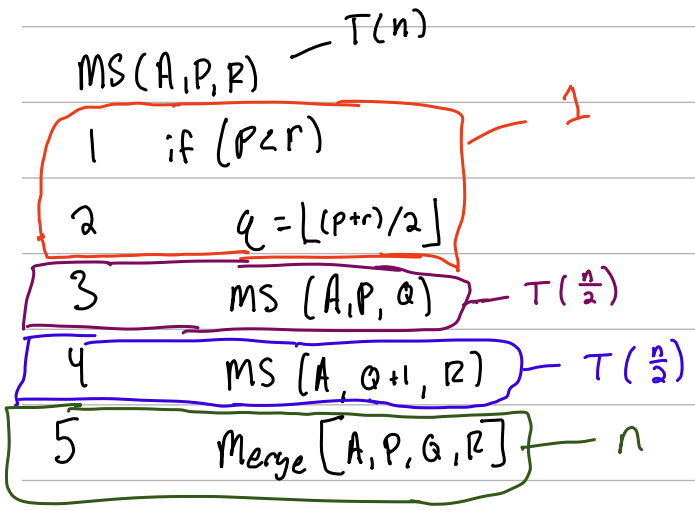
# Lecture 16

## Recursive algorithms & Divide and Conquer

1. Divide
2. Conquer the smaller "subproblem"
3. Combine

## Analyzing Time Complexity Given pseudocode

## Merge Sort




So $T(n)$ of this is $T(n) = 2T(\frac{n}{2}) + n + 1$

# Binary Search

BinarySearch(A, v, low, mid)

```
1  if low ≤ high
2      mid = ⌊(low + high)/2⌋
3      if v == A[mid]
4          return mid
5      else if v < A[mid]
6          return BinarySearch(A, v, low, mid-1)
7      else return BinarySearch(A, v, mid+1, high)
8  return NIL
```

Line 1-5 : 1

Line 6 : $T(\frac{n}{2})$

Line 7 : $T(\frac{n}{2})$

8 : 1

$T(n) = 1 + T(\frac{n}{2}) + 1$ ← only $T(\frac{n}{2})$ and not $2T(\frac{n}{2})$ because line 6 or 7 runs, not both.

$= T(\frac{n}{2}) + 1$

- Always Assume that the function was initially called on an input size of n.
- $T(\frac{n}{2} - 1)$ can be simplified to $T(\frac{n}{2})$
- $T(n-1)$ cannot be simplified, it stays $T(n-1)$