Introduction to Algorithms

Merge Sort

Solving Recurrences

The Master Theorem

Review: Asymptotic Notation

- Upper Bound Notation:
  - f(n) is O(g(n)) if there exist positive constants c and n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$
  - Formally, $O(g(n)) = \{ f(n)\!: \exists$ positive constants c and n_0 such that $f(n) \leq c \cdot g(n)\ \forall\ n \geq n_0$
- 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 $0 \le c \cdot g(n) \le f(n) \quad \forall\ n \ge n_0$
- 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

  $$f(n) < c\, g(n) \;\forall\; n \geq n_0$$

- A function f(n) is ω(g(n)) if $\exists$ positive constants c and n_0 such that

  $$c\, g(n) < f(n) \;\forall\; n \geq n_0$$

- Intuitively,
  - **o() is like <**
  - **ω() is like >**
  - **Θ() is like =**
  - **O() is like ≤**
  - **Ω() 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);
   }
}

// 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) {`	$\Theta(1)$
`mid = floor((left + right) / 2);`	$\Theta(1)$
`MergeSort(A, left, mid);`	$T(n/2)$
`MergeSort(A, mid+1, right);`	$T(n/2)$
`Merge(A, left, mid, right);`	$\Theta(n)$
`}`	
`}`	

- So $T(n) = \begin{cases} \Theta(1) & \text{when } n = 1 \text{, and} \\ 2T(n/2) + \Theta(n) & \text{when } n > 1 \end{cases}$
- So what (more succinctly) is $T(n)$?

Recurrences

- The expression:

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

is a *recurrence*.

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

Recurrence Examples

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

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

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

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

Solving Recurrences

- Substitution method
- Iteration method
- Master method

Solving Recurrences

- 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:
    - $T(n) = 2T(n/2) + \Theta(n) \rightarrow T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor) + n \rightarrow$???

Solving Recurrences

- 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:
    - $T(n) = 2T(n/2) + \Theta(n) \rightarrow \quad T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor) + n \rightarrow T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor)+ 17) + n \rightarrow$???

Solving Recurrences

- 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:
    - $T(n) = 2T(n/2) + \Theta(n) \rightarrow T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor) + n \rightarrow T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n \rightarrow \Theta(n \lg n)$

Solving Recurrences

- 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?
  - 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

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

- s(n)

= 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)

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

- s(n)

= 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)

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

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

- So far for n >= k we have

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

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

- So far for n >= k we have

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

- What if k = n?

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

- So far for n >= k we have

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

- What if k = n?

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

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

- So far for n >= k we have

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

- What if k = n?

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

- Thus in general

$$s(n) \quad = \quad n\frac{n+1}{2}$$

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

- T(n) =

  2T(n/2) + c

  2(2T(n/2/2) + c) + c

  $2^2T(n/2^2) + 2c + c$

  $2^2(2T(n/2^2/2) + c) + 3c$

  $2^3T(n/2^3) + 4c + 3c$

  $2^3T(n/2^3) + 7c$

  $2^3(2T(n/2^3/2) + c) + 7c$

  $2^4T(n/2^4) + 15c$

  …

  $2^kT(n/2^k) + (2^k - 1)c$

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

- So far for n > 2k we have
  - $T(n) = 2^k T(n/2^k) + (2^k - 1)c$
- What if k = lg n?
  - $T(n) = 2^{\lg n} T(n/2^{\lg n}) + (2^{\lg n} - 1)c$

    $= n\,T(n/n) + (n - 1)c$

    $= n\,T(1) + (n-1)c$

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