PART 2 SUMMARY – read and memorize!!!

Analyze algorithms (not programs) for the best case scenario and the worst case scenario given N input items.

Count the basic essential operation (time complexity)
See if it makes copies of the data (space complexity)

For time complexity, we are only interested in the growth rate as N gets huge.

F(N) is the best possible any algorithm can do for the worst case scenario.

MEMORIZE THESE AND BE ABLE TO EXPLAIN WHY!!!!

B(N) is the number of operations given N as the input size in the best case scenario
W(N) is the number of operations given N as the input size in the worst case scenario

Sequential search:
B(N) = 1 comparison (you find it right away)
W(N) = N comparisons (you don't find it til the end or not at all)
Reversing of a string in place:
B(N) = N/2 swaps
W(N) = N/2 swaps (there is no best or the worst string)
Finding the largest number:
B(N) = N-1 comparisons
W(N) = N-1 comparisons (N-1 always must lose)
Check to see if it is a palindrome:
B(N) = 1 comparison (find out right away that it is not a palindrome)
W(N) = N/2 comparisons (when it IS a palindrome)

A(N) is difficult to find because you have to know the probability of each case

O, Theta, Omega are used because all we care about is the growth rate as N increases

Summation formula	Sum k as k varies from 1 to N = N(N+1)/2
Pairs formula	There are N(N-1)/2 pairs
Choices formula	There are C^N possibilities

If you cut in ½, can do it LOG N times
If you go down a triangular binary tree, can do it LOG N times

The perfectly triangular (shortest) binary tree:
B is the bottom level   N = 2^(B+1) -1   N = #nodes
B+1 = log(N+1) -1 i.e. how many levels

F(N) of searching a unsorted list is O(N)
F(N) of searching a sorted list is O(logN)

Binary Search Algorithm is thus optimal

Keep on adding to this sheet

Log N and N and NlogN are acceptable even for large N