START HERE
CLASS #3

Arithmetic series:

$$1+2+\cdots+n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Notice:

$1+n = n+1$

$2+n-1 = n+1$

$3+n-2$

$\vdots$

$\vdots$

$\frac{n}{2}-1+\frac{n}{2}+2$

$\frac{n}{2}+\frac{n}{2}+1 = n+1$

# of these?

$\frac{n}{2}$

START HERE
CLASS #3

Arithmetic series:

$$1+2+\cdots+n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Notice:

$1+n = n+1$

$2+n-1 = n+1$

$3+n-2$

$\vdots$

$\vdots$

$\frac{n}{2}-1+\frac{n}{2}+2$

$\frac{n}{2}+\frac{n}{2}+1 = n+1$

# of these?

$\frac{n}{2}$

Let's think about some sorts.
Let's discuss Selection Sort.

3 7 1 2 4 5 8 6

Keep current 1st
find min in list (idx)
swap to 1st

how long to do find min?

how many times do we do find min?

$n-1$
$n-2$
$n-3$
⋮
1

$$\sum_{i=1}^{n-1} i = \frac{(n-1)\cdot n}{2} \quad \text{is } \Theta(n^2)$$

Is there any best can or worst case?
Are they all the same?

Let's look @ Insertion Sort:

```
                Ins.Sort(list, n)
c1  n     1.  for i ← 2 to n
c2  (n-1) 2.      key ← list[i]
c3  (n-1) 3.      j ← i-1
c4        4.      while j > 0 AND list[j] > key
c5        5.          list[j+1] ← list[j]
c6        6.          j ← j-1
c7  (n-1) 7.      list[j+1] ← key
```

Insert key into appropriate place w/in list[1..i-1]

Any best case or worst case you can see?

best case it does the body of this loop 0 times

worst " it " " " " n " n

when i=2 1

i=3 2

i=4 3

i=n n-1

note also that the line 4 goes 1 + those times:

(i.e. every 1st body of loop (3 & 6) goes 0 times condition is tested once in line 4.

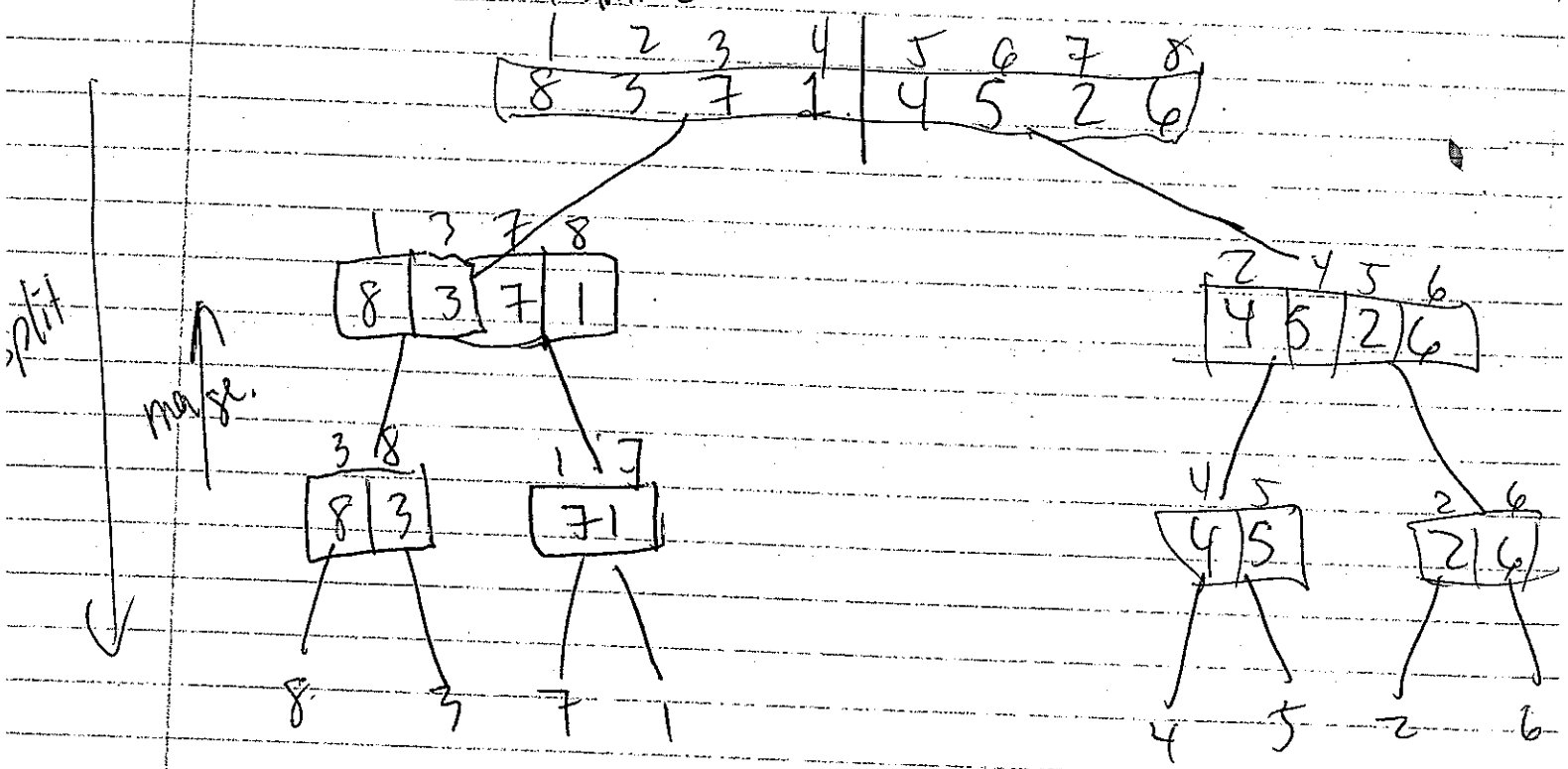
So Best case is $\Theta(n)$

Worst " $\Theta(n^2)$

So. one can say for overall. Insertion sort $O(n^2)$ AND $\Omega(n)$

		Best Case	Worst Case	Overall	Space
in place, small constants	Selection Sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(1)$
	Insertion Sort	$\Theta(n)$	$\Theta(n^2)$	$O(n^2)$ $\Omega(n)$	$\Theta(1)$
larger constants, NOT in place.	Merge Sort	$\Theta(n \lg n)$	$\Theta(n \lg n)$	$\Theta(n \lg n)$	$\Theta(n)$

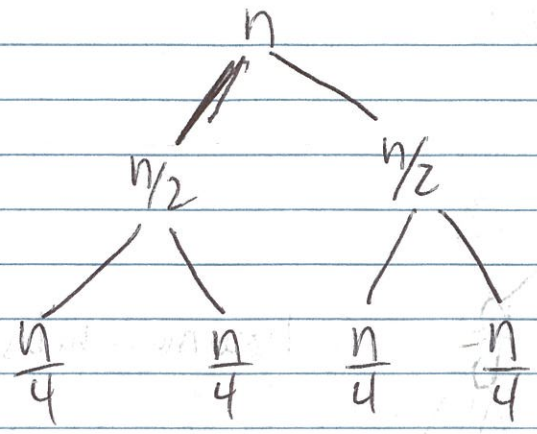
MERGESORT EXAMPLE



takes $\Theta(n)$ extra space

do Merge Sort Rec. Tree next

Merge Sort Recursion tree split in 1/2 each time



1 1 1

# of leaves? n
Each level has how many elements it works w/? n

How many levels? $\log_2 n + 1$

Note $\log_2 n$ is the # of times I can cut n in half until I reach 1.

$\log_2 n$ is also the power of 2 that gives n.