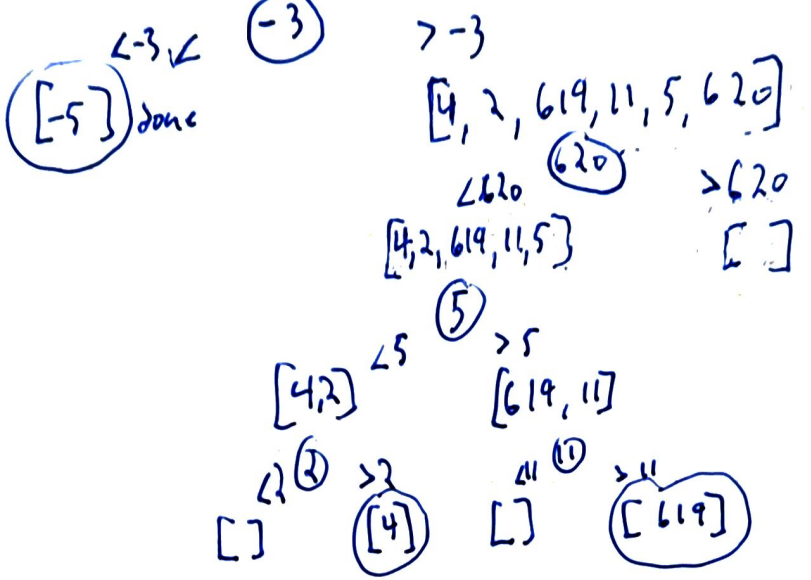
[-21,-10,3,5,7,8,61,10]

i=7 61<10 not true so we swap once and stop

we end with [-21, 10, 3, 5, 7, 8, 10, 61]

3) quick sort

List: [-5, 4, 2, 619, 11, 5, 620, -3] Pivot 3



Sorted List [-5, -3, 2, 4, 5, 11, 619, 620]

4) Shell Sort Steps. n = 8

List: [5, 10, 60, 0, -1, 34, 6, 10]

start with gap = n/2 (4 in this case)

pass 1: (0, 4) (1, 5) (2, 6) (3, 7)

(0, 4) → 5 > -1 swap → [-1, 10, 60, 0, 5, 34, 6, 10]

(1, 5) → 10 < 34 keep

(2, 6) → 60 > 6 swap → [-1, 10, 6, 0, 5, 34, 60, 10]

(3, 7) 0 < 10 keep

[1, 10, 6, 0, 5, 34, 60, 10]
(0,2) (1,3) (2,4) (3,5) (4,6) (5,7)

gap = n/4 = 2
(0,2) → -1 < 6 no swap
(1,3) → 10 > 0 swap → [-1, 0, 6, 10, 5, 34, 60, 10]
(2,4) → 6 > 5 → swap → [-1, 0, 5, 10, 6, 34, 60, 10]
(3,5) → 0 < 34 no swap
(4,6) → 6 < 60 no swap
(5,7) → 34 > 10 swap → [-1, 0, 5, 10, 6, 10, 60, 34]

Gap = 1 ~~Strada~~ normal insertion now

insert 6: [-1, 0, 5, 6, 10, 10, 60, 34]
insert 34: -1, 0, 5, 10, 10, 34, 60]

5) Algorithim analysis

1) quick sort & merge sort. I think these are the fastest since it splits the list then goes through each element so its n and log n. it is a divide and conquer Algorithim

2) Shell sort. Since it's like insertion but you move in gaps and the numbers can jump I think its faster than insertion

3) insertion, selection, bubble sort. These are all O(n²) I think which would make them all the slowest. Since its two loops it's n·n = O(n²)