COP 3502 - 3/26/24

		Worst	best		Avg
(1)	Bubble Sort	$O(n^2)$	$O(n^2)$	comparison based sorts n items	$O(n^2)$
(2)	Insertion Sort	$O(n^2)$	$O(n)$		$O(n^2)$
(3)	Selection Sort	$O(n^2)$	$O(n^2)$		$O(n^2)$
(4)	Merge Sort	$O(n \lg n)$	$O(n \lg n)$		$O(n \lg n)$

Bubble Sort

6, 2, 9, 3, 1, 5, 8, 7

2 6 9 3 1 5 8 7

2 6 9 3 1 5 8 7

2 6 3 9 1 5 8 7

2 6 3 1 9 5 8 7

2 6 3 1 5 9 8 7

2 6 3 1 5 8 9 7

2 6 3 1 5 8 7 | 9

2 3 1 5 6 7 | 8 9

2 1 3 5 6 | 7 8 9

1 2 3 5 | 6 7 8 9

1 iteration of Bubble Sort after it's done max element is in the right spot.

1st iter

2nd iter

3rd iter

4th iter

5th, 6th, 7th

Sketch of code

```
for (int j = n-1; j > 0; j--)
    for (int i = 0; i < j; i++)
        if (a[i] > a[i+1])
            swap(&a[i], &a[i+1])
```

Run time = $(n-1)+(n-2)+(n-3)+\cdots 1 = \frac{(n-1)n}{2} = O(n^2)$

Insertion Sort

AVG CASE
iteration i does $\frac{i}{2}$ swaps

$\sum_{i=1}^{n-1} \frac{i}{2} = \frac{(n-1)n}{4}$
$= O(n^2)$

[2] 9, 1, 6, 8, 5, 3, 4	
[2 9] 1 6 8 5 3, 4	1st iter
[1 2 9] 6 8 5 3 4	2nd iter
[1 2 6 9] 8 5 3 4	3rd iter
[1 2 6 8 9] 5 3 4	4th iter
[1 2 5 6 8 9] 3 4	5th iter
1 2 3 5 6 8 9 4	6th =
1 2 3 4 5 6 8 9	7th =

```
for (int i = 1; i < n; i++) {
    int j = i;
    while (j > 0 && a[j] < a[j-1]) {
        swap(&a[j], &a[j-1]);
        j--;
    }
}
```

Selection Sort

2, 9, 3, 1, 6, 4, 5, 8, 7

maxi 0
1

2 7 3 1 6 4 5 8 9 — 1st iter

2 7 3 1 6 4 5 8 9 — 2nd iter (swapped w/ itself)

2 5 3 1 6 4 7 8 9 — 3rd iter

2 5 3 1 4 6 7 8 9 — 4th iter

2 4 3 1 5 6 7 8 9 — 5th iter

2 1 3 4 5 6 7 8 9 — 6th iter

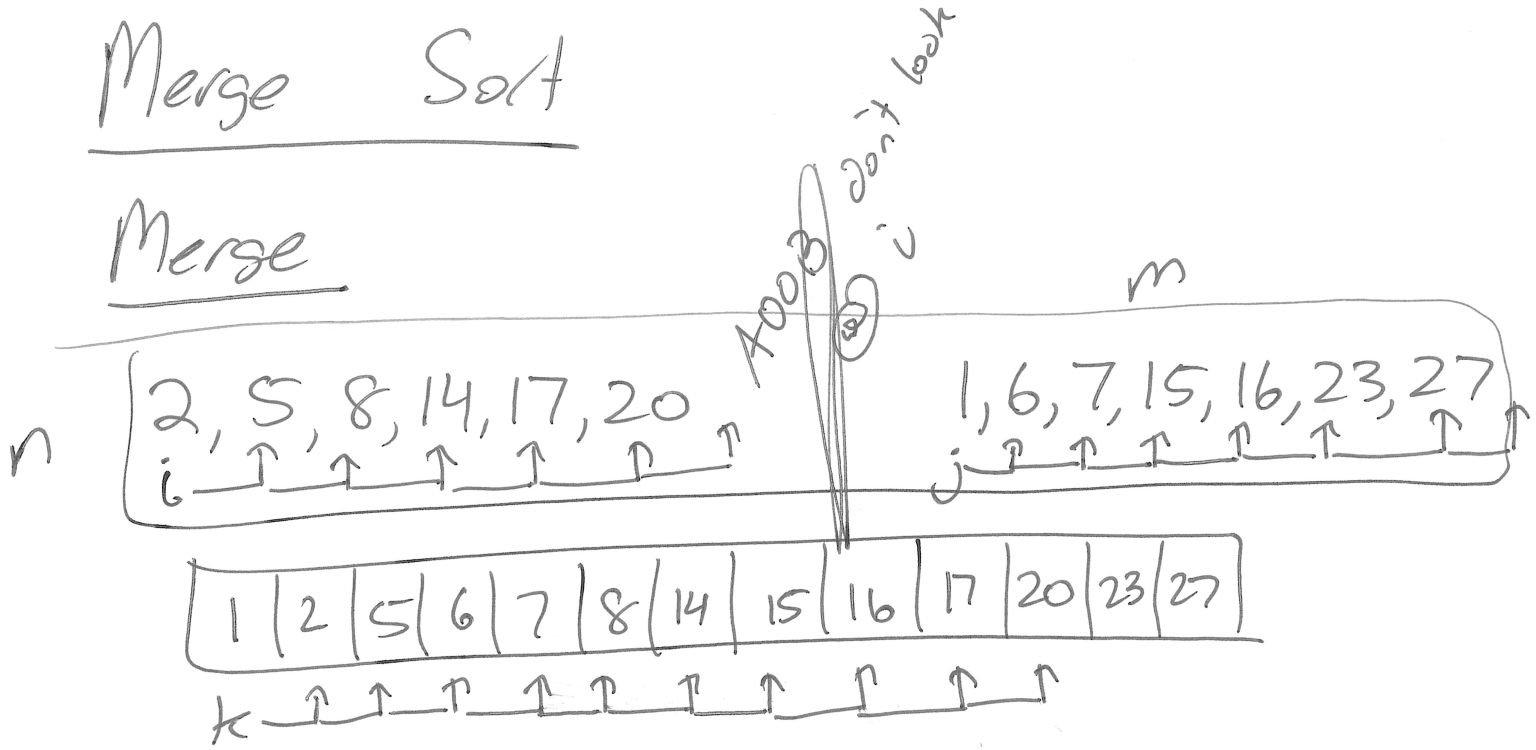
2 1 3 4 5 6 7 8 9 — 7th iter

1 2 3 4 5 6 7 8 9 — 8th iter

```
for (int j=n-1; j>0; j--) {
    int maxi = 0;
    for (int i = 0; i<=j; i++)
        if ( a[i] > a[maxi])
            maxi = i;
    swap( &a[maxi], &a[j]);
}
```

Merge Sort

Merge



Run-Time : $O(n+m)$, $O(\max(n,m))$

```
MergeSort (int* arr, int sI, int eI) {
    if (sI >= eI) return;
    int mid = (sI+eI)/2;
→   MergeSort (arr, sI, mid);
    MergeSort (arr, mid+1, eI);
    Merge (arr, sI, mid+1, eI); // Pass in 2 arrays
}
```

Sort Left Sort Right → Merge

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + O(n)$$

$$= 2T\left(\frac{n}{2}\right) + O(n)$$

Via Iteration, Master Thm $T(n) = O(n \lg n)$