

Asymptotic Dominance in Action								
	O(lg n)	O(n)	O(n lg n)	n²	2 <sup>n</sup>	n!		
10	0.003 µs	0.01 µs	0.033 µs	0.1 µs	1 µs	3.63 ms		
20	0.004 µs	0.02 µs	0.086 µs	0.4 µs	1 ms	77.1 years		
30	0.005 µs	0.03 µs	0.147 µs	0.9 µs	1 sec	8.4*1015 yrs		
10	0.005 µs	0.04 µs	0.213 µs	1.6 µs	18.3 min			
50	0.006 µs	0.05 µs	0.282 µs	2.5 µs	13 days			
100	0.007 µs	0.1 µs	0.644 µs	10 µs	4*1013 yrs			
1,000	0.010 µs	1.00 µs	9.966 µs	1 ms				
10,000	0.013 µs	10 µs	130 µs	100 ms				
100,000	0.017 µs	0.10 ms	1.67 ms	10 sec				
1,000,000	0.020 µs	1 ms	19.93 ms	16.7 min				
10,000,000	0.023 µs	0.01 sec	0.23 sec	1.16 days				
100,000,000	0.027 µs	0.10 sec	2.66 sec	115.7				
1,000,000,000	0.030 µs	1 sec	29.90 sec	3.7 years				
2/9/09 CS380 Algorithm Design and Analysis 2								



# Motivation

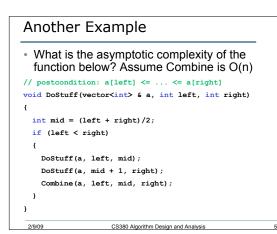
```
• The following structure and function exist:
struct Tree
{
    int info;
    Tree * left;
    Tree * right;
    Tree(int value, Tree * lchild, Tree * rchild) :
        info(value), left(lchild), right(rchild) { }
};
// return true if & only if all values in t are less than val
bool ValsLess(Tree * t, int val)
2/909 CS380 Algorithm Design and Analysis 3
```

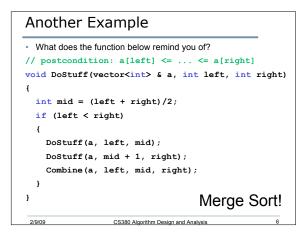


2/9/09

```
// returns true if t represents a binary
// search tree containing no duplicate values;
bool IsBST(Tree * t)
{
    if (t == NULL) return true;
    return ValsLess(t->left, t->info) &&
        ValsGreater(t->right, t->info) &&
        IsBST(t->left) &&
        IsBST(t->right);
}
* What is the complexity or running time of the
    above function?
```

CS380 Algorithm Design and Analysis

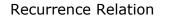




Merge Sort					
What was the running time of the Merge procedure in Merge Sort?	$\begin{array}{l} \underbrace{MERGE\left(A, \ p, \ q, \ r\right)}_{n_{i} \leftarrow q - p + 1} \\ n_{2} \leftarrow r - q \\ \hline \mathtt{create} \ \mathtt{arrays} \ L[1n_{i} + 1] \ \mathtt{and} \ R[1n_{2} + 1] \\ \hline \mathtt{for} \ i \leftarrow 1 \ \mathtt{to} \ n_{1} \\ \hline \mathtt{do} \ L[i] \leftarrow A[p + i - 1] \\ \hline \mathtt{for} \ j \leftarrow 1 \ \mathtt{to} \ n_{2} \\ \hline \mathtt{do} \ R[j] \leftarrow A[q + j] \\ L[n_{i} + 1] \leftarrow \infty \\ i \leftarrow 1 \\ j \leftarrow 1 \\ \hline \mathtt{for} \ t \leftarrow p \ \mathtt{to} \ r \\ \mathbf{do} \ \mathtt{if} \ L[i] \leq R[j] \\ \mathbf{then} \ A[k] \leftarrow L[i] \\ i \leftarrow i + 1 \end{array}$				
O(n)	else $A[k] \leftarrow R[j]$ $j \leftarrow j+1$				
2/9/09 CS380 Algorithm Design and Analysis 7					



# Merge Sort MERGE-SORT (A, p, r) $\nabla$ p & r are indices into the array (p < r)</td> if p < r</td> $\nabla$ Check for base case then q $\leftarrow$ [(p + r) / 2] $\nabla$ Divide MERGE-SORT (A, p, q) $\nabla$ Conquer MERGE-SORT (A, q + 1, r) $\nabla$ Combine



2/9/09

• Let T(n) be the time for Merge-Sort to execute on an n element array. The time to execute on a one element array is O(1)

CS380 Algorithm Design and Analysis

9

· Then we have the following relationship

### Best Case

2/9/09

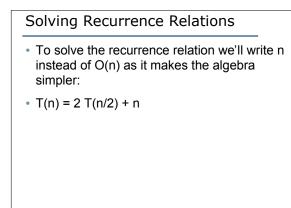
2/9/09

- A *recurrence relation* contains two equations
  - One for the general case
  - One for the base case
- How does this relate to the time for IsBST to execute?

CS380 Algorithm Design and Analysis

10

11



CS380 Algorithm Design and Analysis

Recuri	Recurrence Relations to Remember						
T(n) = T(n)	/2) + 0(1)						
T(n) = T(n-	-1) + 0(1)						
T(n) = 2 T	(n/2) + O(1)						
T(n) = T(n)	-1) + O(n)						
T(n) = 2 T	(n/2) + 0(n)						
2/9/09	2/9/09 CS380 Algorithm Design and Analysis 12						

## Your Turn

• Solve the following recurrence relation using the expansion (iteration) method

• T(n) = T(n-1) + 2n -1

# Approaches to Algorithm Design

Incremental

2/9/09

Job is partly done – do a little more, repeat until done.

CS380 Algorithm Design and Analysis

13

14

15

- Divide-and-Conquer (recursive)
  - Divide problem into sub-problems of the same kind.
  - For small subproblems, solve, else, solve them recursively.

CS380 Algorithm Design and Analysis

CS380 Algorithm Design and Analysis

Combine subproblem solutions to solve the whole thing.

### Problems

2/9/09

2/9/09

- Problem 2-1
- Problem 2-2

### For Next Time

2/9/09

• So far we've covered chapters 1, 2, and 3.

CS380 Algorithm Design and Analysis

16

 We'll skip 4 and 5 for now, and start on chapter 6 next time.