3.1

#	Algorithm	Running time	Best Case	Worst Case
1	Bubble Sort (Non-Recursive)	O(n^2)	O(n) when array is sorted	O(n^2)
2	Bubble Sort (Recursive)	O(n^2)	O(n) when array is sorted	O(n^2)
3	Selection Sort (Non-Recursive)	O(n^2)	O(n^2)	O(n$^{2)}$
4	Insertion Sort (Non-Recursive)	O(n^2)	O(n) when array is sorted	O(n^2)
5	Merge Sort (Recursive)	O(n·log(n))	O(n·log(n))	O(n·log(n))

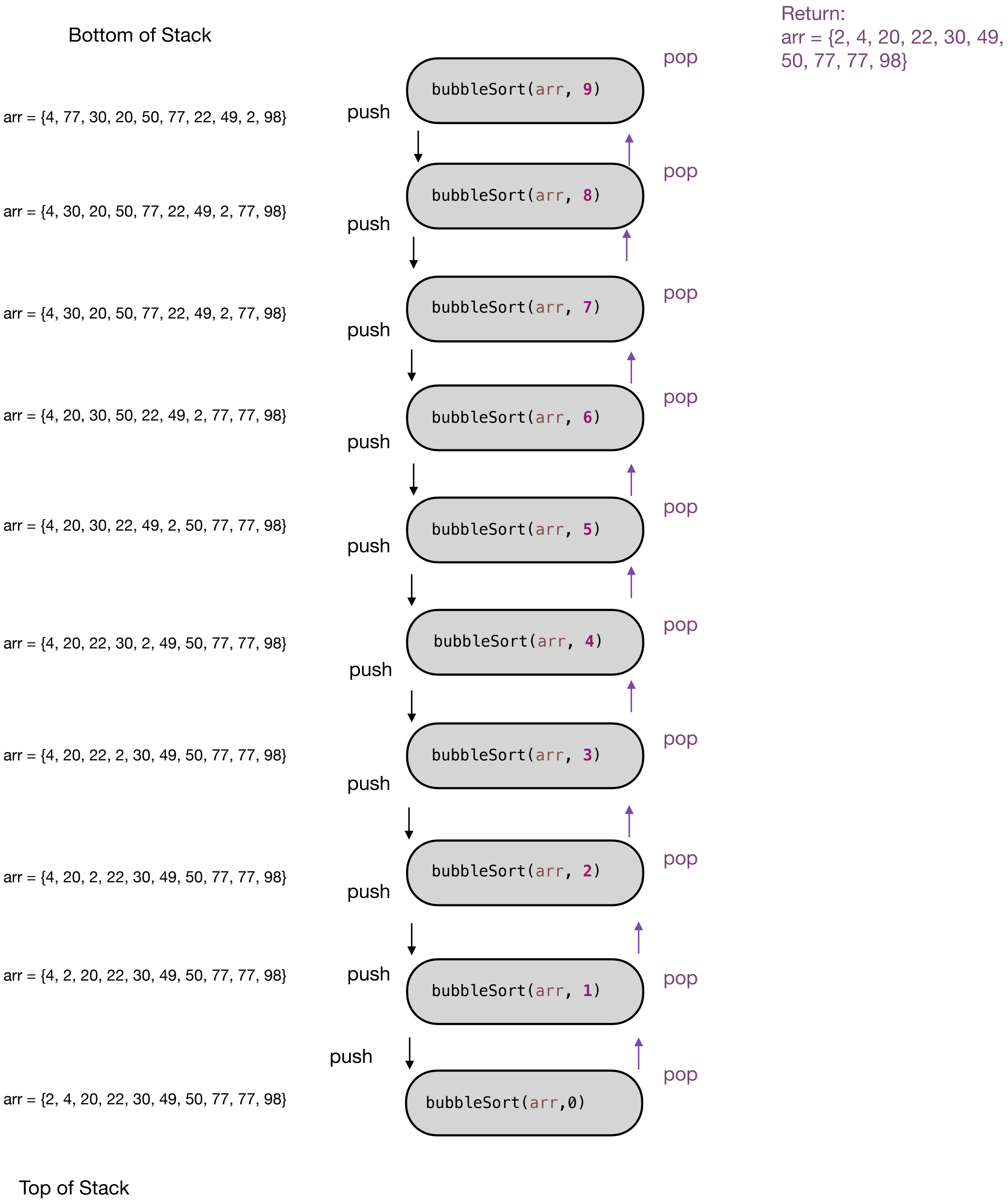
3.2

Bubble Sort

4	77	98	30	20	50	77	22	49	2
i= 0, j < 9. After 1st pass:									
4	77	30	20	50	77	22	49	2	**98**
i = 1, j < 8. After 2nd pass:									
4	30	20	50	77	22	49	2	**77**	**98**
i = 2, j < 7. After 3rd pass:									
4	20	30	50	22	49	2	**77**	**77**	**98**
i = 3, j < 6. After 4th pass:									
4	20	30	22	49	2	**50**	**77**	**77**	**98**
i = 4, j < 5. After 5th pass:									
4	20	22	30	2	**49**	**50**	**77**	**77**	**98**
i = 5, j < 4. After 6th pass:									
4	20	22	2	**30**	**49**	**50**	**77**	**77**	**98**
i = 6, j < 3. After 7th pass:									
4	20	2	**22**	**30**	**49**	**50**	**77**	**77**	**98**
i = 7, j < 2. After 8th pass:									
4	2	**20**	**22**	**30**	**49**	**50**	**77**	**77**	**98**
i = 8, j <1. After 9th pass:									
2	**4**	**20**	**22**	**30**	**49**	**50**	**77**	**77**	**98**
i = 9 > length -1, return.									

Bubble Sort Recursive Stack Trace : bubbleSort(int[ ] arr, int end)

arr = {4, 77, 98, 30, 20, 50, 77, 22, 49, 2}, end = arr.length -1 = 9



Bottom of Stack

Return:
arr = {2, 4, 20, 22, 30, 49, 50, 77, 77, 98}

arr = {4, 77, 30, 20, 50, 77, 22, 49, 2, 98}
push
bubbleSort(arr, 9)
pop

arr = {4, 30, 20, 50, 77, 22, 49, 2, 77, 98}
push
bubbleSort(arr, 8)
pop

arr = {4, 30, 20, 50, 77, 22, 49, 2, 77, 98}
push
bubbleSort(arr, 7)
pop

arr = {4, 20, 30, 50, 22, 49, 2, 77, 77, 98}
push
bubbleSort(arr, 6)
pop

arr = {4, 20, 30, 22, 49, 2, 50, 77, 77, 98}
push
bubbleSort(arr, 5)
pop

arr = {4, 20, 22, 30, 2, 49, 50, 77, 77, 98}
push
bubbleSort(arr, 4)
pop

arr = {4, 20, 22, 2, 30, 49, 50, 77, 77, 98}
push
bubbleSort(arr, 3)
pop

arr = {4, 20, 2, 22, 30, 49, 50, 77, 77, 98}
push
bubbleSort(arr, 2)
pop

arr = {4, 2, 20, 22, 30, 49, 50, 77, 77, 98}
push
bubbleSort(arr, 1)
pop

push
arr = {2, 4, 20, 22, 30, 49, 50, 77, 77, 98}
bubbleSort(arr,0)
pop

Top of Stack

Selection Sort:

Select the smallest in the array and swap it with i th element in the front:

- black-boxed: to-be-swapped
- green-boxed: min in the “non-swapped” part

Swap black-boxed with green-boxed:

4	77	98	30	20	50	77	22	49	2
i = 0, min = 9, swap arr[0], arr[9]									
2	77	98	30	20	50	77	22	49	4
i = 1, min = 9, swap arr[1], arr[9]									
2	4	98	30	20	50	77	22	49	77
i = 2, min = 4, swqp arr[2], arr[4]									
2	4	20	30	98	50	77	22	49	77
i = 3, min = 7, swap arr[3], arr[7]									
2	4	20	22	98	50	77	30	49	77
i = 4, min = 7, swap arr[4], arr[7]									
2	4	20	22	30	50	77	98	49	77
i = 5, min = 8, swap arr[5], arr[8]									
2	4	20	22	30	49	77	98	50	77
i = 6, min = 8, swap arr[6], arr[8]									
2	4	20	22	30	49	50	98	77	77
i = 7, min = 8, swap arr[7], arr[8]									
2	4	20	22	30	49	50	77	98	77
i = 8, min = 9, swap arr[8], arr[9]									
2	**4**	**20**	**22**	**30**	**49**	**50**	**77**	**77**	**98**
i = 9 !< length -1, return.									

Insertion Sort:

Look at following element and decide its place on the “sorted part” based on its value.
Right shifting is important.

- orange-boxed: sorted part
- blue-boxed: unsorted part
- text-in-blue-and-bold: the following term to be placed in the sorted part.

4	**77**	98	30	20	50	77	22	49	2

i = 1= j , look at arr[1], 77 >= 4 so it is placed after 4.

4	77	**98**	30	20	50	77	22	49	2

i = 2 = j, look at arr[2], 98 >= 77 so it is placed after 77.

4	77	98	**30**	20	50	77	22	49	2

i = 3 = j, look at arr[3], 30 < 98; 30<77; 30 >=4 so it is placed after 4.

4	30	77	98	**20**	50	77	22	49	2

i = 4 = j, look at arr[4], 20 < 98; 20 < 77; 20 < 30; 20 >= 4 so it is placed after 4.

4	20	30	77	98	**50**	77	22	49	2

i = 5 = j, look at arr[5], 50 < 98; 50 < 77; 50 >= 30 so it is placed after 30.

4	20	30	50	77	98	**77**	22	49	2

i = 6 = j, look at arr[6], 77 < 98; 77 >= 77 so it is placed after 77.

4	20	30	50	77	77	98	**22**	49	2

i = 7 = j, look at arr[7], 22 < 98; 22 < 77; 22 < 77; 22 < 50; 22 < 30; 22>= 20, placed after 20.

4	20	22	30	50	77	77	98	**49**	2

i = 8 = j, look at arr[8], 49 < 98; 49 < 77…77…50 ; 49 > = 30 so it is placed after 30.

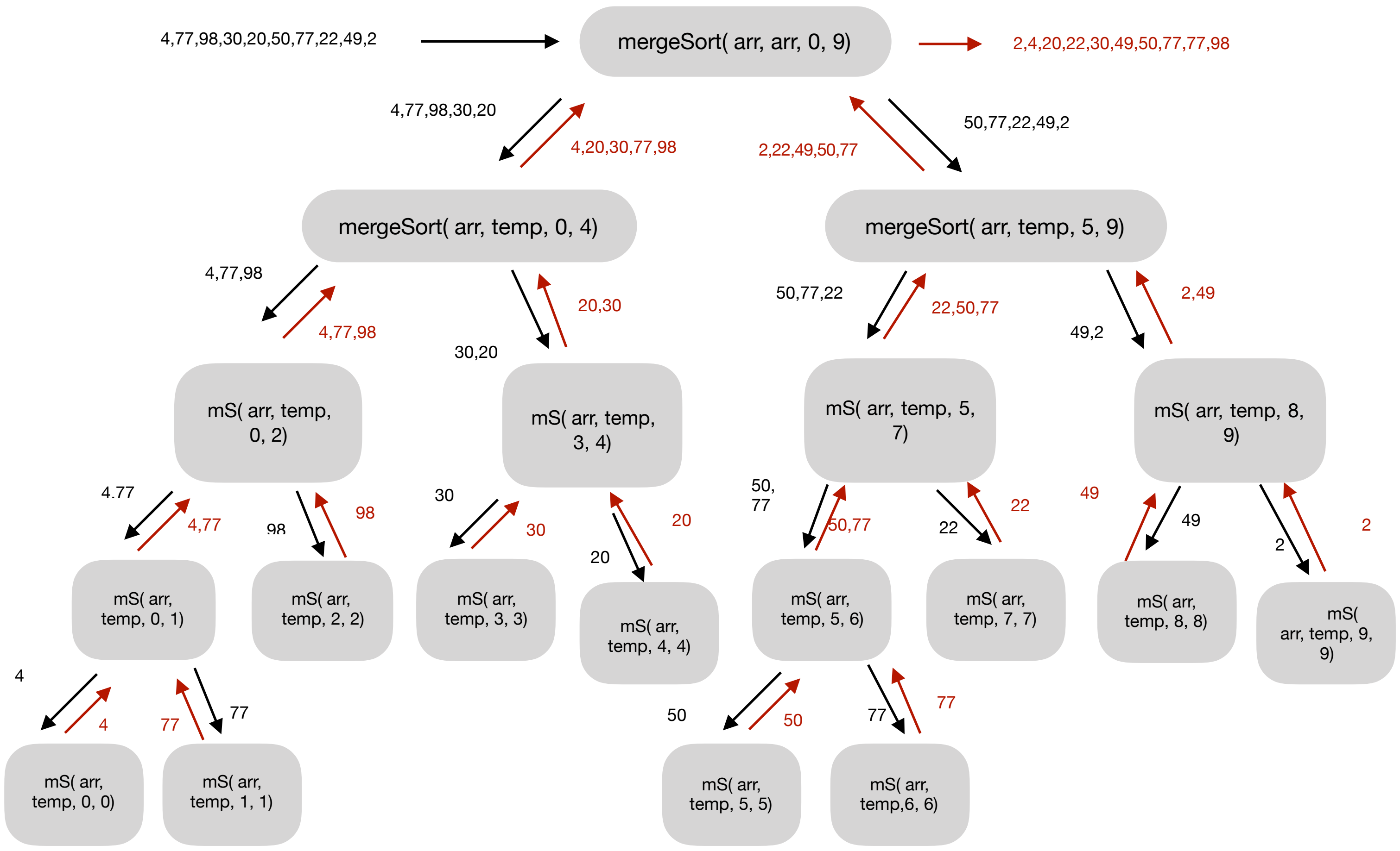
4	20	22	30	49	50	77	77	98	**2**

i = 9 = j, look at arr[9], 2 < 98…77…77…50…49…30…22…20…4, it is placed j-1 = 0, before 4.

2	4	20	22	30	49	50	77	77	98

i = 10 >= size, return.

Merge Sort (Recursive):


4,77,98,30,20,50,77,22,49,2
mergeSort(arr, arr, 0, 9)
2,4,20,22,30,49,50,77,77,98
4,77,98,30,20
4,20,30,77,98
2,22,49,50,77
50,77,22,49,2
mergeSort(arr, temp, 0, 4)
mergeSort(arr, temp, 5, 9)
4,77,98
4,77,98
20,30
30,20
50,77,22
22,50,77
2,49
49,2
mS(arr, temp, 0, 2)
mS(arr, temp, 3, 4)
mS(arr, temp, 5, 7)
mS(arr, temp, 8, 9)
4.77
4,77
98
98
30
30
20
20
50, 77
50,77
22
22
49
49
2
2
mS(arr, temp, 0, 1)
mS(arr, temp, 2, 2)
mS(arr, temp, 3, 3)
mS(arr, temp, 4, 4)
mS(arr, temp, 5, 6)
mS(arr, temp, 7, 7)
mS(arr, temp, 8, 8)
mS(arr, temp, 9, 9)
4
4
77
77
50
50
77
77
mS(arr, temp, 0, 0)
mS(arr, temp, 1, 1)
mS(arr, temp, 5, 5)
mS(arr, temp,6, 6)