Recurrence

Recurrences

Recurrences

- An algorithm contains a recursive call to itself, its running time can be described by a recurrence.

Recurrences

- An algorithm contains a recursive call to itself, its running time can be described by a recurrence.
- A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.

Recurrences

- An algorithm contains a recursive call to itself, its running time can be described by a recurrence.
- A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.
- Worst case running time T(n) of merge sort can be described by

Recurrences

- An algorithm contains a recursive call to itself, its running time can be described by a recurrence.
- A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.
- Worst case running time T(n) of merge sort can be described by
- $T(n) = \begin{cases} \theta(1) & if\ n = 1 \\ 2T(n/2) + \theta(n) & if\ n > 1 \end{cases}$
  - Solution $T(n) = \theta(n\ \log_2 n)$

Recurrences…

Recurrences…

- Three methods to solve recurrences

Recurrences…

- Three methods to solve recurrences
  - substitution method
    - guess a bound and then use mathematical induction to prove our guess correct

Recurrences…

- Three methods to solve recurrences
  - substitution method
    - guess a bound and then use mathematical induction to prove our guess correct
  - recursion-tree method
    - converts the recurrence into a tree whose nodes present the costs incurred at various levels of the recursion

Recurrences…

- Three methods to solve recurrences
  - substitution method
    - guess a bound and then use mathematical induction to prove our guess correct
  - recursion-tree method
    - converts the recurrence into a tree whose nodes present the costs incurred at various levels of the recursion
  - master method
    - provides bounds for recurrences of the form
    - $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, Where $a \geq 1, b > 1$ and $f(n)$ given function

Substitution method

Substitution method

- The substitution method for solving recurrences entails two steps:
  1) Guess the form of the solution.
  2) Use mathematical induction to find the constants and show that the solution works.
- The substitution method can be used to establish either upper or lower bounds on a recurrence.
- Determine an upper bound on the recurrence
  - $T(n) = 2T(\lfloor n/2 \rfloor) + n$ and $T(1) = 1$
- Guess the solution $T(n) = O(n \lg n)$
- Prove that $T(n) \leq cn \lg n$ for an appropriate choice of constant $c > 0$

Substitution method…

- Assume that this bound holds for $\lfloor n/2 \rfloor$ that

$$T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$$

$$\begin{aligned} T(n) &\le 2\left(c\left\lfloor\frac{n}{2}\right\rfloor \lg\left\lfloor\frac{n}{2}\right\rfloor\right) + n \\ &\le cn\lg\left(\frac{n}{2}\right) + n \\ &= cn\lg n - cn\lg 2 + n \\ &= cn\lg n - cn + n \le cn\lg n \end{aligned}$$

It hold when $c \ge 1$

Substitution method…

- Need to show it also holds for boundary conditions
- for $n = 1, T(n) = 0$ which is at odds with $T(1) = 1$; base case fails
- Requires to prove $T(n) \leq cn \log n$ for $n \geq n_0$
- Appropriate value of c and n_0 must be chosen,

  here $c \geq 2$ and $n \geq n_0 (= 2)$
- substitution method provides a succinct proof that a solution to a recurrence is correct but difficult to come up with a good guess.

Recursion tree method

Recursion tree method

- Each node represents the cost of a single sub problem somewhere in the set of recursive function invocations.
- Sum the costs within each level of the tree to obtain a set of per-level costs.
- Sum all the per-level costs to determine the total cost of all levels of the recursion.
- Recursion tree useful when the recurrence describes the running time of a divide-and-conquer algorithm.
- Example : Merge sort

The Master Theorem

The Master Theorem

- The master method provides a “cookbook” method for solving recurrence of the form,

The Master Theorem

- The master method provides a “cookbook” method for solving recurrence of the form,

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

The Master Theorem

- The master method provides a "cookbook" method for solving recurrence of the form,

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- Where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

The Master Theorem

- The master method provides a "cookbook" method for solving recurrence of the form,

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- Where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

1) If $\boldsymbol{f(n) = O(n^{\log_b a-\epsilon})}$ for some constant $\boldsymbol{\epsilon > 0}$, then $\boldsymbol{T(n) = \theta\left(n^{\log_b a}\right)}$.
2) If $\boldsymbol{f(n) = \theta\left(n^{\log_b a}\right)}$ then $\boldsymbol{T(n) = \theta\left(n^{\log_b a}\, lg\, n\right)}$.
3) If $\boldsymbol{f(n) = \Omega(n^{\log_b a+\epsilon})}$ for some constant $\boldsymbol{\epsilon > 0}$, and if $\boldsymbol{af\left(\frac{n}{b}\right) \leq cf(n)}$ for some constant $\boldsymbol{c < 1}$ and all sufficiently large $\boldsymbol{n}$, then $\boldsymbol{T(n) = \theta(f(n))}$.

The Master Theorem: Example #1

The Master Theorem: Example #1

- $T(n) = 4T\left(\frac{n}{2}\right) + n$

The Master Theorem: Example #1

- $T(n) = 4T\left(\frac{n}{2}\right) + n$
  - We have $a = 4, b = 2, f(n) = n$ and $n^{\log_b a} = n^{\log_2 4} = n^2$

The Master Theorem: Example #1

- $T(n) = 4T\left(\frac{n}{2}\right) + n$
  - We have $a = 4, b = 2, f(n) = n$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n = O\left(n^{\log_b a - \epsilon}\right) = O(n^{2-1})$
  - Solution $T(n) =$?

The Master Theorem: Example #1

- $T(n) = 4T\left(\frac{n}{2}\right) + n$
  - We have $a = 4, b = 2, f(n) = n$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n = O\left(n^{\log_b a - \epsilon}\right) = O(n^{2-1})$
  - Solution $T(n) = \theta\left(n^{\log_b a}\right) = \theta(n^2)$

The Master Theorem: Example #2

The Master Theorem: Example #2

- $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

The Master Theorem: Example #2

- $T(n) = 4T\left(\frac{n}{2}\right) + n^2$
  - We have $a = 4, b = 2, f(n) = n^2$ and $n^{\log_b a} = n^{\log_2 4} = n^2$

The Master Theorem: Example #2

- $T(n) = 4T\left(\frac{n}{2}\right) + n^2$
  - We have $a = 4, b = 2, f(n) = n^2$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^2 = \theta\left(n^{\log_b a}\right) = \theta(n^2)$
  - Solution $T(n) =?$

The Master Theorem: Example #2

- $T(n) = 4T\left(\frac{n}{2}\right) + n^2$
  - We have $a = 4, b = 2, f(n) = n^2$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^2 = \theta\left(n^{\log_b a}\right) = \theta(n^2)$
  - Solution $T(n) = \theta(n^2 \log n)$

The Master Theorem: Example #2

- $T(n) = 4T\left(\frac{n}{2}\right) + n^2$
  - We have $a = 4, b = 2, f(n) = n^2$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^2 = \theta\left(n^{\log_b a}\right) = \theta(n^2)$
  - Solution $T(n) = \theta(n^2 \log n)$

  - Think about recurrence $T(n) = 4T\left(\frac{n}{2}\right) + n^2 \log n$.

The Master Theorem: Example #3

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^3 = \Omega\left(n^{\log_b a + \epsilon}\right) = \Omega(n^{2+1})$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^3 = \Omega\left(n^{\log_b a+\epsilon}\right) = \Omega(n^{2+1})$
  - Check the regularity condition, $af\left(\frac{n}{b}\right) \leq cf(n), 0 < c < 1$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^3 = \Omega\left(n^{\log_b a+\epsilon}\right) = \Omega(n^{2+1})$
  - Check the regularity condition, $af\left(\frac{n}{b}\right) \leq cf(n), 0 < c < 1$

    $\Rightarrow 4f\left(\frac{n}{2}\right) < cf(n)$, here $f(n) = n^3$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^3 = \Omega\left(n^{\log_b a+\epsilon}\right) = \Omega(n^{2+1})$
  - Check the regularity condition, $af\left(\frac{n}{b}\right) \le cf(n), 0 < c < 1$

$$\Rightarrow 4f\left(\frac{n}{2}\right) < cf(n), \text{here } f(n) = n^3$$

$$\Rightarrow 4\left(\frac{n}{2}\right)^3 < cn^3$$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^3 = \Omega\left(n^{\log_b a + \epsilon}\right) = \Omega(n^{2+1})$
  - Check the regularity condition, $af\left(\frac{n}{b}\right) \leq cf(n), 0 < c < 1$

    $\Rightarrow 4f\left(\frac{n}{2}\right) < cf(n)$, here $f(n) = n^3$

    $\Rightarrow 4\left(\frac{n}{2}\right)^3 < cn^3$

    $\Rightarrow \frac{n^3}{2} < cn^3$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^3 = \Omega\left(n^{\log_b a + \epsilon}\right) = \Omega(n^{2+1})$
  - Check the regularity condition, $af\left(\frac{n}{b}\right) \le cf(n), 0 < c < 1$

$$\Rightarrow 4f\left(\frac{n}{2}\right) < cf(n), \text{here } f(n) = n^3$$

$$\Rightarrow 4\left(\frac{n}{2}\right)^3 < cn^3$$

$$\Rightarrow \frac{n^3}{2} < cn^3, c = ?$$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^3 = \Omega\left(n^{\log_b a + \epsilon}\right) = \Omega(n^{2+1})$
  - Check the regularity condition, $af\left(\frac{n}{b}\right) \leq cf(n), 0 < c < 1$

    $\Rightarrow 4f\left(\frac{n}{2}\right) < cf(n)$, here $f(n) = n^3$

    $\Rightarrow 4\left(\frac{n}{2}\right)^3 < cn^3$

    $\Rightarrow \frac{n^3}{2} < cn^3, c = ?$
  - Solution $T(n) = ?$

The Master Theorem: Example #3

- $T(n) = 4T\left(\frac{n}{2}\right) + n^3$
  - We have $a = 4, b = 2, f(n) = n^3$ and $n^{\log_b a} = n^{\log_2 4} = n^2$
  - Here $f(n) = n^3 = \Omega\left(n^{\log_b a+\epsilon}\right) = \Omega(n^{2+1})$
  - Check the regularity condition, $af\left(\frac{n}{b}\right) \leq cf(n), 0 < c < 1$

    $\Rightarrow 4f\left(\frac{n}{2}\right) < cf(n)$, here $f(n) = n^3$

    $\Rightarrow 4\left(\frac{n}{2}\right)^3 < cn^3$

    $\Rightarrow \frac{n^3}{2} < cn^3, c =?$
  - Solution $T(n) = \theta(f(n)) = \theta(n^3)$

Changing Variable

Changing Variable

- Consider the following recurrence
- $T(n) = 2T\left(\left\lfloor\sqrt{n}\right\rfloor\right) + \log n$

Changing Variable

- Consider the following recurrence
- $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log n$
  - Take, $m = \log n$

Changing Variable

- Consider the following recurrence
- $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log n$
  - Take, $m = \log n$
  - $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$

Changing Variable

- Consider the following recurrence
- $T(n) = 2T\left(\left\lfloor\sqrt{n}\right\rfloor\right) + \log n$
  - Take, $m = \log n$
  - $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$
  - Now, we can rename $S(m) = T(2^m)$ to produce the new recurrence

Changing Variable

- Consider the following recurrence
- $T(n) = 2T\left(\left\lfloor\sqrt{n}\right\rfloor\right) + \log n$
  - Take, $m = \log n$
  - $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$
  - Now, we can rename $S(m) = T(2^m)$ to produce the new recurrence

$$S(m) = 2S\left(\frac{m}{2}\right) + m$$

Changing Variable

- Consider the following recurrence
- $T(n) = 2T\left(\left\lfloor\sqrt{n}\right\rfloor\right) + \log n$
  - Take, $m = \log n$
  - $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$
  - Now, we can rename $S(m) = T(2^m)$ to produce the new recurrence

$S(m) = 2S\left(\frac{m}{2}\right) + m$

Solution for $S(m) =$?

Changing Variable

- Consider the following recurrence
- $T(n) = 2T\left(\lfloor\sqrt{n}\rfloor\right) + \log n$
  - Take, $m = \log n$
  - $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$
  - Now, we can rename $S(m) = T(2^m)$ to produce the new recurrence

$S(m) = 2S\left(\frac{m}{2}\right) + m$

Solution for $S(m) = O(m \log m)$

Changing Variable

- Consider the following recurrence
- $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log n$
  - Take, $m = \log n$
  - $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$
  - Now, we can rename $S(m) = T(2^m)$ to produce the new recurrence

$S(m) = 2S\left(\frac{m}{2}\right) + m$

Solution for $S(m) = O(m \log m)$

Changing back from $S(m)$ to $T(n)$ then $T(n) =$?

Changing Variable

- Consider the following recurrence
- $T(n) = 2T\left(\lfloor\sqrt{n}\rfloor\right) + \log n$
  - Take, $m = \log n$
  - $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$
  - Now, we can rename $S(m) = T(2^m)$ to produce the new recurrence

$S(m) = 2S\left(\frac{m}{2}\right) + m$

Solution for $S(m) = O(m \log m)$

Changing back from $S(m)$ to $T(n)$

$T(n) = T(2^m) = S(m) = O(m \log m) = O(\log n \log \log n)$