Solving recurrences Master Theorem.

Slide credit: David Luebke (Virginia)

Review: 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.
// Code for this is in the book.  It requires O(n)
// time, and *does* require allocating O(n) space
```

Review: Analysis of Merge Sort

Statement	Effort

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

- So T(n) = $\Theta(1)$ when n = 1, and

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

- This expression is a *recurrence*

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

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

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

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

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

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

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

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

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

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

- T(n) =

  2T(n/2) + c

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

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

  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

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

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

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

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

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

Generalize previous result

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

- More general recurrence than last one
- Previous example: a=b=2.
- New feature: *result depends on whether a<b, a=b or a>b !*

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

- T(n) =

  aT(n/b) + cn

  a(aT(n/b/b) + cn/b) + cn

  $a^2T(n/b^2) + cna/b + cn$

  $a^2T(n/b^2) + cn(a/b + 1)$

  $a^2(aT(n/b^2/b) + cn/b^2) + cn(a/b + 1)$

  $a^3T(n/b^3) + cn(a^2/b^2) + cn(a/b + 1)$

  $a^3T(n/b^3) + cn(a^2/b^2 + a/b + 1)$

  …

  $a^kT(n/b^k) + cn(a^{k-1}/b^{k-1} + a^{k-2}/b^{k-2} + \ldots + a^2/b^2 + a/b + 1)$

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

- So we have
  - $T(n) = a^kT(n/b^k) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
- For $k = \log_b n$
  - $n = b^k$
  - $T(n) = a^kT(1) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= a^kc + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= ca^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= cna^k/b^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if a = b?
  - $T(n) = cn(k + 1)$

    $= cn(\log_b n + 1)$

    $= \Theta(n \log n)$

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

- So with k = $\log_b$ n
  - T(n) = cn(a^k/b^k + ... + a^2/b^2 + a/b + 1)
- What if a < b?

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$
  - So:

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$
  - So:

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

  - $T(n) = cn \cdot \Theta(1) = \Theta(n)$

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

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

- So with k = $\log_b$ n
  - T(n) = cn(a^k/b^k + ... + a^2/b^2 + a/b + 1)
- What if a > b?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

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

- So with k = $\log_b$ n
  - T(n) = cn(a^k/b^k + ... + a^2/b^2 + a/b + 1)
- What if a > b?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - T(n) = cn · Θ(a^k / b^k)

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$$= cn \cdot \Theta(a^{\log n} / b^{\log n}) = cn \cdot \Theta(a^{\log n} / n)$$

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$= cn \cdot \Theta(a^{\log n} / b^{\log n}) = cn \cdot \Theta(a^{\log n} / n)$

recall logarithm fact: $a^{\log n} = n^{\log a}$

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

  $= cn \cdot \Theta(a^{\log n} / b^{\log n}) = cn \cdot \Theta(a^{\log n} / n)$

  recall logarithm fact: $a^{\log n} = n^{\log a}$

  $= cn \cdot \Theta(n^{\log a} / n) = \Theta(cn \cdot n^{\log a} / n)$

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

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

  $= cn \cdot \Theta(a^{\log n} / b^{\log n}) = cn \cdot \Theta(a^{\log n} / n)$

  recall logarithm fact: $a^{\log n} = n^{\log a}$

  $= cn \cdot \Theta(n^{\log a} / n) = \Theta(cn \cdot n^{\log a} / n)$

  $= \Theta(n^{\log a})$

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

- So…

$$T(n) = \begin{cases} \Theta(n) & a < b \\ \Theta(n \log_b n) & a = b \\ \Theta\left(n^{\log_b a}\right) & a > b \end{cases}$$

The Master Theorem

- Given: a *divide and conquer* algorithm
  - An algorithm that divides the problem of size *n* into *a* subproblems, each of size *n*/*b*
  - Let the cost of each stage (i.e., the work to divide the problem + combine solved subproblems) be described by the function *f*(n)
- Then, the Master Theorem gives us a cookbook for the algorithm's running time:

The Master Theorem

- if $T(n) = aT(n/b) + f(n)$ then

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} \log n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \text{ AND} \\ & af(n/b) < cf(n) \text{ for large } n \end{cases} \quad \begin{array}{l} \varepsilon > 0 \\ c < 1 \end{array}$$

Using The Master Method

- $T(n) = 9T(n/3) + n$
  - $a=9$, $b=3$, $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
  - Since $f(n) = O(n^{\log_3 9 - \varepsilon})$, where $\varepsilon=1$, case 1 applies:

$$T(n) = \Theta\left(n^{\log_b a}\right) \text{ when } f(n) = O\left(n^{\log_b a - \varepsilon}\right)$$

  - Thus the solution is $T(n) = \Theta(n^2)$

Sorting revisited

- We have seen algorithms for sorting: INSERTION-SORT, MERGESORT
- More generally:
- We are given a sequence of items
- Each item has a characteristic called sorting key. The values of the sorting key belong to a set on which there exists a total order relationship
- Sorting the sequence = arrange its elements such that the sorting keys are in increasing (or decreasing) order

Sorting revisited

Other assumptions:

- We shall consider that the sequence is stored in a random access memory (e.g. in an array)
- This means that we will discuss about internal sorting
- We shall analyze only sorting methods which are in place (the additional space needed for sorting has at most the size of an element/few elements.
- Stability: preserves ordering of elements with identical keys

Stability

Example:

- Initial configuration:

  ((Adam,9), (John, 10), (Peter,9), (Victor,8))

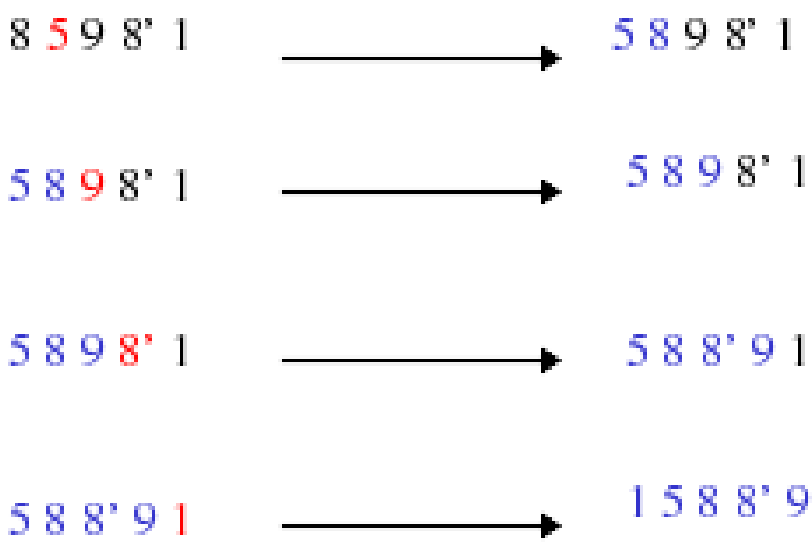
- Stable sorting :

  ((John,10),(Adam,9),(Peter,9),(Victor,8))

- Unstable sorting :

  ((John,10), (Peter,9),(Adam,9), (Victor,8))

Insertion sort – stability



The insertion method is stable

Coming up: Sorting methods

- (done) INSERTION-SORT, MERGESORT
- SELECTION-SORT, QUICKSORT
- HEAPSORT
- Sorting in linear time
- Order statistics: median, etc.