Algorithms

Merge Sort Solving Recurrences The Master Theorem

Administrative Question

- Who here cannot make Monday-Wednesday office hours at 10 AM?
- If nobody, should we change class time?

Homework 1

- Homework 1 will be posted later today
 - (Problem with the exercise numbering, sorry)
 - Due Monday Jan 28 at 9 AM
 - Should be a fairly simple warm-up problem set

Review: Asymptotic Notation

• Upper Bound Notation:

- f(n) is O(g(n)) if there exist positive constants cand n_0 such that $f(n) \le c \cdot g(n)$ for all $n \ge n_0$
- Formally, $O(g(n)) = \{ f(n): \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } f(n) \le c \cdot g(n) \forall n \ge n_0 \}$

• Big O fact:

• A polynomial of degree k is $O(n^k)$

Review: Asymptotic Notation

- Asymptotic lower bound:
 - f(n) is $\Omega(g(n))$ if \exists positive constants *c* and n_0 such that $0 \le c \cdot g(n) \le f(n) \forall n \ge n_0$
- Asymptotic tight bound:
 - f(n) is $\Theta(g(n))$ if \exists positive constants c_1, c_2 , and n_0 such that $c_1 g(n) \le f(n) \le c_2 g(n) \forall n \ge n_0$
 - $f(n) = \Theta(g(n))$ if and only if f(n) = O(g(n)) AND $f(n) = \Omega(g(n))$

Other Asymptotic Notations

- A function f(n) is o(g(n)) if \exists positive constants c and n_0 such that $f(n) < c g(n) \forall n \ge n_0$
- A function f(n) is $\omega(g(n))$ if \exists positive constants *c* and n_0 such that $c g(n) < f(n) \forall n \ge n_0$
- Intuitively,
 - o() is like <</p>
 - O() is like ≤
- ω () is like >
 - $\Omega()$ is like \geq
- Θ () is like =

Merge Sort

```
MergeSort(A, left, right) {
    if (left < right) {
        mid = floor((left + right) / 2);
        MergeSort(A, left, mid);
        MergeSort(A, mid+1, right);
        Merge(A, left, mid, right);
    }
}</pre>
```

// Merge() takes two sorted subarrays of A and
// merges them into a single sorted subarray of A
// (how long should this take?)

Merge Sort: Example

• Show MergeSort() running on the array

$$A = \{10, 5, 7, 6, 1, 4, 8, 3, 2, 9\};$$

Analysis of Merge Sort

| Statement | Effort |
|---------------------------------------------|--------------|
| MergeSort(A, left, right) { | T(n) |
| if (left < right) { | Θ (1) |
| <pre>mid = floor((left + right) / 2);</pre> | Θ (1) |
| MergeSort(A, left, mid); | T(n/2) |
| <pre>MergeSort(A, mid+1, right);</pre> | T(n/2) |
| <pre>Merge(A, left, mid, right);</pre> | Θ (n) |
| } | |
| • So $T(n) = \Theta(1)$ when $n = 1$, and | |
| | |

 $2T(n/2) + \Theta(n)$ when n > 1

• So what (more succinctly) is T(n)?

Recurrences

• The expression:

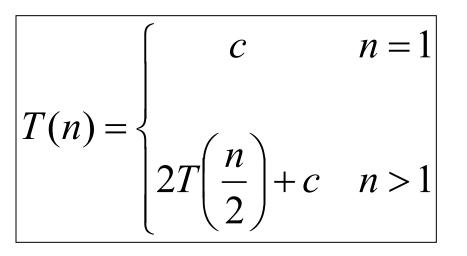
$$T(n) = \begin{cases} c & n = 1\\ 2T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

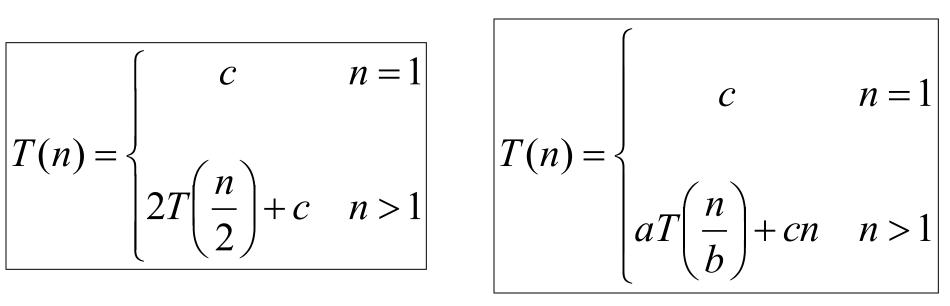
- is a recurrence.
 - Recurrence: an equation that describes a function in terms of its value on smaller functions

Recurrence Examples

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

$$s(n) = \begin{cases} 0 & n = 0\\ n + s(n-1) & n > 0 \end{cases}$$





- Substitution method
- Iteration method
- Master method

- The substitution method (CLR 4.1)
 - A.k.a. the "making a good guess method"
 - Guess the form of the answer, then use induction to find the constants and show that solution works
 - Examples:
 - T(n) = 2T(n/2) + Θ(n) → T(n) = Θ(n lg n)
 T(n) = 2T(⌊n/2⌋) + n → ???

- The substitution method (CLR 4.1)
 - A.k.a. the "making a good guess method"
 - Guess the form of the answer, then use induction to find the constants and show that solution works
 - Examples:
 - T(n) = 2T(n/2) + Θ(n) → T(n) = Θ(n lg n)
 T(n) = 2T(⌊n/2⌋) + n → T(n) = Θ(n lg n)
 - $T(n) = 2T(\lfloor n/2 \rfloor) + 17) + n \rightarrow ???$

- The substitution method (CLR 4.1)
 - A.k.a. the "making a good guess method"
 - Guess the form of the answer, then use induction to find the constants and show that solution works
 - Examples:
 - T(n) = 2T(n/2) + Θ(n) → T(n) = Θ(n lg n)
 T(n) = 2T(⌊n/2⌋) + n → T(n) = Θ(n lg n)
 T(n) = 2T(⌊n/2⌋+17) + n → Θ(n lg n)

- Another option is what the book calls the "iteration method"
 - Expand the recurrence
 - Work some algebra to express as a summation
 - Evaluate the summation
- We will show several examples

$$s(n) = \begin{cases} 0 & n = 0\\ c + s(n-1) & n > 0 \end{cases}$$

• s(n) =c + s(n-1)c + c + s(n-2)2c + s(n-2)2c + c + s(n-3)3c + s(n-3). . . kc + s(n-k) = ck + s(n-k)

$$s(n) = \begin{cases} 0 & n = 0\\ c + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have
 - s(n) = ck + s(n-k)
- What if k = n?

$$\bullet s(n) = cn + s(0) = cn$$

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have
 - s(n) = ck + s(n-k)
- What if k = n?

•
$$s(n) = cn + s(0) = cn$$

• So

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

• Thus in general

•
$$s(n) = cn$$

$$s(n) = \begin{cases} 0 & n = 0\\ n + s(n-1) & n > 0 \end{cases}$$

- = n + s(n-1)
- = n + n 1 + s(n 2)
- = n + n 1 + n 2 + s(n 3)
- = n + n 1 + n 2 + n 3 + s(n 4)
- = ...
- = n + n 1 + n 2 + n 3 + ... + n (k 1) + s(n k)

$$s(n) = \begin{cases} 0 & n = 0\\ n + s(n-1) & n > 0 \end{cases}$$

= ...

- = n + s(n-1)
- = n + n 1 + s(n 2)
- = n + n 1 + n 2 + s(n 3)
- = n + n 1 + n 2 + n 3 + s(n 4)
- = n + n 1 + n 2 + n 3 + ... + n (k 1) + s(n k)= $\sum_{i=n-k+1}^{n} i + s(n-k)$

$$s(n) = \begin{cases} 0 & n = 0\\ n + s(n-1) & n > 0 \end{cases}$$

$$\sum_{i=n-k+1}^{n} i + s(n-k)$$

$$s(n) = \begin{cases} 0 & n = 0\\ n + s(n-1) & n > 0 \end{cases}$$

$$\sum_{i=n-k+1}^{n} i + s(n-k)$$

• What if k = n?

$$s(n) = \begin{cases} 0 & n = 0\\ n + s(n-1) & n > 0 \end{cases}$$

$$\sum_{i=n-k+1}^{n} i + s(n-k)$$

• What if k = n?

$$\sum_{i=1}^{n} i + s(0) = \sum_{i=1}^{n} i + 0 = n \frac{n+1}{2}$$

$$s(n) = \begin{cases} 0 & n = 0\\ n + s(n-1) & n > 0 \end{cases}$$

$$\sum_{i=n-k+1}^{n} i + s(n-k)$$

• What if k = n?

$$\sum_{i=1}^{n} i + s(0) = \sum_{i=1}^{n} i + 0 = n \frac{n+1}{2}$$

• Thus in general $s(n) = n \frac{n+1}{2}$

$$T(n) = \begin{cases} c & n = 1\\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

• T(n) =2T(n/2) + c2(2T(n/2/2) + c) + c $2^{2}T(n/2^{2}) + 2c + c$ $2^{2}(2T(n/2^{2}/2) + c) + 3c$ $2^{3}T(n/2^{3}) + 4c + 3c$ $2^{3}T(n/2^{3}) + 7c$ $2^{3}(2T(n/2^{3}/2) + c) + 7c$ $2^{4}T(n/2^{4}) + 15c$

 $2^{k}T(n/2^{k}) + (2^{k} - 1)c$

$$T(n) = \begin{cases} c & n = 1\\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

•
$$T(n) = 2^k T(n/2^k) + (2^k - 1)c$$

• What if
$$k = \lg n$$
?

•
$$T(n) = 2^{\lg n} T(n/2^{\lg n}) + (2^{\lg n} - 1)c$$

= $n T(n/n) + (n - 1)c$
= $n T(1) + (n-1)c$
= $nc + (n-1)c = (2n - 1)c$