EECS 3101

Prof. Andy Mirzaian

Summations

&

Recurrence Relations

STUDY MATERIAL:

- [CLRS] Appendix A, chapter 4
- Lecture Note 3

Summations:

$$\sum_{i=1}^{n} f(i) = f(1) + f(2) + \cdots + f(n) = \Theta(\,?\,)$$

$$\sum_{i=1}^{n} 2^{\sqrt{i}} = \Theta\left(\sqrt{n}\, 2^{\sqrt{n}}\right)$$

Recurrence Relations:

$$T(n) = \begin{cases} T(n-1) + f(n) & \forall n \geq 1 \\ 0 & \forall n < 1 \end{cases} \quad \Longrightarrow \quad T(n) = \sum_{i=1}^{n} f(i)$$

$$T(n) = \begin{cases} 2T(n-1) + 1 & \forall n \geq 1 \\ 0 & \forall n < 1 \end{cases} \quad \Longrightarrow \quad T(n) = \Theta(2^n)$$

TOPICS

Summations & Recurrence Relations

arise in the running time analysis of algorithms.

- **SUMMATIONS:**
  - Classic Methods: Arithmetic, Geometric, Harmonic, Bounded Tail
  - Approximating Summation by Integration
  - Summation by Parts
  - **ASSUME:** ASymptotic SUmmation Made Easy

- **RECURRENCE RELATIONS:**
  - Iteration method
  - Recursion tree method
  - Divide-&-Conquer Recurrence: The Master Method
  - Guess-&-Verify method
  - Full History Recurrences
  - Variable Substitution method

SUMMATION

$$f : \mathbb{N} \to \mathbb{R}^+$$

$$S(n) = \sum_{i=1}^{n} f(i) = f(1) + f(2) + f(3) + \cdots + f(n)$$

Classic Methods

- **Arithmetic (or polynomial):** $S(n) = \Theta(n\, f(n))$

  $f(n) = n$: $1 + 2 + \cdots + n = n(n+1)/2 = \Theta(n^2)$

  $f(n) = n^2$: $1^2 + 2^2 + \cdots + n^2 = n(n+1)(2n+1)/6 = \Theta(n^3)$

  $f(n) = n^d$: $1^d + 2^d + \cdots + n^d = \Theta(n^{d+1})$, for any constant real $d > -1$

- **Geometric (or exponential):** $S(n) = \Theta(f(n))$

  $f(n) = x^n$: $1 + x + x^2 + \cdots + x^n = (x^{n+1} - 1)/(x - 1)$ for any constant real $0 \neq |x| \neq 1$

  $= \Theta(x^n)$ if $x > 1$ (Geometric increasing)

  $f(n) = 2^n$: $1 + 2 + 2^2 + \cdots + 2^n = \Theta(2^n)$ (with $x = 2$)

- **Harmonic:** $S(n) = \Theta(\log n)$

  $f(n) = 1/n$: $H(n) = 1 + 1/2 + 1/3 + \cdots + 1/n \approx \ln n = \Theta(\log n)$

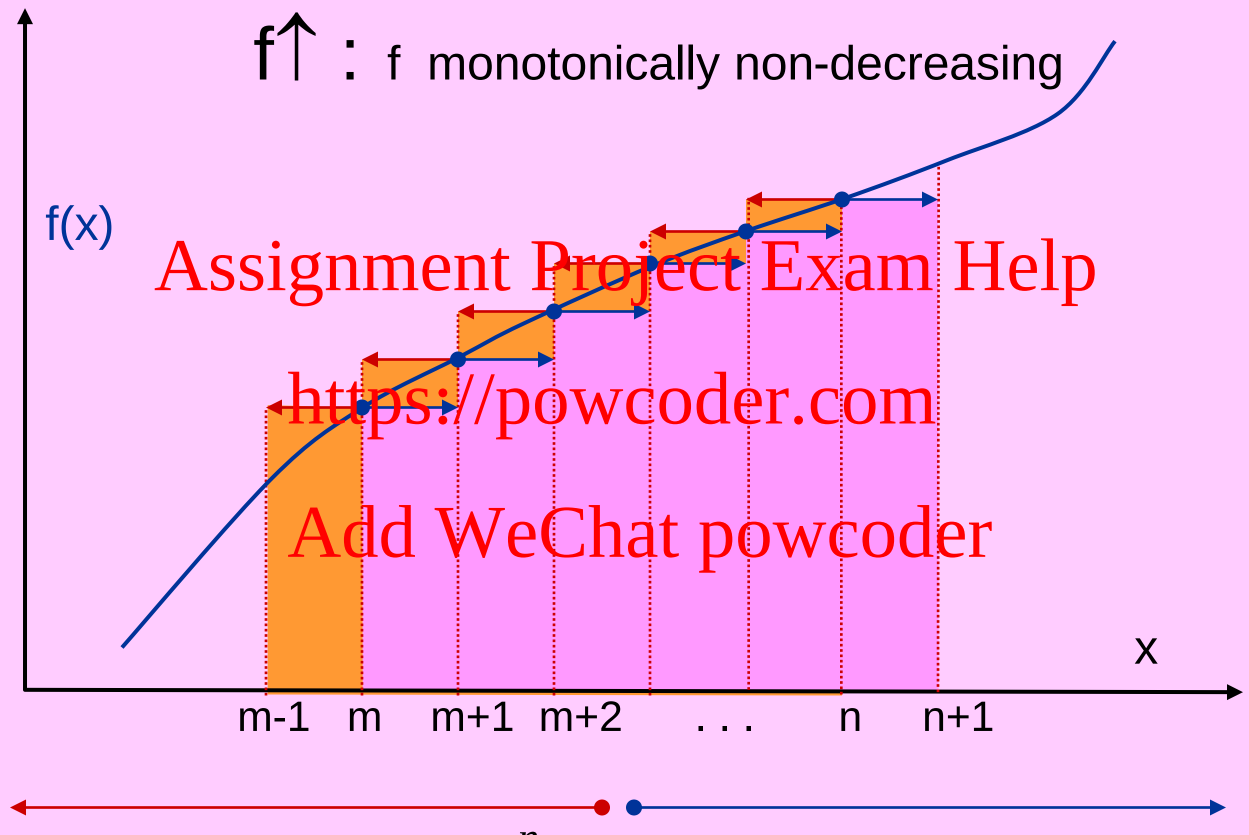
- **Bounded Tail:** $S(n) = \Theta(1)$

  $1 + x + x^2 + \cdots + x^n = (1 - x^{n+1})/(1-x) = \Theta(1)$ if $0<x<1$. (Geometric decreasing)

  $f(n) = 2^{-n}$: $1 + 1/2 + 1/4 + \cdots + 1/2^n = \Theta(1)$ (with $x = ½$)

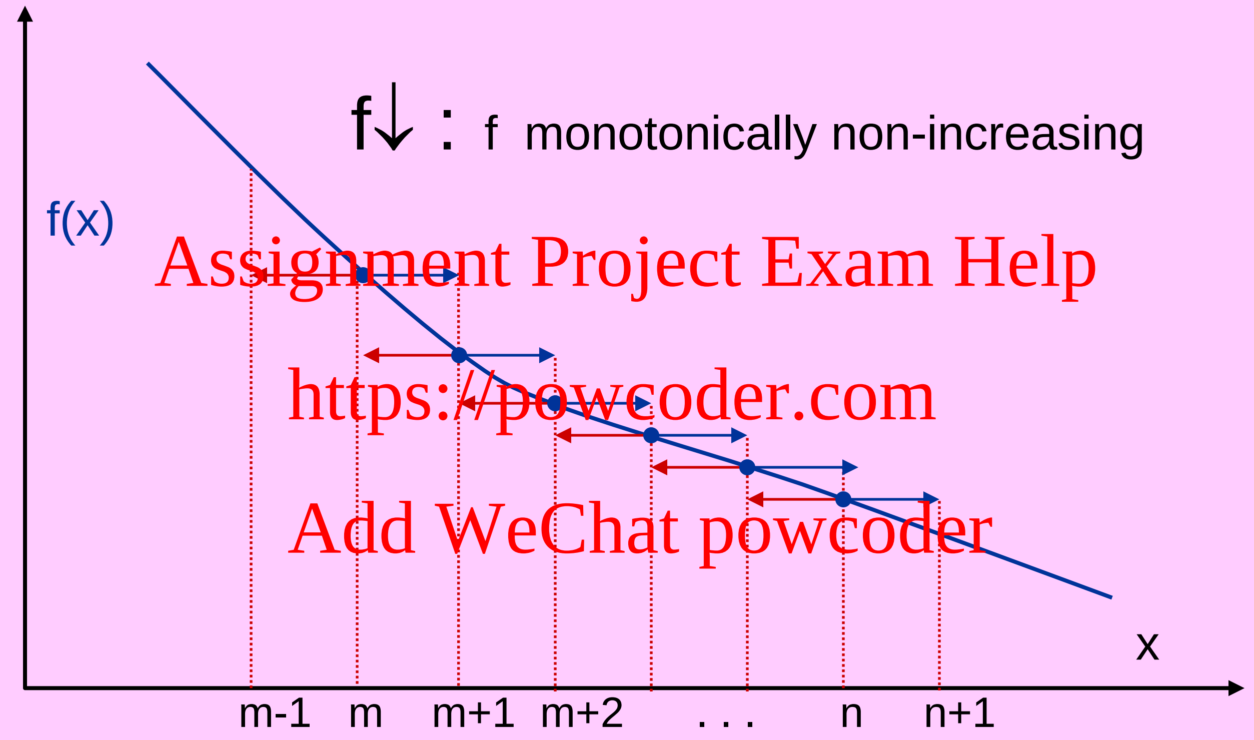
  $f(n) = n^{-2}$: $1 + 1/2^2 + 1/3^2 + \cdots + 1/n^2 = \Theta(1)$ (Arithmetic decreasing)

Approximating $\sum f(i)$ by $\int f(x)dx$



$$\int_{m-1}^{n} f(x)dx \le \sum_{i=m}^{n} f(i) \le \int_{m}^{n+1} f(x)dx$$

Approximating $\sum f(i)$ by $\int f(x)dx$



$$\int_{m-1}^{n} f(x)dx \geq \sum_{i=m}^{n} f(i) \geq \int_{m}^{n+1} f(x)dx$$

NOTE: direction of inequalities changed

Approximating $\sum f(i)$ by $\int f(x)dx$

$$f\uparrow: \quad \int_{m-1}^{n} f(x)dx \le \sum_{i=m}^{n} f(i) \le \int_{m}^{n+1} f(x)dx$$

$$f\downarrow: \quad \int_{m-1}^{n} f(x)dx \ge \sum_{i=m}^{n} f(i) \ge \int_{m}^{n+1} f(x)dx$$

$S(n) = f(1) + f(2) + \ldots + f(n)$ $(m = 1)$

Example 1: $S(n) = 1^3 + 2^3 + \ldots + n^3$.

$f(x) = x^3$: $f\uparrow$ and $\int f(x)\,dx = \int x^3\,dx = x^4 / 4.$

$n^4 / 4 \le S(n) \le ((n+1)^4 - 1)/4 \quad \Rightarrow \quad S(n) = \Theta(n^4)$

Example 2: $H(n) = 1 + 1/2 + 1/3 + \ldots + 1/n$.

$f(x) = 1/x$: $f\downarrow$ and $\int f(x)\,dx = \int dx/x = \ln x.$

$1 + \ln n \ge H(n) \ge \ln(n+1) \quad \Rightarrow \quad H(n) = \Theta(\log n)$

Caution against dividing by zero!
Separate f(1) from the summation.

Summation by Parts: Arithmetic

$f(n) = n:\quad S(n) = f(1) + f(2) + f(3) + \cdots + f(n).$

$$S(n) = \underbrace{\overbrace{\underbrace{1 + 2 + 3 + \cdots + (\lfloor n/2 \rfloor - 1) + \lfloor n/2 \rfloor}_{\substack{\xleftrightarrow{\lfloor n/2 \rfloor} \\ \ge 0}} + \underbrace{(\lfloor n/2 \rfloor + 1) + (\lfloor n/2 \rfloor + 2) + \cdots + n}_{\substack{\xleftrightarrow{\lceil n/2 \rceil} \\ \ge \lceil n/2 \rceil \cdot \lceil n/2 \rceil \\ \ge \Omega(n^2)}}}^{\le n \cdot n \ \le O(n^2)}}_{\ge \Omega(n^2)}$$

$\therefore \quad S(n) = \Theta(n^2)$

Summation by Parts: Harmonic

$$H(n) = \underbrace{\overbrace{\underbrace{\overbrace{1}^{1}}_{1} + \underbrace{\overbrace{\tfrac{1}{2}}^{<1}}_{½} + \underbrace{\overbrace{\tfrac{1}{3} + \tfrac{1}{4}}^{< 2(1/2) = 1}}_{>2(1/4) = ½} + \underbrace{\overbrace{\tfrac{1}{5} + \tfrac{1}{6} + \tfrac{1}{7} + \tfrac{1}{8}}^{< 4(1/4) = 1}}_{> 4(1/8) = ½} + \underbrace{\overbrace{\tfrac{1}{9} + \tfrac{1}{10} + \tfrac{1}{11} + \tfrac{1}{12} + \tfrac{1}{13} + \tfrac{1}{14} + \tfrac{1}{15} + \tfrac{1}{16}}^{< 8(1/8) = 1}}_{> 8(1/16) = ½} + \cdots + \tfrac{1}{n}}^{\le\ 1 + \lceil \log n \rceil\ \le\ O(\log n)}}_{\ge\ 1 + ½ \lfloor \log n \rfloor\ \ge\ \Omega(\log n)}$$

$$\therefore \quad H(n) = \Theta(\log n)$$

ASSUME : ASymptotic SUmmation Made Easy

$f: \mathcal{N} \to \Re^+ :\quad S(n) = f(1) + f(2) + f(3) + \cdots + f(n)$

$$\max_n(f) = \max \{ f(1), f(2), \ldots, f(n) \}$$
$$\min_n(f) = \min \{ f(1), f(2), \ldots, f(n) \}$$
$$\text{ave}_n(f) = \text{average} \{ f(1), f(2), \ldots, f(n) \}$$

$$S(n) = n \cdot \text{ave}_n(f)$$

$$0 \le \min_n(f) \le \text{ave}_n(f) \le \max_n(f)$$

$$f\uparrow \;\Rightarrow \min_n(f) = f(1), \quad \max_n(f) = f(n)$$

$$f\downarrow \;\Rightarrow \min_n(f) = f(n), \quad \max_n(f) = f(1)$$

$$\max\{ n \cdot \min_n(f), \boxed{\max_n(f) \} \le S(n) \le n \cdot \max_n(f)}$$

$$\therefore S(n) = \Theta(g(n) \cdot \max_n(f)) \quad \text{for some} \quad 1 \le g(n) \le n$$

ASSUME: A method for finding g(n) (and hence, S(n)).

ASSUME on monotone f(n)

$f: \mathcal{N} \rightarrow \Re^+$ $\quad S(n) = f(1) + f(2) + f(3) + \cdots + f(n)$

FACT: f is monotone $\Rightarrow$ exactly one of the following holds:

(1) $f(n) = 0 \quad \forall n$

(2) $f(n) = \Theta(1)$

(3) $f(n) = \omega(1)$ and $f\uparrow$

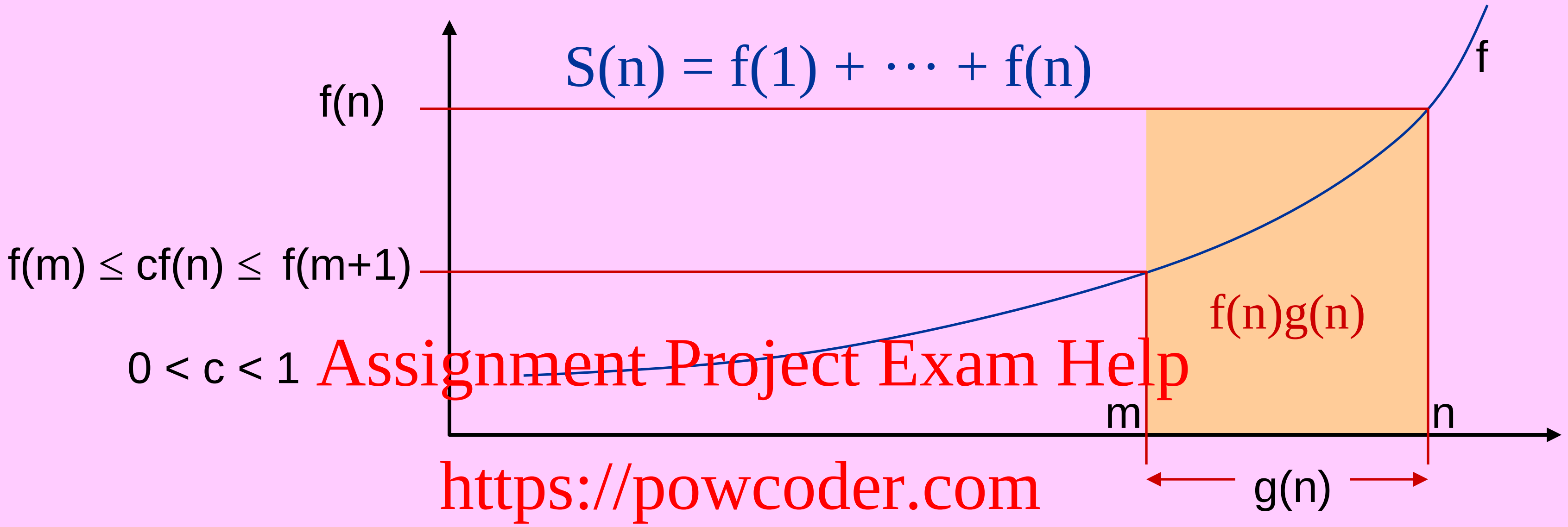
(4) $f(n) = o(1)$ and $f\downarrow$ and $f(1) > 0$

Cases (1) & (2): $g(n) = \Theta(n)$, $\quad S(n) = \Theta(g(n)\, f(n)) = \Theta(n\, f(n))$

Case (3): Considered next.

Case (4): Similar methods apply. Exercise.

ASSUME for f↑ & f(n) = ω(1)



Step 1) Find **m** such that **log f(m) = log f(n) – Θ(1)**

Step 2) **g(n) ← ⌈n – m⌉** [Note: 1 ≤ g(n) ≤ n]

THEOREM: f↑ & g↑ ⇒ S(n) = Θ(f(n)g(n)).

ASSUME: examples

Polynomial: $f(n) = n^d$ (const. $d \geq 0$) :

$\log f(m) = \log f(n) - \Theta(1)$

$d \log m = d \log n - d = d \log n/2$

$m = n/2 \Rightarrow g(n) = n-m = n/2.$

$f\uparrow$ & $g\uparrow \Rightarrow S(n) = \Theta(f(n)g(n)) = \Theta(n^{d+1}).$

Exponential: $f(n) = 2^n$:

$\log f(m) = \log f(n) - \Theta(1)$

$m = n - 1 \Rightarrow g(n) = n-m = 1$

$f\uparrow$ & $g\uparrow \Rightarrow S(n) = \Theta(f(n)g(n)) = \Theta(2^n).$

$\log(1 - x) = \Theta(\ln(1 - x)) = -\Theta(x)$ for $x = o(1)$

Taylor Series Expansion :

$$f(x) = f(0) + \frac{f^{(1)}(0)}{1!}x + \frac{f^{(2)}(0)}{2!}x^2 + \frac{f^{(3)}(0)}{3!}x^3 + \cdots$$

$f(x) = \ln(1 - x)$:
$$\ln(1 - x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \cdots$$

Example:
$$\begin{aligned} &\log(n - \Theta(\log n)) \\ &= \log\left[n\left(1 - \Theta\left(\frac{\log n}{n}\right)\right)\right] \\ &= \log n + \log\left(1 - \Theta\left(\frac{\log n}{n}\right)\right) \qquad \left[\ \text{take } x = \Theta\left(\frac{\log n}{n}\right)\ \right] \\ &= \log n - \Theta\left(\frac{\log n}{n}\right) \\ &= (\log n)\left(1 - \Theta\left(\frac{1}{n}\right)\right) \end{aligned}$$

ASSUME: examples

Super-polynomial sub-exponential 1: $f(n) = 2^{\sqrt{n}}$:

$$\log f(m) = \log f(n) - \Theta(1)$$

$$\sqrt{m} = \sqrt{n} - 1$$ [Now square both sides]

$$m = (\sqrt{n} - 1)^2 = n - 2\sqrt{n} + 1$$

$$\Rightarrow g(n) = n - m = 2\sqrt{n} - 1 = \Theta(\sqrt{n}).$$

$f\uparrow$ & $g\uparrow$ $\Rightarrow$ $S(n) = \Theta(f(n)g(n)) = \Theta(\sqrt{n}2^{\sqrt{n}})$.

Super-polynomial sub-exponential 2: $f(n) = 2^{n/\log n}$:

$$\log f(m) = \log f(n) - \Theta(1)$$

(*) $m/\log m = n/\log n - \Theta(1)$ [Multiply across by log m ≈ log n]

$m = n - \Theta(\log n)$ [Verify this satisfies (*) with "="]

$m/\log m = (n - \Theta(\log n)) / \log (n - \Theta(\log n))$

$= (n - \Theta(\log n)) / (\log n)(1 - \Theta(1/n))$ [See previous page]

$= (n/\log n - \Theta(1)) / (1 - \Theta(1/n))$

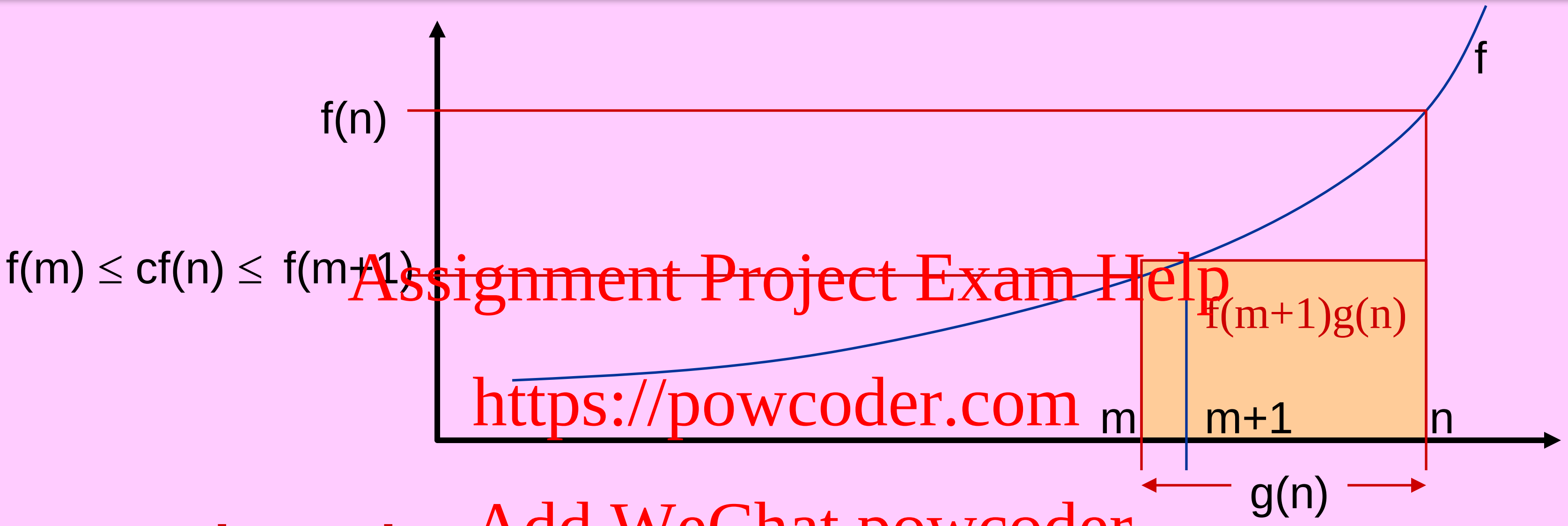
$= n/\log n - \Theta(1)$ [Why? Multiply sides by $1 - \Theta(1/n)$]

$$\Rightarrow g(n) = n - m = \Theta(\log n)$$

$f\uparrow$ & $g\uparrow$ $\Rightarrow$ $S(n) = \Theta(f(n)g(n)) = \Theta(2^{n/\log n} \log n)$.

ASSUME: Lower Bound Proof

THEOREM: $f\uparrow$ & $g\uparrow \Rightarrow S(n) = f(1) + \cdots + f(n) = \Theta(f(n)g(n))$.

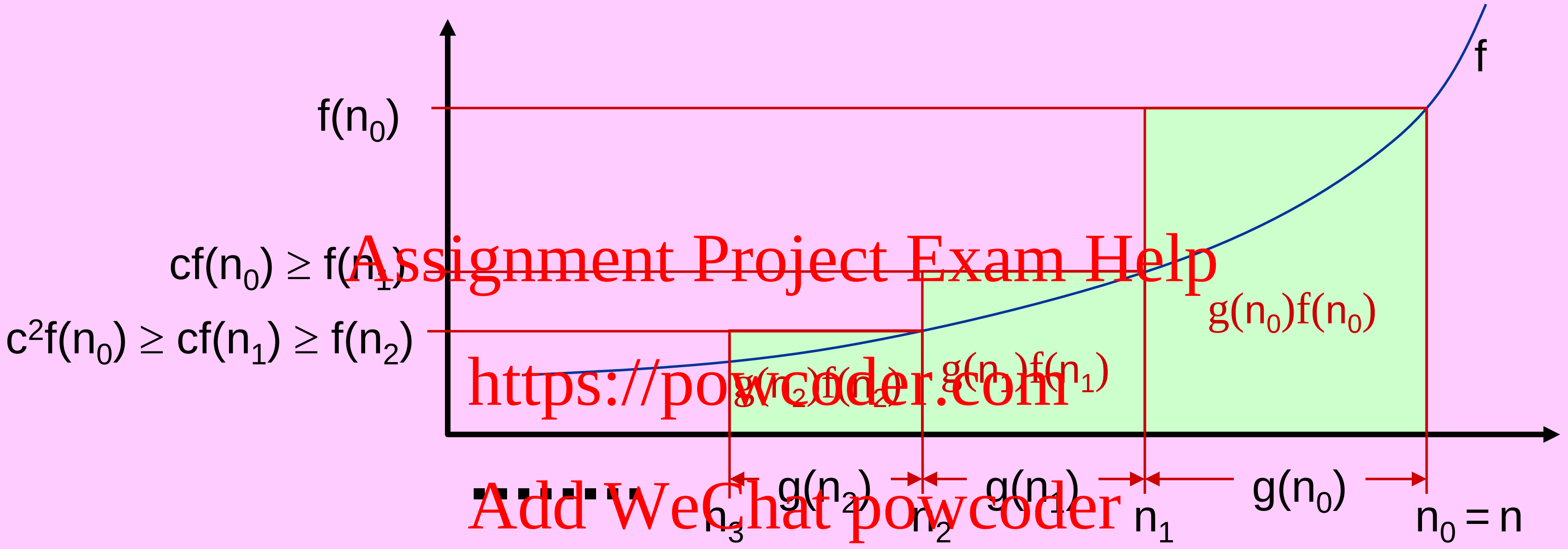


Lower bound:

$$\begin{aligned} S(n) &\geq S(n) - S(m) \\ &= f(m+1) + f(m+2) + \cdots + f(n) \\ &\geq \text{shaded rectangular area} \qquad \text{since } f\uparrow \\ &= f(m+1)\, g(n) \\ &\geq c\, f(n)\, g(n) \\ &\geq \Omega(f(n)\, g(n)). \end{aligned}$$

ASSUME: Upper Bound Proof

THEOREM: $f\uparrow \;\&\; g\uparrow \;\Rightarrow\; S(n) = f(1) + \cdots + f(n) = \Theta(f(n)g(n))$.



Upper bound:

$S(n) \le$ sum of shaded rectangular areas — since $f\uparrow$

$= g(n_0)f(n_0) + g(n_1)f(n_1) + g(n_2)f(n_2) + \cdots$

$\le g(n)\;[\,f(n_0) + f(n_1) + f(n_2) + \cdots\,]$ — since $g\uparrow$

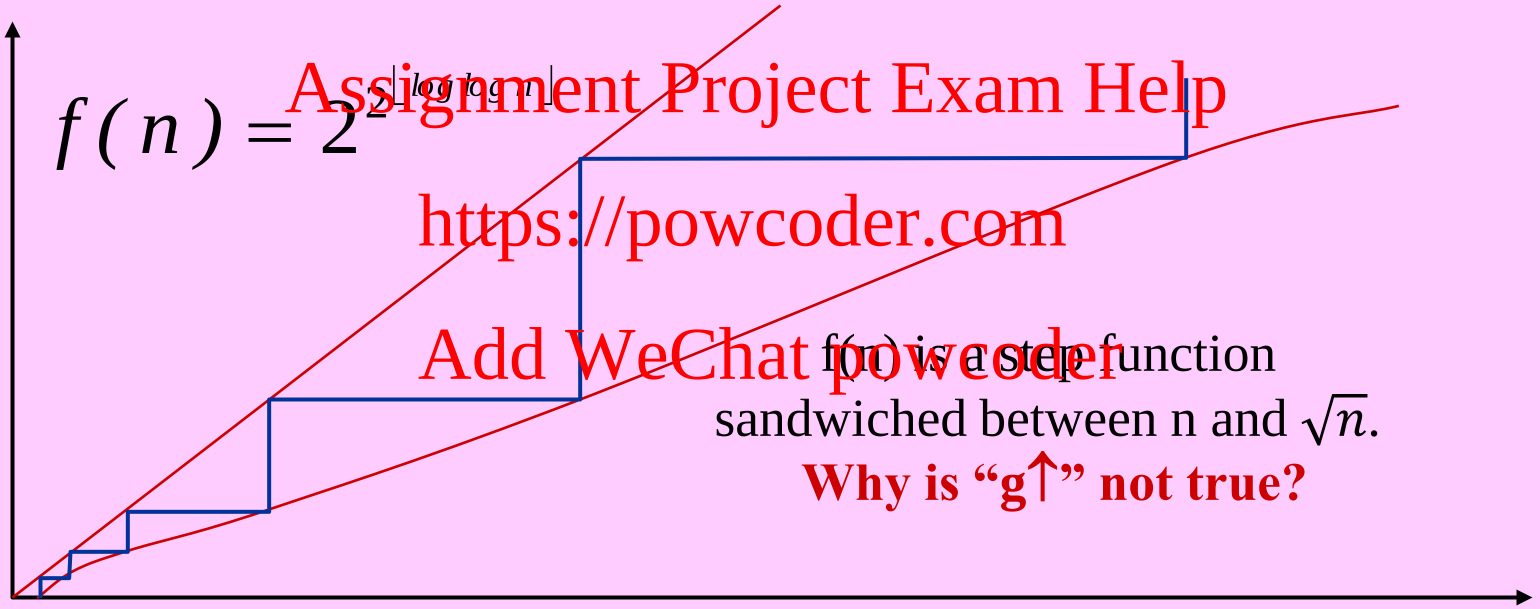
$\le g(n)f(n)\,[\,1 + c + c^2 + \cdots\,]$ — $f(n_{i+1}) \le c\, f(n_i)$

$\le g(n)f(n) / (1 - c)$ — classic geometric decreasing sum

$\le O(\,f(n)\,g(n)\,)$.

Caution!

To guarantee conclusion of the theorem, its premise must hold.
The premise insists that not only f↑, but also g↑ holds.
f↑ without g↑ is possible. *The following is an example:*



$$f(n) = 2^{2^{\lfloor \log\log n \rfloor}}$$

f(n) is a step function sandwiched between n and $\sqrt{n}$.

Why is "g↑" not true?

However, with Simple Analytical Functions (SAF), i.e., functions composed of only constants, arithmetic, exponential, logarithm, and functional composition, such a "bizarre" situation will not arise, and hence, the statement requiring "g↑" can be omitted from the premise of the theorem.

Simple Analytical Function (SAF)

SAF : a function composed of a finite number of applications of constants, arithmetic, exponential, logarithm, & functional composition.

Example:
$$\frac{4n^{2\sqrt{n}\log\log n}}{(1 + 6(\log n) / n)^{2+7n}}$$

FACTS: A SAF f(n) has the following properties:

a) log f(n) and ∂f(x)/∂x are also SAFs.

b) has only a finite # of roots. *(does not oscillate)*

c) is asymptotically sign invariant. *(past its last real root)*

d) is asymptotically monotonic. *(part (c) on its derivative)*

e) In ASSUME: f(n)↑ ⇒ g(n)↑

f) when comparing SAFs asymptotically, "$\exists n_0 \forall n \geq n_0$" can be replaced by the phrase "for infinitely many n" ("for i.m. n" for short).
E.g., "for all n that are positive integer powers of 2". *("past last real root")*

Our short article says more on this topic.
The next table shows a summary of ASSUME on SAF.

f(n) : a SAF

S(n) = f(1) + f(2) + ⋯ + f(n)

Sum Type	$f(n)$	**Example f(n)**	$S(n)$
Geometric (or exponential)	$2^{\Omega(n)}$	$\frac{3^{2n}\log n}{n^3}$	$\Theta(f(n))$
Arithmetic (or polynomial)	$n^{\Theta(1)-1}$	$n^{3.5}\log n$	$\Theta(nf(n))$
quasi Harmonic	$\Theta\left(\frac{\log^e n}{n}\right)$	$\frac{\log^3 n}{n}$	$\Theta(\log^{e+1} n)$ *if* $e > -1$ $\Theta(\log\log n)$ *if* $e = -1$ $\Theta(1)$ *if* $e < -1$
Bounded Tail	$n^{-\Omega(1)-1}$	$\frac{n^4}{2^{3n}}, \frac{1}{n^{2+\log n}}$	$\Theta(1)$

A Simple form

The SAFs we encounter, usually have the following simple form:

$$f(n) = \Theta\left(t^n n^d \log^e n\right)$$

for some real constants t>0, d, e. We classify S(n) for such f(n) as:

$t > 1$		Geometric (increasing)
$t = 1$	$d > -1$	Arithmetic
	$d = -1$	quasi Harmonic
	$d < -1$	Bounded tail (polynomially decreasing)
$0 < t < 1$		Bounded tail (geometrically decreasing)

RECURRENCE RELATIONS

$$T(n) = \begin{cases} 2\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \Theta(n) & \text{for } \mathrm{n} > \mathrm{O}(1) \\ O(1) & \text{for } \mathrm{n} = \mathrm{O}(1) \end{cases}$$

$$\Downarrow$$

$$T(n) = \Theta(n \log n)$$

The Iteration Method

$$T(n) = \begin{cases} 0 & \text{for} \quad n = 0 \\ T(n-1) + f(n) & \text{for} \quad n \ge 1 \end{cases}$$

The iteration method (or unwinding the recurrence):

T(n) = f(n) + T(n –1)

= f(n) + f(n-1) + T(n –2)

= f(n) + f(n-1) + f(n-2) + T(n –3)

= …

= f(n) + f(n-1) + f(n-2) + ⋯ + f(3) + f(2) + T(1)

= f(n) + f(n-1) + f(n-2) + ⋯ + f(3) + f(2) + f(1) + T(0)

= f(n) + f(n-1) + f(n-2) + ⋯ + f(3) + f(2) + f(1).

f(n) is
the driving function
of the recurrence

Example: $f(n) = \Theta(n) \quad \Rightarrow \quad T(n) = \Theta(n^2)$.

$f(n) = \Theta(2^n) \quad \Rightarrow \quad T(n) = \Theta(2^n)$.

The Iteration Method

$$T(n) = \begin{cases} 0 & \text{for } n < 1 \\ 2T\left(\frac{n}{2}\right) + n & \text{for } n \geq 1 \end{cases}$$

$T(n) = n + 2T(n/2)$

$= n + 2\,[n/2 + 2T(n/2^2)]$

$= 2n + 2^2\,T(n/2^2)$

$= 2n + 2^2\,[n/2^2 + 2\,T(n/2^3)]$

$= 3n + 2^3\,T(n/2^3)$

$= 3n + 2^3\,[n/2^3 + 2\,T(n/2^4)]$

$= 4n + 2^4\,T(n/2^4)$

$= \ldots$

$= kn + 2^k\,T(n/2^k)$ take $k = 1 + \lfloor \log n \rfloor$, $T(n/2^k) = 0$

$= n(1 + \lfloor \log n \rfloor)$

$= \Theta(n \log n)$

The Iteration Method

$$T(n) = \begin{cases} 0 & \text{for } n < 1 \\ 2T\left(\frac{n}{2}\right) + n^2 & \text{for } n \ge 1 \end{cases}$$

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

$= n^2 + 2\,[(n/2)^2 + 2T(n/2^2)]$

$= n^2(1 + \tfrac{1}{2}) + 2^2\,T(n/2^2)$

$= n^2(1 + \tfrac{1}{2}) + 2^2\,[(n/2^2)^2 + 2\,T(n/2^3)]$

$= n^2(1 + \tfrac{1}{2} + \tfrac{1}{2^2}) + 2^3\,T(n/2^3)$

$= n^2(1 + \tfrac{1}{2} + \tfrac{1}{2^2}) + 2^3\,[(n/2^3)^2 + 2T(n/2^3)]$

$= n^2(1 + \tfrac{1}{2} + \tfrac{1}{2^2} + \tfrac{1}{2^3}) + 2^4\,T(n/2^4)$

$= \ \ldots$

$= n^2\,(1 + \tfrac{1}{2} + \tfrac{1}{2^2} + \tfrac{1}{2^3} + \cdots + \tfrac{1}{2^k}) + 2^{k+1}\,T(n/2^{k+1})$

take $k = \lfloor \log n \rfloor$, $T(n/2^{k+1}) = 0$

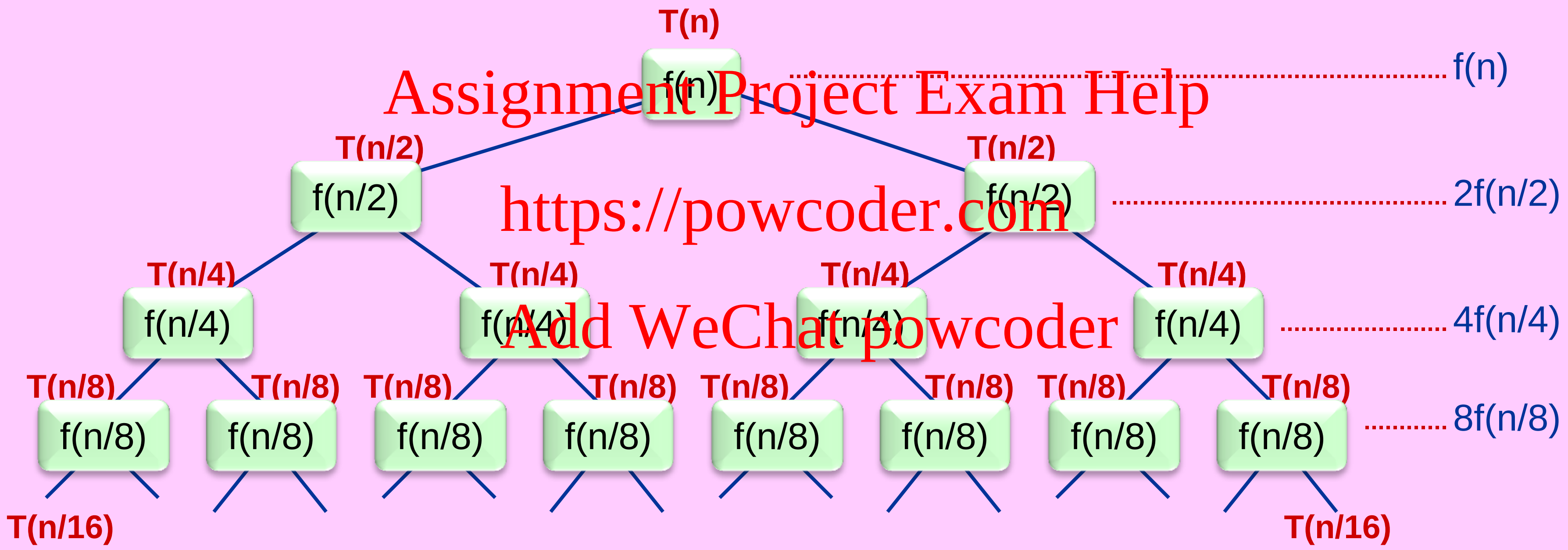
$= n^2\,(1 + \tfrac{1}{2} + \tfrac{1}{2^2} + \tfrac{1}{2^3} + \cdots + \tfrac{1}{2^k})$

$= n^2 \cdot \Theta(1)$ geometric decreasing

$\longrightarrow = \Theta(n^2)$

The Recursion Tree Method

$$T(n) = \begin{cases} 0 & \text{for } n < 1 \\ 2T\left(\frac{n}{2}\right) + f(n) & \text{for } n \geq 1 \end{cases}$$



$$T(n) = f(n) + 2f(n/2) + 4f(n/4) + 8f(n/8) + \cdots$$

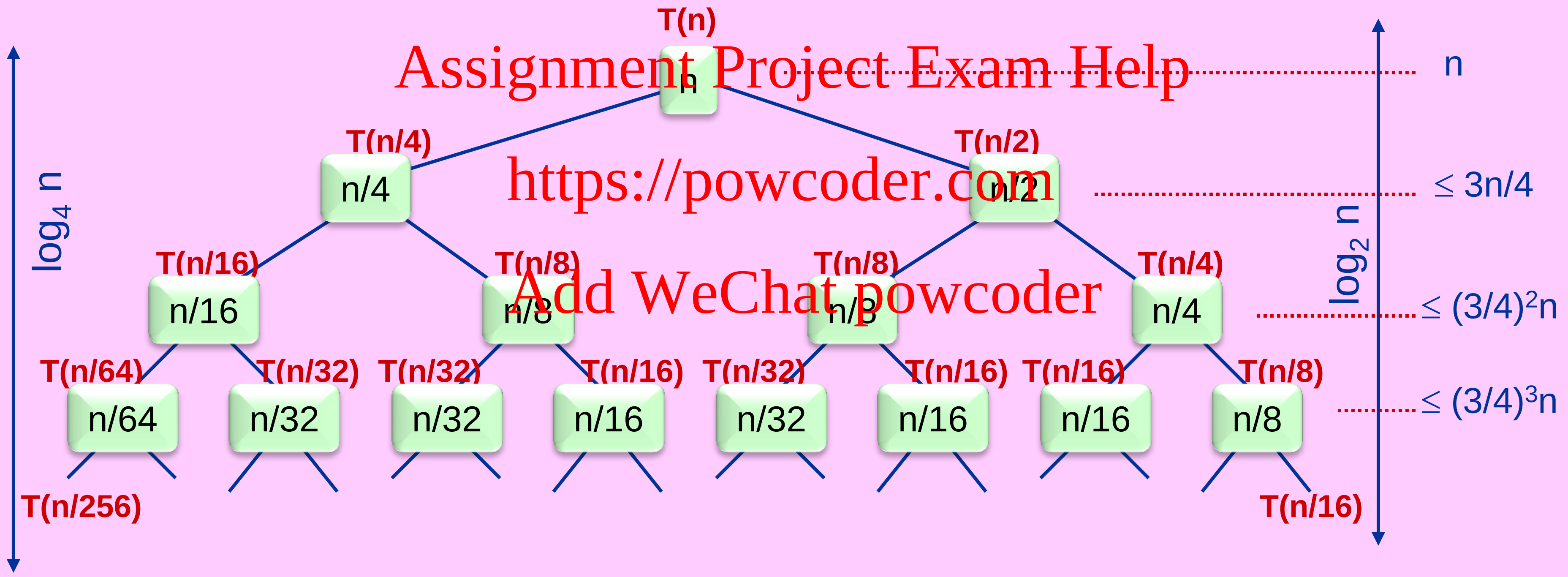
$$= \sum_{i=0}^{\lfloor \log n \rfloor} 2^i f\left(n/2^i\right)$$

The Recursion Tree Method

$$T(n) = \begin{cases} 0 & \text{for } n < 4 \\ T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n & \text{for } n \geq 4 \end{cases}$$

CLAIM: T(n) = Θ(n).

Lower bound: T(n) ≥ Ω(n) obvious



$$T(n) \leq n + \left(\tfrac{3}{4}\right)n + \left(\tfrac{3}{4}\right)^2 n + \left(\tfrac{3}{4}\right)^3 n + \cdots$$

$$= n\left(1 + \left(\tfrac{3}{4}\right) + \left(\tfrac{3}{4}\right)^2 + \left(\tfrac{3}{4}\right)^3 + \cdots\right) \leq 4n \leq O(n).$$

Upper bound

The Recursion Tree Method

$$T(n) = \begin{cases} 0 & \text{for } n < 4 \\ T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n & \text{for } n \geq 4 \end{cases}$$

This is a special case of the following recurrence:

$$T(n) = T(\alpha n) + T(\beta n) + \Theta(n)$$

where $0 \leq \alpha < 1$ and $0 \leq \beta < 1$ are real constant parameters.

FACT:

(1) $\alpha + \beta < 1 \Rightarrow T(n) = \Theta(n)$ [linear]

(2) $\alpha + \beta = 1 \Rightarrow T(n) = \Theta(n \log n)$

(3) $\alpha + \beta > 1 \Rightarrow T(n) = \Theta(n^d)$ [super-linear poly]

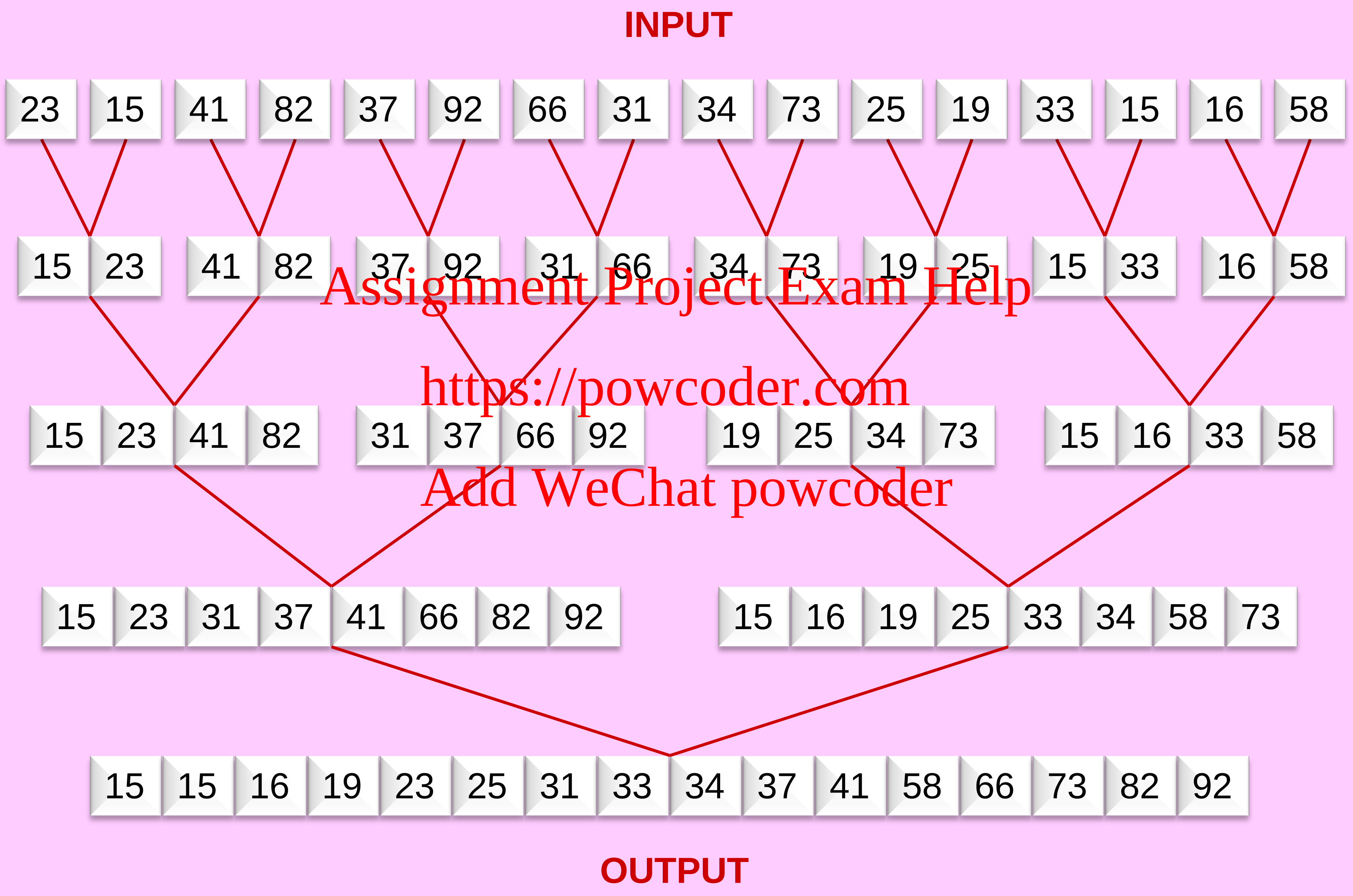
where $d > 1$ is the unique constant
that satisfies the equation $\alpha^d + \beta^d = 1$.
[See the **guess-&-verify method** later & Exercise 9.]

Divide-&-Conquer

MergeSort is the prototypical divide-&-conquer algorithm. Here is a high level description:

```
Algorithm  MergeSort( S )                         § John von Neumann [1945]
Pre-Condition:    input S is a sequence of numbers
Post-Condition:   output is a sorted permutation of the input sequence
Base:         if |S| ≤ 1 then return S
Divide:       Divide S into its left and right halves S=⟨L, R⟩, |L|≈|R|≈ ½|S|
Conquer:      L' ← MergeSort( L );     R' ← MergeSort( R )
Combine:      S' ← Merge( L', R' )
Output:       return S'
end
```

MergeSort Example

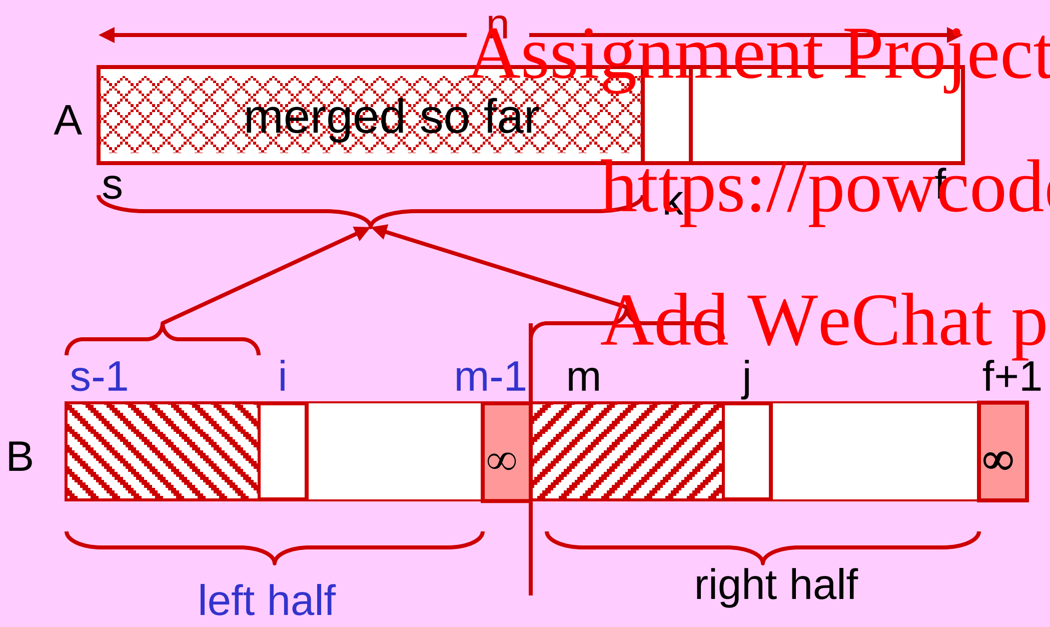


```
Algorithm MergeSort( A[1..n] )
    B[0..n+1] ← auxiliary array for merging
    MS(1,n)
end
```

```
procedure MS( s, f )   § sort A[s..f]
    Base:      if s ≥ f then return
    Divide:    m ← ⌈(s+f)/2⌉
    Conquer:   MS( s, m –1 ) ; MS( m, f )
    Combine:   Merge( s, m, f )
end
```

```
procedure Merge( s, m, f )
    B[s-1 .. m-2] ← A[s,m-1]
    B[m .. f] ← A[m,f]
    B[m-1] ← B[f+1] ← ∞   § barrier sentinel
    i ← s-1 ; j ← m
    for k ← s .. f do
        if B[i] ≤ B[j]
            then A[k] ← B[i]; i++
            else A[k] ← B[j]; j++
end
```

MERGE Loop Invariant:



Merge time = $\Theta(n)$, where $n = f – s + 1$.

```
Algorithm MergeSort( A[1..n] )
B[0..n+1] ← auxiliary array for merging
MS(1,n)
end
```

MergeSort Recurrence:

$T(n) = \Theta(1) \quad \forall n \leq 1$

$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) \quad \forall n > 1$

```
procedure MS( s, f )   § sort A[s..f]
Base:      if s ≥ f then return
Divide:    m ← ⌈(s+f)/2⌉
Conquer:   MS( s, m –1 ) ; MS( m, f )
Combine:   Merge( s, m, f )
end
```

Simplified Recurrence:

$$T(n) = 2\,T(n/2) + \Theta(n) \quad \text{for } n > 1$$
$$T(n) = \Theta(1) \quad \text{for } n \leq 1$$

Solution:

$$T(n) = \Theta(n \log n).$$

```
procedure Merge( s, m, f )
B[s-1 .. m-2] ← A[s,m-1]
B[m .. f] ← A[m,f]
B[m-1] ← B[f+1] ← ∞   § barrier sentinel
i ← s-1 ; j ← m
for k ← s .. f do
    if B[i] ≤ B[j]
        then A[k] ← B[i]; i++
        else A[k] ← B[j]; j++
end
```

Divide-&-Conquer Recurrence

$$T(n) = \begin{cases} aT\left(\frac{n}{b}\right) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \leq 1 \end{cases}$$

a = # of recursive calls (a > 0)

n/b = size of each sub-instance called recursively (b > 1)

f(n) = cost of divide + combine steps (every thing except the recursive calls).

In MergeSort we have:

a = 2 recursive calls

of size **n/b = n/2** each.

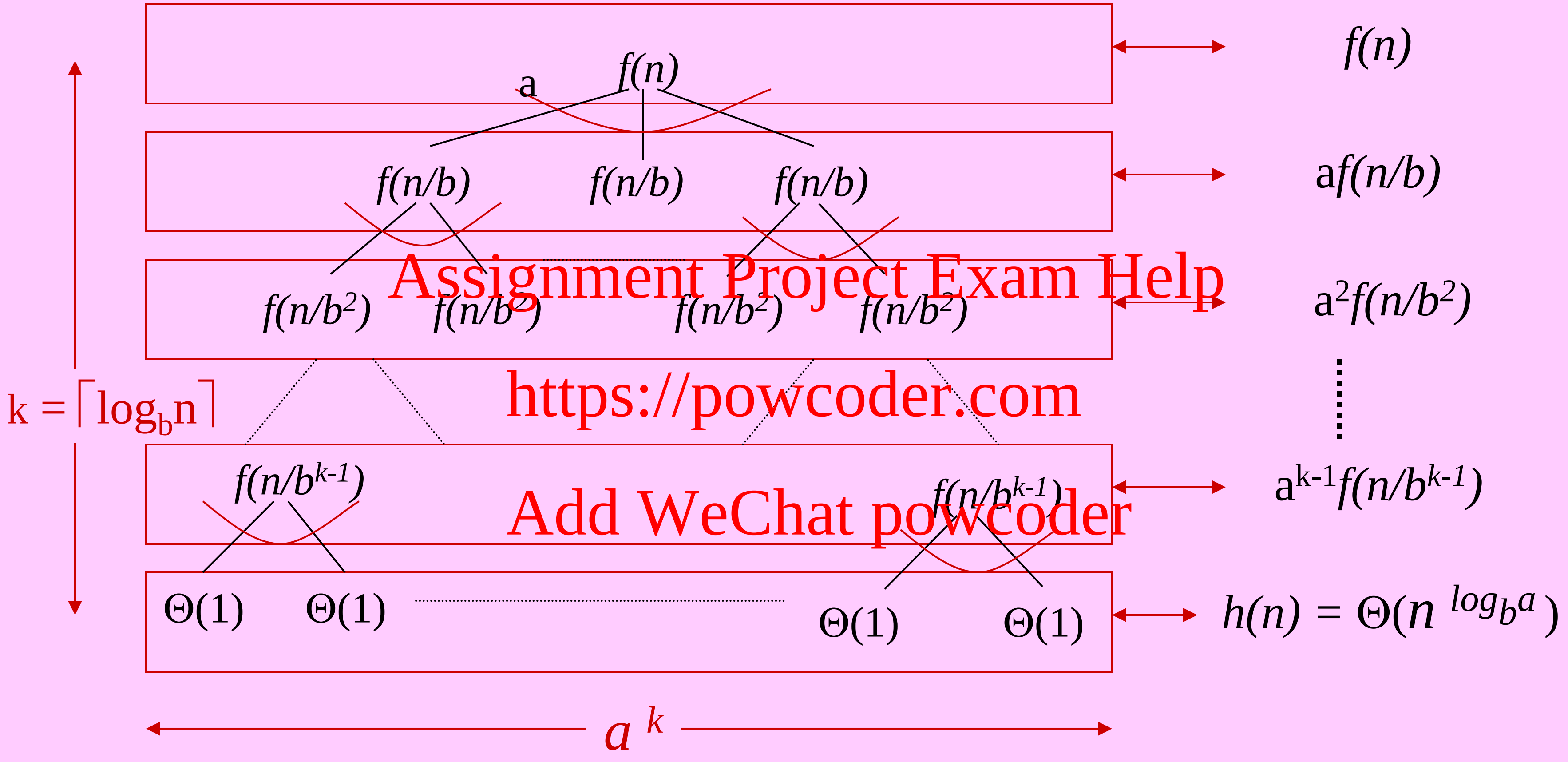
Cost of divide = Θ(1)

Cost of Combine, i.e., merge, is Θ(n).

So, **f(n) = Θ(n).**

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \le 1 \end{cases}$$

Recursion Tree:



$a^{\log_b n} = n^{\log_b a}$ (compare their logarithms)

$h(n) = a^k \cdot \Theta(1) = \Theta(a^k) = \Theta(a^{\lceil \log_b n \rceil}) = \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$

$k = \lceil \log_b n \rceil = \Theta(\log n).$

$$T(n) = \begin{cases} aT(\tfrac{n}{b}) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \le 1 \end{cases}$$

$T(n) = S(n) + h(n) = \Theta(\max\{S(n), h(n)\})$

$h(n) = \Theta(n^{\log_b a})$

$$S(n) = \sum_{i=0}^{k-1} a^i f\left(\tfrac{n}{b^i}\right)$$

$$= f(n) + af(\tfrac{n}{b}) + a^2 f(\tfrac{n}{b^2}) + \cdots + a^{k-1} f\left(\tfrac{n}{b^{k-1}}\right)$$

$S(n) = \Sigma$

$\boxed{f(n)}$

$\boxed{af(n/b)}$

$\boxed{a^2 f(n/b^2)}$

$\vdots$

$\boxed{a^{k-1} f(n/b^{k-1})}$

$\boxed{h(n) = \Theta(n^{\log_b a})}$

$a^{\log_b n} = n^{\log_b a}$ (compare their logarithms)

$h(n) = a^k \cdot \Theta(1) = \Theta(a^k) = \Theta(a^{\lceil \log_b n \rceil}) = \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$

$k = \lceil \log_b n \rceil = \Theta(\log n)$.

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \le 1 \end{cases}$$

$$T(n) = S(n) + h(n)$$

Special Solution: contribution of the internal nodes of the recursion tree.

$$S(n) = \begin{cases} aS(\frac{n}{b}) + f(n) & \forall n > 1 \\ 0 & \forall n \le 1 \end{cases}$$

Homogeneous Solution: contribution of the leaves of the recursion tree.

$$h(n) = \begin{cases} ah(\frac{n}{b}) & \forall n > 1 \\ \Theta(1) & \forall n \le 1 \end{cases}$$

$$h(n) = \Theta(n^{\log_b a})$$

Define: $Q(n) = \frac{S(n)}{h(n)}, \quad r(n) = \frac{f(n)}{h(n)}$

$$T(n) = h(n)(\, Q(n) + 1\,)$$

$$Q(n) = \begin{cases} Q(\frac{n}{b}) + r(n) & \forall n > 1 \\ 0 & \forall n \le 1 \end{cases}$$

$$Q(n) = r(n) + r\left(\frac{n}{b}\right) + r\left(\frac{n}{b^2}\right) + r\left(\frac{n}{b^3}\right) + \cdots$$

f(n) is a SAF $\Rightarrow$ r(n) is a SAF

$$Q(n) = \frac{S(n)}{h(n)}, \qquad r(n) = \frac{f(n)}{h(n)}$$

$$T(n) = h(n)(\ Q(n) + 1\ )$$

f(n) is a SAF $\Rightarrow$ r(n) is a SAF

$\hookrightarrow$ Let $n = b^k$ for integer $k = \Theta(\log n)$.

$$Q(n) = \Theta\left(\sum_{i=1}^{k} r\left(b^i\right)\right)$$

$r(n)$	$r(b^i)$	$Q(n)=\Theta(\ ?\ )$	$T(n)=\Theta(\ ?\ )$
$\Omega(n^{+\varepsilon})$	$\Omega(b^{+i\varepsilon})$	$r(n)$	$f(n)$
$O(n^{-\varepsilon})$	$O(b^{-i\varepsilon})$	1	$h(n)$
$\Theta(\log^e n)$ $e > -1$	$\Theta(i^e)$	$r(n) \cdot k$	$f(n) \cdot \log n$
$\Theta(\log^{-1} n)$	$\Theta(1/i)$	$\log k$	$h(n) \cdot \log\log n$
$O(\log^e n)$ $e < -1$	$O(i^{-e})$	1	$h(n)$

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \leq 1 \end{cases}$$

$$h(n) = \Theta(n^{\log_b a})$$

THE MASTER THEOREM

Suppose f(n) is a SAF, and ε > 0 & e are constant reals.

$f(n) \,/\, h(n)$		$T(n)$
$\Omega(n^{+\varepsilon})$ Compare with CLRS. See Exercise 4.	f(n) » h(n)	$\Theta(f(n))$
$\Theta(\log^e n)$ (e > -1)	f(n) ≈ h(n)	$\Theta(f(n)\log n) = \Theta(n^{\log_b a}\log^{e+1} n)$
$\Theta(\log^e n)$ (e = -1)		$\Theta(h(n)\log\log n) = \Theta(n^{\log_b a}\log\log n)$
$O(\log^e n)$ (e < -1)	f(n) « h(n)	$\Theta(h(n)) = \Theta(n^{\log_b a})$
$O(n^{-\varepsilon})$		

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \le 1 \end{cases}$$

$$h(n) = \Theta(n^{\log_b a})$$

$f(n) = \Theta(t^n n^d \log^e n)$ constants $t > 0$, d, e.			
$t > 1$			$T(n) = \Theta(f(n))$
$t = 1$	$b^d > a$		$T(n) = \Theta(f(n))$
	$b^d = a \Leftrightarrow d = \log_b a$	$e > -1$	$T(n) = \Theta(f(n) \log n)$
		$e = -1$	$T(n) = \Theta(h(n) \log \log n)$
		$e < -1$	$T(n) = \Theta(h(n))$
	$b^d < a$		$T(n) = \Theta(h(n))$
$0 < t < 1$			$T(n) = \Theta(h(n))$

EECS 3101 Sums & Recurrences Fact Sheet

$\log xy = \log x + \log y$

$\log x^y = y \log x$

$\log_y x = \log x / \log y$

$\log(1 \pm x) = \pm\Theta(x)$ for $x = o(1)$

$$\sum_{i=1}^{n} i^d = \begin{cases} \Theta(n^{d+1}) & \text{if } d > -1 \\ \Theta(\log n) & \text{if } d = -1 \\ \Theta(1) & \text{if } d < -1 \end{cases}$$

$$\sum_{i=0}^{n} x^i = \begin{cases} \dfrac{x^{n+1}-1}{x-1} & \text{if } 0 \neq |x| \neq 1 \\ \Theta(x^n) & \text{if } x > 1 \\ \Theta(1) & \text{if } 0 < x < 1 \end{cases}$$

For $f\uparrow$: $\displaystyle\int_{m-1}^{n} f(x)dx \le \sum_{i=m}^{n} f(i) \le \int_{m}^{n+1} f(x)dx$

For $f\downarrow$: $\displaystyle\int_{m-1}^{n} f(x)dx \ge \sum_{i=m}^{n} f(i) \ge \int_{m}^{n+1} f(x)dx$

ASSUME [ASymptotic SUmmation Made Easy]

$S(n) = \sum_{i=1}^{n} f(i)$ for $f(n)\uparrow = \omega(1)$

$g(n) \leftarrow [n-m]$ where $\log f(m) = \log f(n) - \Theta(1)$:

$\mathbf{f\uparrow}$ **and** $\mathbf{g\uparrow} \Rightarrow \mathbf{S(n) = \Theta(\, g(n)\, f(n)\,)}$.

Sum Type	$f(n)$: a SAF	$S(n) = \sum_{i=1}^{n} f(i)$
Geometric	$2^{\Omega(n)}$	$\Theta(\, f(n)\,)$
Arithmetic	$n^{\Theta(1)-1}$	$\Theta(\, n\, f(n)\,)$
Harmonic	$\Theta(n^{-1}\log^e n)$, e = a real constant	$\Theta(\log^{e+1} n)$ if $e > -1$ $\Theta(\log\log n)$ if $e = -1$ $\Theta(1)$ if $e < -1$
Bounded Tail	$n^{-\Omega(1)-1}$ or $even$ $O(n^{-1}\log^e n)$, $e < -1$	$\Theta(1)$

MASTER METHOD: $T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \le 1 \end{cases}$, $h(n) = \Theta(n^{\log_b a})$, $f(n)$: **a SAF**, constants $\varepsilon > 0$ & e.

$f(n) : h(n)$	$\dfrac{f(n)}{h(n)} =$	$T(n) =$
$f(n) \gg h(n)$	$\Omega(n^\varepsilon)$	$\Theta(\, f(n)\,)$
$f(n) \approx h(n)$	$\Theta(\log^e n)$, $e > -1$	$\Theta(\, f(n) \log n\,)$
	$\Theta(\log^{-1} n)$, $e = -1$	$\Theta(\, h(n) \log\log n\,)$
$f(n) \ll h(n)$	$O(n^{-\varepsilon})$ or $even$ $O(\log^e n)$, $e < -1$	$\Theta(\, h(n)\,)$

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \le 1 \end{cases}$$

$$h(n) = \Theta(n^{\log_b a})$$

Example 1: Assume $a = 2$, $b = 4$, $f(n) = \Theta(n^d \log^2 n)$.

Express T(n) based on the constant parameter d.

Solution:

$\log_b a = \log_4 2 = ½ \Rightarrow h(n) = \Theta(\sqrt{n})$

$d > ½ \Rightarrow T(n) = \Theta(f(n)) = \Theta(n^d \log^2 n)$

$d = ½ \Rightarrow T(n) = \Theta(f(n) \log n) = \Theta(\sqrt{n} \log^3 n)$

$d < ½ \Rightarrow T(n) = \Theta(h(n)) = \Theta(\sqrt{n})$

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & \forall n > 1 \\ \Theta(1) & \forall n \le 1 \end{cases}$$

$$h(n) = \Theta(n^{\log_b a})$$

Example 2: Assume $b = 3$, $f(n) = \Theta(n^4 / \log n)$.

Express T(n) based on the constant parameter a.

Solution:

$h(n) = \Theta(n^{\log_3 a})$, $b^d = 3^4 = 81$

$a < 81 \Rightarrow T(n) = \Theta(f(n)) = \Theta(n^4 / \log n)$

$a = 81 \Rightarrow T(n) = \Theta(h(n) \log\log n) = \Theta(n^4 \log\log n)$

$a > 81 \Rightarrow T(n) = \Theta(h(n)) = \Theta(n^{\log_3 a})$

Guess-&-Verify

BASIC INGRIDIANTS:

- Guess a solution to the recurrence.

- Verify it by mathematical induction.
  The induction variable must be a natural number.
  E.g., height of the recursion tree (max recursion depth), or
  size of the recursion tree (# times you apply the recurrence).
  n may also be a candidate if it "stays" integer.

- If you spot a place where the verification does not go through, examine it to help revise your guess, and try again.

- One key guiding principle, taken with a grain of salt, is to first make sure the leading term (with its constant coefficient) matches on the LHS and RHS of the recurrence. That will show you to guess higher or lower the next time! After figuring out the leading term, you can apply the same principle to find the lower order terms.

Guess-&-Verify: Example 1

$$T(n) = \begin{cases} 4T\left(\frac{n}{2}\right) + n^2 & for\ \ n > 1 \\ 1 & for\ \ n \le 1 \end{cases}$$

- Clearly $T(n) = \Omega(n^2)$. Is $T(n) = O(n^2)$?

- Guess # 1: $T(n) = cn^2 + L(n)$

const. $c > 0$, $L(n) = o(n^2)$ lower order terms (to be determined)

- Plug this guess in the recurrence and verify: $T(n)\ ?{=}?\ 4T\left(\frac{n}{2}\right) + n^2$

$$cn^2 + L(n) \quad ?{=}? \quad 4\left(c\left(\frac{n}{2}\right)^2 + L\left(\frac{n}{2}\right)\right) + n^2 \quad = \quad (c+1)n^2 + 4L\left(\frac{n}{2}\right)$$

- LHS leading term $<$ RHS leading term $\Rightarrow$ guess is too low.

Guess-&-Verify: Example 1

$$T(n) = \begin{cases} 4T\left(\frac{n}{2}\right) + n^2 & for\ \ n > 1 \\ 1 & for\ \ n \le 1 \end{cases}$$

- Guess # 2: $T(n) = cn^{2+\varepsilon} + L(n)$

  const. $c > 0,\ \varepsilon > 0,\ \ L(n)$ lower order terms (to be determined)

- Plug this guess in the recurrence and verify: $T(n)\ ?=?\ 4T\left(\frac{n}{2}\right) + n^2$

$$cn^{2+\varepsilon} + L(n)\quad ?=?\quad 4\left(c\left(\frac{n}{2}\right)^{2+\varepsilon} + L\left(\frac{n}{2}\right)\right) + n^2 \quad = \quad \frac{1}{2^{\varepsilon}}\, cn^{2+\varepsilon} + 4L\left(\frac{n}{2}\right) + n^2$$

- LHS leading term $>$ RHS leading term $\Rightarrow$ guess is too high.

Guess-&-Verify: Example 1

$$T(n) = \begin{cases} 4T\left(\frac{n}{2}\right) + n^2 & for\ \ n > 1 \\ 1 & for\ \ n \le 1 \end{cases}$$

- Guess # 3: $T(n) = cn^2 \log n + L(n)$

  const. $c > 0$, $L(n)$ lower order terms (to be determined)

- Plug this guess in the recurrence and verify: $T(n)\ ?=?\ 4T\left(\frac{n}{2}\right) + n^2$

$cn^2 \log n + L(n) \quad ?=? \quad 4\left(c\left(\frac{n}{2}\right)^2 \log\frac{n}{2} + L\left(\frac{n}{2}\right)\right) + n^2$

$= \quad cn^2 \log n + 4L\left(\frac{n}{2}\right) + (1-c)n^2$

- LHS leading term = RHS leading term $\Rightarrow$ correct guess of high order term.
- Solve the residual recurrence for $L(n)$ (with revised boundary condition):

  $L(n) = 4L\left(\frac{n}{2}\right) + (1-c)n^2$

Claim: $c = 1$. Why? If $1 - c > 0$, then $L(n)$ is not lower order.

If $1 - c < 0$, then $L(n)$ becomes negative but the leading term gets larger.

Residual recurrence: $L(n) = 4L\left(\frac{n}{2}\right)$ (plus revised boundary condition)

Solution: $\mathrm{L}(n) = O(n^2)$ and $T(n) = n^2 \log n + O(n^2) = \Theta(n^2 \log n)$.

Guess-&-Verify: Example 2

Recurrence: $T(n) = 2T(\lfloor n/2 \rfloor) + n^2$, $T(0) = 0$

$T(n) \geq n^2 \geq \Omega(n^2)$ is obvious. Let's see if $O(n^2)$ is an upper-bound:

Guess: $T(n) \leq an^2 + bn + c$ (for some constants a>0, b, c to be determined)

Basis (n = 0): $T(0) = 0 \leq a0^2 + b0 + c \longleftrightarrow \boxed{c \geq 0}$ we need

Ind. Step (n ≥1): $T(n) = 2T(\lfloor n/2 \rfloor) + n^2$

$\leq 2\left[a\lfloor n/2 \rfloor^2 + b\lfloor n/2 \rfloor + c\right] + n^2 \longleftarrow$ by induction hypothesis

if n is even $\longrightarrow$ $= 2\left[a(n/2)^2 + b(n/2) + c\right] + n^2 \longleftarrow$ not "≤" for odd n, if b<0

$= (1 + a/2)n^2 + bn + 2c$

$\leq an^2 + bn + c \longleftarrow$ our guessed upper-bound

We need: for all **even** n ≥1:

$an^2 + bn + c \geq (1 + a/2)n^2 + bn + 2c$, i.e.,

$(a/2 - 1)n^2 - c \geq 0.$

Guess-&-Verify: Example 2

Recurrence: $T(n) = 2T(\lfloor n/2 \rfloor) + n^2$, $T(0) = 0$

$T(n) \geq n^2 \geq \Omega(n^2)$ is obvious. Let's see if $O(n^2)$ is an upper-bound:

Guess: $T(n) \leq an^2 + bn + c$ (for some constants a>0, b, c to be determined)

Basis (n = 0): $T(0) = 0 \leq a0^2 + b0 + c \longleftrightarrow$ we need $\boxed{c \geq 0}$

Ind. Step (n ≥1): $T(n) = 2T(\lfloor n/2 \rfloor) + n^2$

$\leq 2\left[a\lfloor n/2 \rfloor^2 + b\lfloor n/2 \rfloor + c\right] + n^2 \longleftarrow$ by induction hypothesis

if n is odd $\longrightarrow$ $= 2\left[a((n-1)/2)^2 + b((n-1)/2)+c\right] + n^2$

$= (1+ a/2)n^2 + (b-a)n + (2c-b+a/2)$

$\leq an^2 + bn + c \longleftarrow$ our guessed upper-bound

We need: for all **odd** n ≥1:
$an^2 + bn + c \geq (1+ a/2)n^2 + (b-a)n + (2c-b+a/2)$, i.e.,
$(a/2 - 1)n^2 + an + (b - c - a/2) \geq 0.$

We need: for all **even** n ≥1:
$an^2 + bn + c \geq (1+ a/2)n^2 + bn + 2c$, i.e.,
$(a/2 - 1)n^2 - c \geq 0.$

a = 2
b = -1
c = 0
works for all n ≥ 0.

$$n^2 \leq T(n) \leq 2n^2 - n$$

Guess-&-Verify: Example 2

Recurrence: $T(n) = 2T(\lfloor n/2 \rfloor) + n^2$, $T(0) = 0$

Exercise 8: Using this same method, verify the following **tighter LOWER BOUND:**

$$\forall n \geq 1: \quad 2n^2 - n - 2n \log n \leq T(n) \leq 2n^2 - n.$$

$$\Downarrow$$

$$T(n) = 2n^2 - n - O(n \log n).$$

For more examples of guess-&-verify see
Lecture Note 3
&
Sample Solutions ….

a = 2
b = -1
c = 0
works for all $n \geq 0$.

$$n^2 \leq T(n) \leq 2n^2 - n$$

Full History Recurrence

Such recurrences often arise in average-case analysis.

$$\text{Example 1: } T(n) = \sum_{i=0}^{n-1} T(i) + f(n), \qquad \forall n \geq 0$$

Values of T(n) for some small n:

T(0) = f(0)

T(1) = T(0) + f(1) = f(0) + f(1)

T(2) = T(0) + T(1) + f(2) = 2f(0) + f(1) + f(2)

T(3) = T(0) + T(1) + T(2) + f(3) = 4f(0) + 2f(1) + f(2)+ f(3)

T(4) = T(0) + T(1) + T(2) + T(3) + f(4) = 8f(0) + 4f(1) + 2f(2) + f(3) + f(4)

$$\text{Example 2: } T(n) = \frac{2}{n} \sum_{i=0}^{n-1} T(i) + n, \qquad \forall n \geq 0$$

This is the QuickSort expected time recurrence (details shown later in LS5)

Full History Recurrence

Example 1: $T(n) = \sum_{i=0}^{n-1} T(i) + f(n), \quad \forall n \ge 0$

A Solution Method:

$n \leftarrow n\text{-}1$: $\quad T(n-1) = \sum_{i=0}^{n-2} T(i) + f(n-1), \quad \forall n-1 \ge 0 \quad (\text{i.e., } \forall n \ge 1)$

Subtract: $\quad T(n) - T(n-1) = T(n-1) + f(n) - f(n-1), \quad \forall n \ge 1$

Rearrange: $\quad T(n) = \begin{cases} 2T(n-1) + \left[f(n) - f(n-1)\right] & \forall n \ge 1 \\ f(0) & \text{for } n = 0 \end{cases}$

Now there is only one appearance of T(.) on the RHS.
Continue with conventional methods.

Variable Substitution: Example 1

Sometimes we can considerably simplify the recurrence by a change of variable.

$T(n) = T(n/2) + \log n$

[assume the usual boundary condition T(O(1))=O(1).]

Change of variable: $n = 2^m$, i.e., $m = \log n$.

Now $T(n) = T(2^m)$. It is some function of m.
Rename it S(m). That is, $T(n) = T(2^m) = S(m)$.

The recurrence becomes:

$S(m) = S(m-1) + m$.

By the iteration method: $S(m) = \Theta(m^2)$.
So, $T(n) = T(2^m) = S(m) = \Theta(m^2) = \Theta(\log^2 n)$.

$T(n) = \Theta(\log^2 n)$.

Variable Substitution: How?

Recurrence | Boundary Condition

$$T(n) = T(n/2) + f(n) \qquad T(n) = 0 \text{ for } n < 1$$

$$\underset{g(m)}{\uparrow} \qquad \underset{g(m-1)}{\uparrow} \qquad \underset{g(m)}{\uparrow}$$

Rename: $\underbrace{n = g(m),\ n/2 = g(m-1)},\ S(m) = T(g(m)),\ h(m) = f(g(m))$

Now solve 2 recurrences:

1) $g(m) = 2g(m-1),$ $\Rightarrow$ $g(m) = 2^m g(0) = 2^m$.
$g(0) = 1$ $\leftarrow$ our choice

2) $S(m) = S(m-1) + h(m)$ $\Rightarrow$ $S(m) = h(0)+h(1)+ \cdots +h(m)$.
$S(0) = T(g(0)) = T(1) = f(1) = h(0)$

Now back substitute:

$T(n) = T(g(m)) = S(m) = S(g^{-1}(n))$,
where g^{-1} is functional inverse of g.

See another example on the next page

Variable Substitution: Example 2

$T(n) = T(\sqrt{n}) + 1$

$\uparrow g(m) \quad \uparrow g(m-1)$

Rename: $n = g(m)$, $\sqrt{n} = g(m-1)$, $S(m) = T(g(m))$.

$$g(m) = g(m-1)^2 = g(m-2)^{2^2} = g(m-3)^{2^3} = \cdots = g(0)^{2^m}$$

$g(0) = 2 \Rightarrow n = g(m) = 2^{2^m} \Rightarrow m = \log \log n.$

$S(m) = S(m-1)+1 \Rightarrow S(m) = \Theta(m) = \Theta(\log \log n).$

$S(0) = T(2) = \Theta(1)$

$\downarrow$

T(n) = $\Theta(\log \log n)$.

Exercises

1. For each pseudo-code below derive the simplified asymptotic running time in Θ(?) notation.

(a)
```
for i ← 1 .. n do
    for j ← 1 .. 2*i do print( i+j )
```

(b)
```
for i ← 1 .. n do
    for j ← 2*i .. n do print( i+j )
```

(c)
```
for i ← 1 .. n do
    j ← 1
    while j ≤ i do j ← j+3
```

(d)
```
for i ← 1 .. n do
    j ← 1
    while i+j ≤ n do j ← j+3
```

(e)
```
for i ← 1 .. n do
    j ← 1
    while j ≤ i do j ← i+j
```

(f)
```
for i ← 1 .. n do
    j ← 1
    while j*j ≤ i do j ← j+1
```

(g)
```
for i ← 1 .. n do
    j ← 2
    while j ≤ n do j ← j*j
```

(h)
```
for i ← 1 .. n do
    j ← 2
    while j ≤ i do j ← j*j
```

(i)
```
for i ← 1 .. n do
    j ← 1
    while i*j ≤ n do j ← j+1
```

(j)
```
for i ← 1 .. n do
    j ← 1
    while i*i ≥ j do j ← j+1
```

(k)
```
for i ← 1 .. n do
    j ← n
    while i < j*j do j ← j – 2
```

(l)
```
for i ← 1 .. n do
    j ← n
    while i ≤ j*j do j ← j div 2
```

2. ASSUME is a method to find $\Theta(?)$ asymptotic value of $S(n) = f(1) + f(2) + \cdots + f(n)$.
We described the method when f(n) is monotonically increasing.
Describe the method when f(n) is monotonically decreasing.

3. Let $S(n) = f(1) + f(2) + \cdots + f(n)$.
For each f(n) below, derive the answer to the question $S(n) = \Theta(?)$.
Simplify your answer as much as you can.

(a) $f(n) = 1/(2n+3)$ (b) $f(n) = n^2 \log^3 n$

(c) $f(n) = 2^n \log^3 n$ (d) $f(n) = 2^n n^5 \log^3 n$

(e) $f(n) = 2^n / \log n$ (f) $f(n) = 2^{\sqrt{n \log n}}$

(g) $f(n) = n^{\sqrt{n}}$ (h) $f(n) = (\log n)^n$

(i) $f(n) = n^{\log n}$ (j) $f(n) = n^{\log \log n}$

(k) $f(n) = (1 + 1/n)^{n^2}$ (l) $f(n) = (1 + 1/n)^{n^{1.5}}$

(m) Consider any super-polynomial sub-exponential function $f(n) = 2^{n/g(n)}$,
where $g(n)$ is a SAF and $g(n) \in \omega(1) \cap o(n)$.
Under what extra conditions, if any, do we have $S(n) = \Theta(g(n) f(n))$?

4. The Master Theorem: CLRS versus these Slides:

The case "$f(n)/h(n) = \Omega(n^{+\varepsilon})$" of the Master Theorem [page 94 of CLRS] has the following extra condition:

(*) *if $af(n/b) < cf(n)$ for some constant $c < 1$ and all sufficiently large n.*

The same theorem on page 40 of these slides instead uses the condition:

(**) *$f(n)$ is a SAF.*

Show that (**) implies (*) in the case "$f(n)/h(n) = \Omega(n^{+\varepsilon})$" .

[Hint: Consider the SAF $r(n) = f(n)/h(n) = \Omega(n^{+\varepsilon})$. So, $r(b^i) = \Omega(b^{+i\varepsilon})$ is increasing at least exponentially as a function of i. Then, with some extra derivation, show that $r(n/b) / r(n) \le c < 1$ for some constant c and all sufficiently large n. $c = b^{-\varepsilon/2}$ works.]

5. Exercise 6 of Lecture Note 2 (page 9) solves the recurrence $T(n) = 3T(n/3 +5) + n/2$ by the guess-&-verify method. Such recurrences can also be solved by the variable substitution method. Consider the somewhat generalized recurrence $T(n) = a\,T(n/b + c) + f(n)$, where $a>0$, $b>1$, and c are constants, and $f(n)$ is the driving function which is a SAF.

a) Show the substitution $S(n) = T(n + cb/(b-1))$ results in the new recurrence $S(n) = aS(n/b) + f(n + cb/(b-1))$. [Now this can be solved by the Master Theorem.]

b) Use this method to solve the recurrence $T(n) = 4T(n/2 +7) + n^2$.

6. Consider the following recurrences with the boundary condition T(O(1)) = Θ(1). Solve these recurrences by your specified method of choice. Express your answers based on the possible range of values for the constants a, d, e.

(a) $T(n) = a\,T(n/3) + n^3 \log^e n$

(b) $T(n) = 16\,T(n/4) + n^d \log^e n$

(c) $T(n) = 8\,T(n/4) + n^d \log^e n$

(d) $T(n) = 4\,T(n/4+3) + n \log n$

(f) $T(n) = T(n/2) + T(n/5) + n/\log n$

(g) $T(n) = T(n/2) + T(n/4) + T(n/5) + n$

(h) $T(n) = T(n/2 + 5) + n$

(i) $T(n) = T(n/2 + \sqrt{n}\,) + n$

(j) $T(n) = T(n/2) + T(n/5) + n$

(k) $T(n) = T(n/2) + T(n/5) + \log n$

(l) $T(n) = T(n/2) + T(n/5) + \sqrt{n}$

[Caution: T(n) = S(n) + h(n). Which one dominates? S(n) or h(n)?]

(m) $T(n) = \sum_{i=0}^{n-1} i \cdot T(i) + n$

7. Solve the following recurrences by your specified method of choice.
Where not explicitly stated, assume the boundary condition: T(n) = 0 for n < 2, and the recurrence holds for all n ≥ 2.

(a) $T(n) = (T(n-1))^2 + 1$, for $n > 0$, and $T(0) = 0$.

(b) $T(n) = (T(n-1))^2 + 1$, for $n > 0$, and $T(0) = 2$.

(c) $T(n) = (T(n-1))^2 + 1$, for $n > 0$, and $T(0) = 3$.

(d) $T(n) = T(\log n) + 1$.

(e) $T(n) = n\,T(n-1) + 1$.

(f) $T(n) = n\,T(n/2) + 1$.

(g) $T(n) = 2T(\sqrt{n}) + 1$.

(h) $T(n) = 2T(\sqrt{n}) + \log n$.

(i) $T(n) = 2T(\sqrt{n}) + n$.

(j) $T(n) = \max\{\, T(k) + T(n-k) + n \mid 0 < k \le n/2 \,\}$.

(k) $T(n) = \max\{\, T(k) + T(n-k) + k \mid 0 < k \le n/2 \,\}$.

(l) $T(n) = \min\{\, T(k) + T(n-k) + n \mid 0 < k \le n/2 \,\}$.

(m) $T(n) = \min\{\, T(k) + T(n-k) + k \mid 0 < k \le n/2 \,\}$.

(n) $T(n) = \frac{n}{n-1} T(n-1) + 1$

(o) $T(n) = \frac{n-1}{n} T(n-1) + 1$

8. Show that the recurrence $$\begin{aligned} T(n) &= 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n^2 , \\ T(0) &= 0 \end{aligned}$$

has the solution: $T(n) = 2n^2 - n - O(n \log n)$.

9. Show that the recurrence

$$T(n) = \begin{cases} T(\alpha n) + T(\beta n) + n & for\ n \geq n_o \\ c & for\ 0 \leq n < n_o \end{cases}$$

for any real constants $0 \le \alpha < 1$, $0 \le \beta < 1$, $c > 0$, $n_o > 0$,
has the following solutions:

a) $\alpha + \beta < 1 \quad \Rightarrow \quad T(n) = \Theta(n)$

b) $\alpha + \beta = 1 \quad \Rightarrow \quad T(n) = \Theta(n \log n)$

c) $\alpha + \beta > 1 \quad \Rightarrow \quad T(n) = \Theta(n^d)$, where d > 1 is the unique constant that satisfies the equation: $\alpha^d + \beta^d = 1$.

10. Towers of Hanoi with sufficiently many stacks:

a) Using the generalized recursive algorithm GTH, show that $T_k(n) = \Theta(n)$ for all $k \geq 2+(n-1)/2$. [Hint: set $m := n - 2$ in the algorithm.]

b) Describe the iterative version of the recursive algorithm in part (a).

c) Generalize part (a) by showing that $T_k(n) = \Theta(n)$ for all $k \geq 2+(n-1)/c$ for any positive constant c.

d) Show that $T_k(n) = \Theta(n)$ for all $k \geq 1 + \sqrt{n}$ (i.e., $n \leq (k-1)^2$).

e) Show that $T_k(n) = \Theta(n)$ for all $n \leq (k-1)^c$, where c is any constant.

[This exercise is repeated from Lecture Slide 1.]

11. Compare efficiency of two algorithms:

We have two divide-&-conquer algorithms, A and B, that solve the same computational problem. Their running times for inputs of size n are denoted by $T_A(n)$ and $T_B(n)$ respectively, and expressed by their recurrences shown below.

As usual assume the boundary condition $T(O(1)) = \Theta(1)$ for both recurrences.

Assuming the constant parameter k can be any nonnegative integer, find the range of values of this parameter for which algorithm A is asymptotically at least as efficient as algorithm B.

$$T_A(n) = k \cdot T_A(n/5) + \Theta(n^3 \log^{k-1} n)$$

$$T_B(n) = 81 \cdot T_B(n/3) + \Theta(n^4 \log^k n).$$

12. Derive the most simplified answers to the following summations. Mention which methods you used to derive your solution.

a) $\sum_{i=1}^{\lceil \log n \rceil} \left(2i^{\sqrt{i}} + 7i^2 5^i + 4i^6 \log i \right) = \Theta(?)$.

b) $\sum_{i=1}^{n} \sum_{j=1}^{2^i} \frac{(2j+7)^5}{(3i+4)\left(8^i+3i^5\right)(9j-1)^3} = \Theta(?)$.

c) $\sum_{i=1}^{2^n} \sum_{j=1}^{n} \frac{2^i+j}{j^4} = \Theta(?)$.

d) $\sum_{i=1}^{\sqrt{n}} \sum_{j=1}^{i^4} \frac{(i^2+j)^4}{\left(2i \log i + 5\sqrt{j}\right)^2} = \Theta(?)$.

e) $\sum_{i=1}^{n^2} \sum_{j=1}^{\lceil \log i \rceil} \frac{7^{3j}+5\,j^2}{2j^3+9j \log j} = \Theta(?)$.

f) $\sum_{i=1}^{n} \sum_{j=1}^{i \lceil \log i \rceil} \frac{i \log j}{j+\log i} = \Theta(?)$

13. Recursion Time Analysis:

A certain recursive algorithm takes an input list of n elements. Divides the list into $\sqrt{n}$ sub-lists, each with $\sqrt{n}$ elements. Recursively solves each of these $\sqrt{n}$ smaller sub-instances. Then spends an additional $\Theta(n)$ time to combine the solutions of these sub-instances to obtain the solution of the main instance.

As a base case, if the size of the input list is at most a specified positive constant, then the algorithm solves such a small instance directly in $\Theta(1)$ time.

a) Express the recurrence relation that governs T(n), the time complexity of this algorithm.

b) Derive the solution to this recurrence relation: $T(n) = \Theta(?)$.
Mention which methods you used to derive your solution.

14. The expected-case running time of a particular algorithm is given by the following recurrence relation (with the usual boundary condition $T(O(1)) = \Theta(1)$).
Show that the solution to this recurrence is $T(n) = \Theta(n \log n)$.

$$T(n) = \frac{1}{2}\left(T\left(\frac{3n}{4}\right) + T\left(\frac{n}{4}\right)\right) + \frac{1}{2}T(n-1) + \Theta(n).$$

[Akra-Bazzi] A Generalization of the Master Theorem [See LN3a for details]:

Consider the following (continuous) recurrence

$$T(x) = \sum_{i=1}^{k} a_i T(b_i x + \varepsilon_i(x)) \quad + \quad f(x)$$

where k is fixed, $a_i > 0$, $0 < b_i < 1$, $|\varepsilon_i(x)| = O\left(\frac{x}{\log^2 x}\right)$, $f(x)$ has polynomial growth rate.

Theorem [Akra-Bazzi, 1996] The asymptotic solution to the above recurrence is

$$T(x) = \Theta\left(x^p \left(1 + \int_1^x \frac{f(u)}{u^{1+p}}\, du \right)\right)$$

where p is the unique real number such that $\sum_{i=1}^{k} a_i b_i^p = 1$.

15. Use Akra-Bazzi Theorem to show (or answer) the following:

a) $T(x) = 2T(x/4) + 3T(x/6) + \Theta(x \log x) \implies p = 1$ and $T(x) = \Theta(x \log^2 x)$.

b) $T(x) = 2T(x/2) + \frac{8}{9}T(3x/4) + \Theta(x^2/\log x) \implies p = 2$ and $T(x) = \Theta(x^2 / \log\log x)$.

c) $T(x) = 2T(x/2) + \Theta(\log x) \implies p = 0$ and $T(x) = \Theta(\log^2 x)$.

d) $T(x) = \frac{1}{2}T(x/2) + \Theta(1/x) \implies p = -1$ and $T(x) = \Theta((\log x)/x)$.

e) $T(x) = 4T(x/2) + \Theta(x) \implies p = 2$ and $T(x) = \Theta(x^2)$.

f) $T(x) = 2T(x/4 + 5\sqrt{x}) + 3T(x/6 - 7\log x) + \Theta(x^2 \log x) \implies p = ???$ and $T(x) = \Theta(???)$.

END