Lecture 9
Contents covered in class

1. Divide and conquer Algorithms
Quick Sort

[1 : n]

```
QUICKSORT (A, l, h)
{
      if l >= h
            return
      p = PARTITION (A, l, h)
      QUICKSORT (A, l, p-1)
      QUICKSORT (A, p+1, h)
}
```

position of pivot

left

right

O(nlgn)

Quick Sort Partition Algorithm 1:

```
// selects last element as pivot
PARTITION (A, l, h)
{
      p = A[h]
      i = l-1
      for j = l to h-1
      {     if A[j] <= p
                  i = i+1
                  swap (A[i], A[j])
      }
      swap (A[i+1], A[h])
      return i+1
}
```

(last element)

~~13~~	~~19~~	~~9~~	~~5~~	~~12~~	~~8~~	~~7~~	~~4~~	21	11

9 5 13 19 4 13 19 12 p

8 7

9 5 8 7 4 (13) 19 12 21 (11)

Small

[9, 5, 8, 7, 4] [11] [19, 12, 21, 13]

O(n)

Quick Sort Partition Algorithm 2:

```
// selects last element as pivot
PARTITION (A, l, h)
{
      p = A[h]
      i = l
      j = h-1
      while i <= j
      {     while i <= j and A[i] <= p
                  i = i+1
            while i <= j and A[j] >= p
                  j = j-1
            if i < j
                  swap (A[i], A[j])
      }
      swap (A[i], A[h])
      return i
}
```

4 7 8 19 19 13 12

~~13~~	~~19~~	9	5	~~12~~	~~8~~	~~7~~	~~4~~	21	11

i+1

4 7 9 5 8 (11) 19 13 21 12

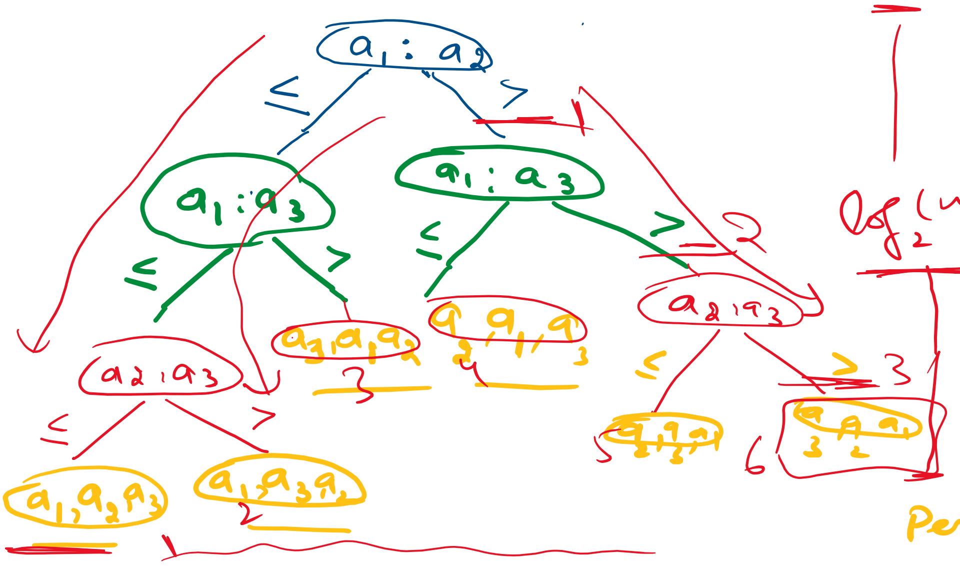
O(n)

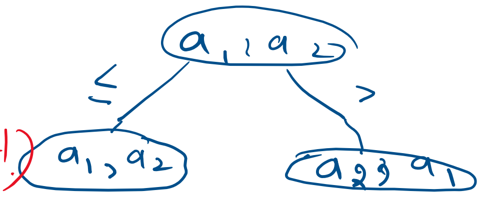
2. A Lower Bound for Sorting: Comparison based

Proof that the lower bound for sorting n numbers is Ω(nlgn)
Create the comparison tree.

$n = 3$ (a_1, a_2, a_3)

2 options $a_1 \le a_2$, $a_1 > a_2$





2 elements 1 comparison.

2^h

$3 = 3 \cdot 2 \cdot 1 = 3!$
$4 = 4 \cdot 3 \cdot 2 \cdot 1 = 4!$

Permutation of three numbers. $n = n!$ leaf

123 132 312 213 231 321

for any number $n \rightarrow$ permutations

$n = 3 \rightarrow 2^3 \rightarrow 8$ $\log_2(2^n)$

n $2^n \rightarrow$ permutation maximum $h = n$

height = $n!$ leaf nodes.

$\log_2(n!)$

$2^h \ge n!$

$h \ge \log_2(n!)$

height $\rightarrow \simeq (n\log n)$

$\Omega(n\log n)$

at least

Quick —
merge — $\Omega(n\lg n)$
heap —

3. Stable vs unstable sorting:

1	2	3	4	5	6	7	8	9	10
13	19	9	5	12	5	7	9	21	11
		a	a		b		b		

Keeps order of same keys as provided in input.

→ 5a, 5b, 7, 9a, 9b, 11, 12, 13, 19, 21

Search

→ insertion Sort $O(n^2)$
→ Bubble sort $O(n^2)$
→ Selection Sort $O(n^2)$

Stable $L[i] \le R[j]$ $A[k] = L[i]$
→ Merge Sort $O(n \lg n)$
→ Quick Sort $O(n \lg n)$
→ heap Sort $O(n \lg n)$
unstable

4. Sorting in Linear Time O(n): Count Sort uses counting for sorting.

2	8	1	4	1	2	6	8

→ non-comparison Based algo