Recurrence Relations – Running Time for Recursive Functions

Chapter 2

Asymptotic Dominance in Action

	O(lg n)	O(n)	O(n lg n)	n^2	2^n	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	$4*10^{13}$ 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		

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)
```

Motivation

```
// 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?

Another Example

- What is the asymptotic complexity of the function below? Assume Combine is O(n)

```
// postcondition: a[left] <= ... <= a[right]
void DoStuff(vector<int> & a, int left, int right)
{
  int mid = (left + right)/2;
  if (left < right)
  {
    DoStuff(a, left, mid);
    DoStuff(a, mid + 1, right);
    Combine(a, left, mid, right);
  }
}
```


Another Example

- What does the function below remind you of?

```
// postcondition: a[left] <= ... <= a[right]
void DoStuff(vector<int> & a, int left, int right)
{
  int mid = (left + right)/2;
  if (left < right)
  {
    DoStuff(a, left, mid);
    DoStuff(a, mid + 1, right);
    Combine(a, left, mid, right);
  }
}
```

Merge Sort!

Merge Sort

- What was the running time of the Merge procedure in Merge Sort?

```
MERGE(A, p, q, r)
n1 ← q − p + 1
n2 ← r − q
create arrays L[1..n1 + 1] and R[1..n2 + 1]
for i ← 1 to n1
  do L[i] ← A[p + i − 1]
for j ← 1 to n2
  do R[j] ← A[q + j]
L[n1 + 1] ← ∞
R[n2 + 1] ← ∞
i ← 1
j ← 1
for k ← p to r
  do if L[i] ≤ R[j]
     then A[k] ← L[i]
          i ← i + 1
     else A[k] ← R[j]
          j ← j + 1
```

O(n)

Merge Sort

```
MERGE-SORT(A, p, r)
∇ p & r are indices into the array (p < r)
if p < r                    ∇Check for base case
  then q ← ⌊(p + r) / 2⌋      ∇Divide
    MERGE-SORT(A, p, q)       ∇Conquer
    MERGE-SORT(A, q + 1, r)   ∇Conquer
    MERGE(A, p, q, r)         ∇Combine
```


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

Best Case

- 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?

Solving Recurrence Relations

- To solve the recurrence relation we'll write n instead of O(n) as it makes the algebra simpler:
- T(n) = 2 T(n/2) + n

Recurrence Relations to Remember

`T(n) = T(n/2) + O(1)`		
`T(n) = T(n-1) + O(1)`		
`T(n) = 2 T(n/2) + O(1)`		
`T(n) = T(n-1) + O(n)`		
`T(n) = 2 T(n/2) + O(n)`		

Your Turn

- Solve the following recurrence relation using the expansion (iteration) method
- $T(n) = T(n-1) + 2n - 1$

Approaches to Algorithm Design

- Incremental
  - Job is partly done – do a little more, repeat until done.
- Divide-and-Conquer (recursive)
  - Divide problem into sub-problems of the same kind.
  - For small subproblems, solve, else, solve them recursively.
  - Combine subproblem solutions to solve the whole thing.

Problems

- Problem 2-1
- Problem 2-2

For Next Time

- So far we've covered chapters 1, 2, and 3.
- We'll skip 4 and 5 for now, and start on chapter 6 next time.