

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	
40	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 days			
1,000,000,000	0.030 µs	1 sec	29.90 sec	3.7 years			
2/6/11	2/6/11 CS380 Algorithm Design and Analysis 2						





# Divide and Conquer Algorithms

• Analysis of divide and conquer algorithms requires knowledge of:

CS380 Algorithm Design and Analysis

- Mathematical Induction
- Substitution/Iterative Method
- o Recurrences

2/6/11

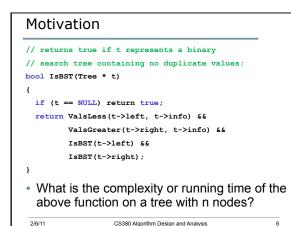
# Motivation • The following structure and function exist: struct Tree { int info; Tree \* left; Tree \* right;

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)

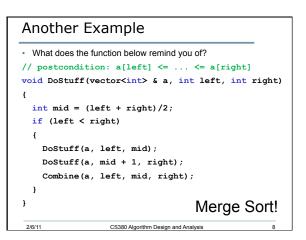
2/6/11 CS380 Algorithm Design and Analysis

Thank you Owen Astrachan

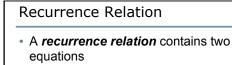


```
Another Example
• What is the asymptotic complexity of the
  function below? Assume Combine is O(n)
// postcondition: a[left] <= ... <= a[right]</pre>
void DoStuff(vector<int> & a, int left, int right)
{
 int mid = (left + right)/2;
 if (left < right)</pre>
 ł
   DoStuff(a, left, mid);
   DoStuff(a, mid + 1, right);
   Combine(a, left, mid, right);
 }
}
                  CS380 Algorithm Design and Analysis
```

2/6/11



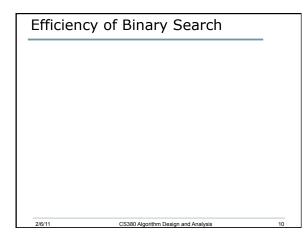
CS380 Algorithm Design and Analysis

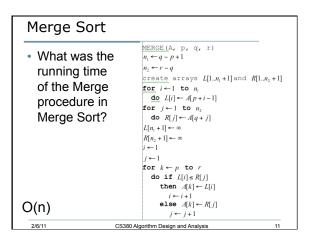


One for the general case

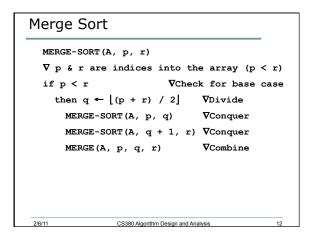
o One for the base case

2/6/11











### **Recurrence Relation**

- 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)
- Then we have the following relationship:

# Merge Sort

2/6/11

• To solve the recurrence relation we'll write n instead of O(n) as it makes the algebra simpler:

CS380 Algorithm Design and Analysis

13

14

• T(n) = 2 T(n/2) + n

- o T(1) = 1
- Solve the recurrence by iteration (substitution)

2/6/11 CS380 Algorithm Design and Analysis

• Use induction to prove the solution is correct

Recurrence Relations to Remember					
T(n) = T(n/2) + O(n/2)	(1)				
T(n) = T(n-1) + O(	(1)				
T(n) = 2 T(n/2) +	0(1)				
T(n) = T(n-1) + O(n-1)	(n)				
T(n) = 2 T(n/2) +	0 (n)				
2/6/11	CS380 Algorithm Design and Analysis	15			



### Your Turn

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

∘ T(n) = T(n-1) + 2n -1

• T(0) = 0

# Approaches to Algorithm Design

Incremental

2/6/11

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

CS380 Algorithm Design and Analysis

16

17

18

- 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

Combine subproblem solutions to solve the whole thing.

# For Next Time

2/6/11

2/6/11

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

CS380 Algorithm Design and Analysis