Algorithm Analysis, PMI, Asymptotics, and Recurrences
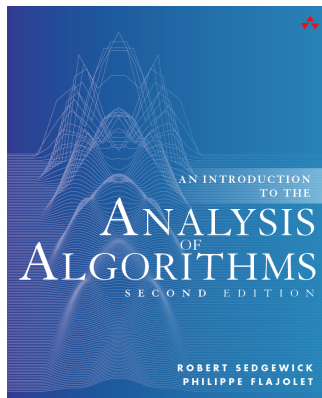## (A Little Mathematics for Computer Science)

Hengfeng Wei

hfwei@nju.edu.cn

April 17, 2018

AN INTRODUCTION
TO THE

ANALYSIS
OF
ALGORITHMS

SECOND EDITION

ROBERT SEDGEWICK
PHILIPPE FLAJOLET

Problem P Algorithm A

Problem P Algorithm A

Inputs: $\mathcal{X}_n$ of size n

Problem P Algorithm A

Inputs: $\mathcal{X}_n$ of size n

$$W(n) = \max_{X \in \mathcal{X}_n} T(X)$$

Problem P Algorithm A

Inputs: $\mathcal{X}_n$ of size n

$$W(n) = \max_{X \in \mathcal{X}_n} T(X)$$

$$B(n) = \min_{X \in \mathcal{X}_n} T(X)$$

Problem P Algorithm A

Inputs: $\mathcal{X}_n$ of size n

$$W(n) = \max_{X \in \mathcal{X}_n} T(X)$$

$$B(n) = \min_{X \in \mathcal{X}_n} T(X)$$

$$A(n) = \boxed{\sum_{X \in \mathcal{X}_n} T(X) \cdot P(X)}$$

Problem P $\qquad$ Algorithm A

Inputs: $\mathcal{X}_n$ of size n

$$W(n) = \max_{X \in \mathcal{X}_n} T(X)$$

$$B(n) = \min_{X \in \mathcal{X}_n} T(X)$$

$$A(n) = \boxed{\sum_{X \in \mathcal{X}_n} T(X) \cdot P(X)} = \mathop{\mathbb{E}}_{X \in \mathcal{X}_n} [T(X)]$$

Average-case Time Complexity (Problem 1.8)

$$r \in [1, n], \ r \in \mathbb{Z}^+$$

$$P\{r = i\} = \begin{cases} \frac{1}{n}, & 1 \leq i \leq \frac{n}{4} \\ \frac{2}{n}, & \frac{n}{4} < i \leq \frac{n}{2} \\ \frac{1}{2n}, & \frac{n}{2} < i \leq n \end{cases} \qquad T(r) = \begin{cases} 10, & r \leq \frac{n}{4} \\ 20, & \frac{n}{4} < r \leq \frac{n}{2} \\ 30, & \frac{n}{2} < r \leq \frac{3n}{4} \\ n, & \frac{3n}{4} < r \leq n \end{cases}$$

Average-case Time Complexity (Problem 1.8)

$$r \in [1, n], \ r \in \mathbb{Z}^+$$

$$P\{r = i\} = \begin{cases} \frac{1}{n}, & 1 \le i \le \frac{n}{4} \\ \frac{2}{n}, & \frac{n}{4} < i \le \frac{n}{2} \\ \frac{1}{2n}, & \frac{n}{2} < i \le n \end{cases} \qquad T(r) = \begin{cases} 10, & r \le \frac{n}{4} \\ 20, & \frac{n}{4} < r \le \frac{n}{2} \\ 30, & \frac{n}{2} < r \le \frac{3n}{4} \\ n, & \frac{3n}{4} < r \le n \end{cases}$$
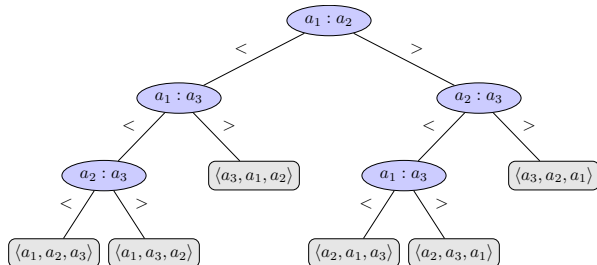
$$\begin{aligned} A &= \sum_{X \in \mathcal{X}} T(X) \cdot P(X) \\ &= T(1)P(1) + T(2)P(2) + \cdots + T(n)P(n) \\ &= \frac{n}{4} \times 10 \times \frac{1}{n} + \frac{n}{4} \times 20 \times \frac{2}{n} + \frac{n}{4} \times 30 \times \frac{1}{2n} + \frac{n}{4} \times n \times \frac{1}{2n} \\ &= \cdots \end{aligned}$$

3-element Sorting (Problem 1.1)

(1) Design an algorithm for sorting 3 distinct elements.

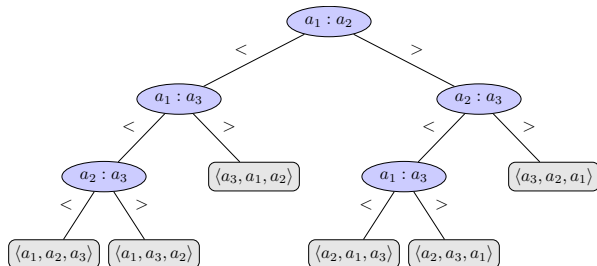(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

3-element Sorting (Problem 1.1)

(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.
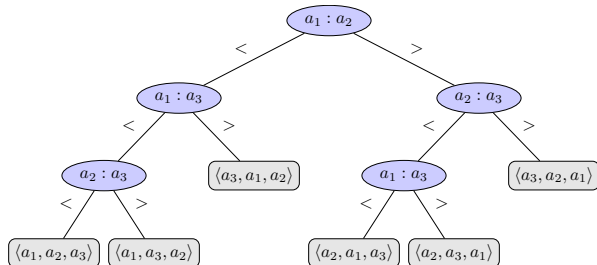
(3) Worst-case lower bound.

$W(3) =$

(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

$$W(3) = 3$$

(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

$$W(3) = 3 \qquad B(3) =$$
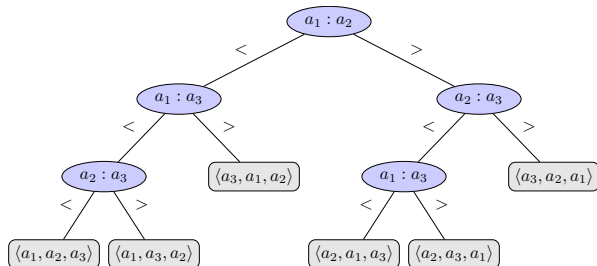
(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

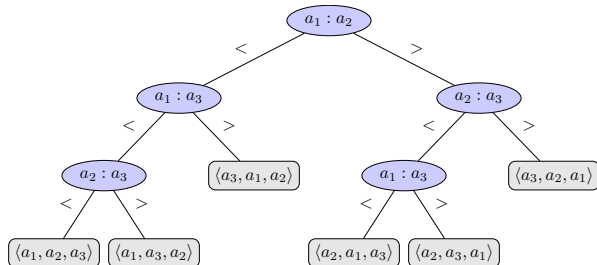$$W(3) = 3 \qquad B(3) = 2$$

3-element Sorting (Problem 1.1)

(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) =$$

3-element Sorting (Problem 1.1)

(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{1}{6}(3 + 3 + 2 + 3 + 3 + 2) = \frac{8}{3}$$
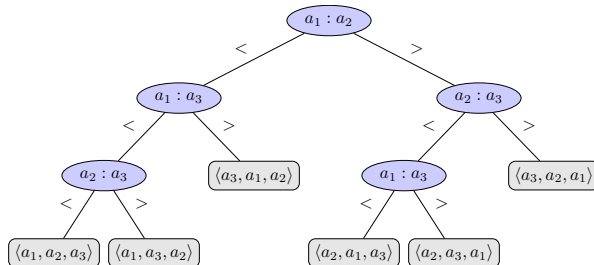
3-element Sorting (Problem 1.1)

(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{1}{6}(3 + 3 + 2 + 3 + 3 + 2) = \frac{8}{3}$$
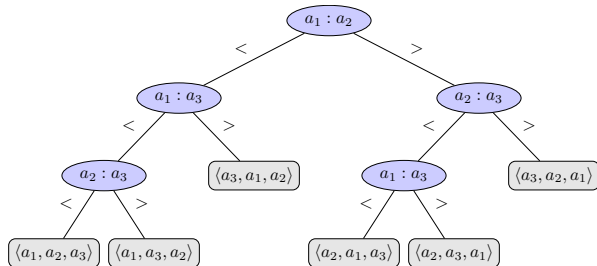
$$\mathsf{LB}(3) =$$

3-element Sorting (Problem 1.1)

(1) Design an algorithm for sorting 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{1}{6}(3 + 3 + 2 + 3 + 3 + 2) = \frac{8}{3}$$
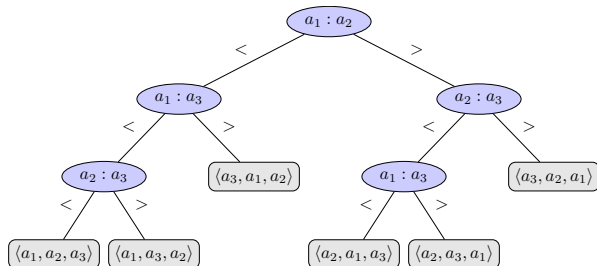
$$\mathsf{LB}(3) = 3$$

3-element Sorting (Problem 1.1)

(1) Design an algorithm for sorting 3 distinct elements.

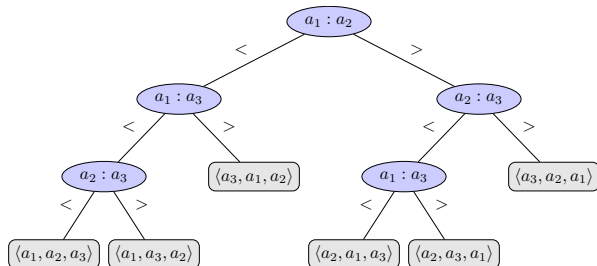(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{1}{6}(3 + 3 + 2 + 3 + 3 + 2) = \frac{8}{3}$$

$$\mathsf{LB}(3) = 3 \qquad (\mathsf{LB}(3) \geq \log 3!)$$

3-element Median Seletion (Problem 1.2)

(1) Design an algorithm for selecting the median of 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

(1) Design an algorithm for selecting the median of 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

3-element Median Seletion (Problem 1.2)
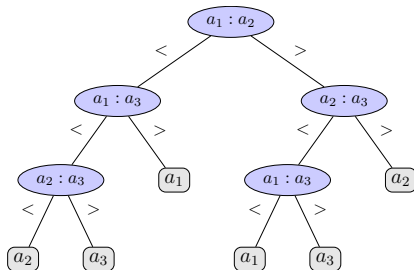
(1) Design an algorithm for selecting the median of 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.

$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{8}{3}$$

3-element Median Seletion (Problem 1.2)

(1) Design an algorithm for selecting the median of 3 distinct elements.

(2) Worst-case and average-case time complexity.
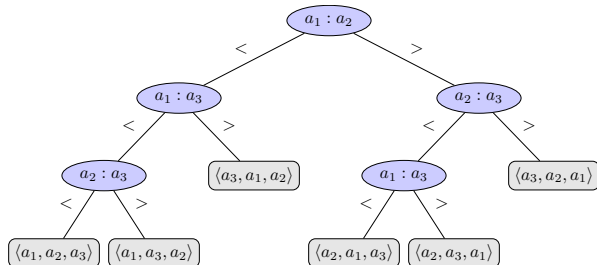
(3) Worst-case lower bound.
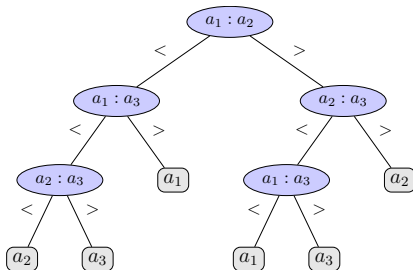
$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{8}{3}$$

$$\mathsf{LB}(3) =$$

3-element Median Seletion (Problem 1.2)

(1) Design an algorithm for selecting the median of 3 distinct elements.

(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.
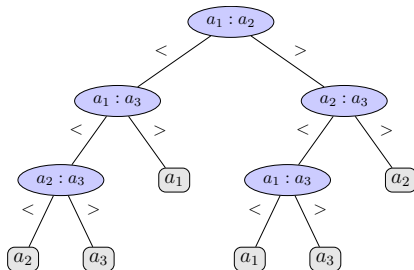
$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{8}{3}$$

$$\mathsf{LB}(3) = 3$$

3-element Median Seletion (Problem 1.2)

(1) Design an algorithm for selecting the median of 3 distinct elements.

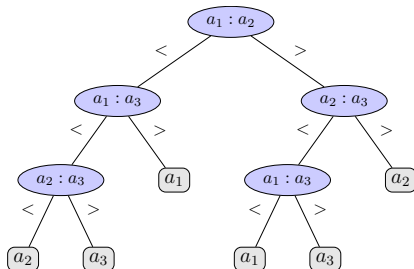(2) Worst-case and average-case time complexity.

(3) Worst-case lower bound.
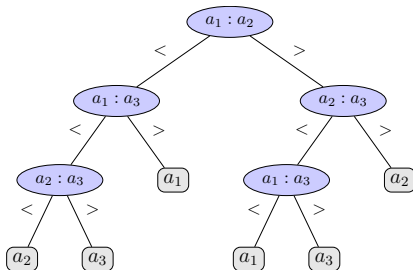
$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{8}{3}$$

$$\mathsf{LB}(3) = 3 \qquad (\mathsf{LB}(3) \geq \frac{3n}{2} - \frac{3}{2})$$

KEEP CALM AND JUST WAIT

$$LB = 2$$

LB = 2

```
1: procedure MEDIAN(a, b, c)
2:     if (a − b)(a − c) < 0 then
3:         return a
4:     if (b − a)(b − c) < 0 then
5:         return b
6:     return c
```

KEEP CALM AND JUST WAIT

1: **procedure** MEDIAN(a, b, c)
2:      **if** $(a - b)(a - c) < 0$ **then**
3:          **return** a
4:      **if** $(b - a)(b - c) < 0$ **then**
5:          **return** b
6:      **return** c

$LB = 2$

Not comparison-based!

$$n = 5$$

Exercise

$$n = 5$$

Reference

"The Art of Computer Programming, Vol 3: Sorting and Searching (Section 5.3.1)" by Donald E. Knuth

$$S(21) = 66$$

Analysis of Bubblesort (Problem 3.2)

(a) Correctness

(b) $W(n)$ & $A(n)$

(c) Improved version $A'(n)$:

$$l \triangleq \max_k \left\{ \text{SWAP}(A[k], A[k+1]) \right\}$$

1: **procedure** BUBBLESORT$(A[1 \cdots n])$
2:      **for** $i \leftarrow n$ **downto** 2 **do**
3:          **for** $j \leftarrow 1$ **to** $i - 1$ **do**
4:              **if** $A[j] > A[j + 1]$ **then**
5:                  $\text{SWAP}(A[j], A[j + 1])$

Analysis of Bubblesort (Problem 3.2)

(a) Correctness

(b) $W(n)$ & $A(n)$

(c) Improved version $A'(n)$:

$$l \triangleq \max_k \left\{ \text{SWAP}(A[k], A[k+1]) \right\}$$

1: **procedure** BUBBLESORT($A[1 \cdots n]$)
2:     **for** $i \leftarrow n$ **downto** 2 **do**
3:         **for** $j \leftarrow 1$ **to** $i - 1$ **do**
4:             **if** $A[j] > A[j+1]$ **then**
5:                 SWAP($A[j], A[j + 1]$)

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

Analysis of Bubblesort (Problem 3.2)

(a) Correctness

(b) $W(n)$ & $A(n)$

(c) Improved version $A'(n)$:

$$l \triangleq \max_k \Big\{ \text{SWAP}(A[k], A[k+1]) \Big\}$$

```
1: procedure BUBBLESORT(A[1···n])
2:     for i ← n downto 2 do
3:         for j ← 1 to i − 1 do
4:             if A[j] > A[j + 1] then
5:                 SWAP(A[j], A[j + 1])
```

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

$$A'(n) =$$

Analysis of Bubblesort (Problem 3.2)

(a) Correctness

(b) $W(n)$ & $A(n)$

(c) Improved version $A'(n)$:

$$l \triangleq \max_k \Big\{ \text{SWAP}(A[k], A[k+1]) \Big\}$$

```
1: procedure BUBBLESORT(A[1 · · · n])
2:      for i ← n downto 2 do
3:          for j ← 1 to i − 1 do
4:              if A[j] > A[j + 1] then
5:                  SWAP(A[j], A[j + 1])
```

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

$$A'(n) = \frac{1}{2}\Big(n^2 - n\ln n - (\gamma + \ln 2 - 1)n\Big) + O(\sqrt{n}) =$$

Analysis of Bubblesort (Problem 3.2)

(a) Correctness

(b) $W(n)$ & $A(n)$

(c) Improved version $A'(n)$:

$$l \triangleq \max_k \Big\{ \text{SWAP}(A[k], A[k+1]) \Big\}$$

```
1: procedure BUBBLESORT(A[1 · · · n])
2:     for i ← n downto 2 do
3:         for j ← 1 to i − 1 do
4:             if A[j] > A[j + 1] then
5:                 SWAP(A[j], A[j + 1])
```

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

$$A'(n) = \frac{1}{2}\Big(n^2 - n \ln n - (\gamma + \ln 2 - 1)n\Big) + O(\sqrt{n}) = \Theta(n^2)$$

Analysis of Bubblesort (Problem 3.2)

(a) Correctness

(b) $W(n)$ & $A(n)$

(c) Improved version $A'(n)$:

$$l \triangleq \max_k \Big\{ \text{Swap}(A[k], A[k+1]) \Big\}$$

1: **procedure** Bubblesort($A[1 \cdots n]$)
2:     **for** $i \leftarrow n$ **downto** 2 **do**
3:         **for** $j \leftarrow 1$ **to** $i - 1$ **do**
4:             **if** $A[j] > A[j + 1]$ **then**
5:                 Swap($A[j], A[j + 1]$)

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

$$A'(n) = \frac{1}{2}\Big(n^2 - n \ln n - (\gamma + \ln 2 - 1)n\Big) + O(\sqrt{n}) = \Theta(n^2)$$

Reference

"The Art of Computer Programming, Vol 3: Sorting and Searching (Section 5.2.2)" by Donald E. Knuth

People who analyze algorithms have double happiness.

— Donald E. Knuth (1995)
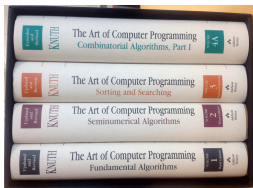
People who analyze algorithms have double happiness.

Then they receive a practical payoff when their theories make it possible to get other jobs done more quickly and more economically.

— Donald E. Knuth (1995)
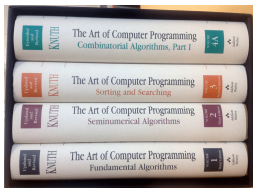
People who analyze algorithms have double happiness.

First of all they experience the sheer beauty of elegant mathematical patterns that surround elegant computational procedures.

Then they receive a practical payoff when their theories make it possible to get other jobs done more quickly and more economically.

— *Donald E. Knuth (1995)*
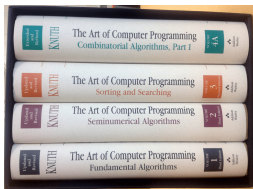
Mathematical Induction

Horner's rule (Problem 1.5)

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

1: **procedure** HORNER($A[0 \ldots n], x$) $\qquad \triangleright A : \{a_0 \ldots a_n\}$
2: $\quad p \leftarrow A[n]$
3: $\quad$ **for** $i \leftarrow n - 1$ **downto** 0 **do**
4: $\quad\quad p \leftarrow px + A[i]$
5: $\quad$ **return** p

Horner's rule (Problem 1.5)

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

1: **procedure** HORNER($A[0 \ldots n], x$) $\triangleright A : \{a_0 \ldots a_n\}$
2:     $p \leftarrow A[n]$

3:     **for** $i \leftarrow n - 1$ **downto** 0 **do**
4:         $p \leftarrow px + A[i]$

5:     **return** p

Loop invariant (after the k-th loop):

Horner's rule (Problem 1.5)

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

1: **procedure** HORNER($A[0 \ldots n], x$) $\qquad \qquad \triangleright A : \{a_0 \ldots a_n\}$
2: $\quad p \leftarrow A[n]$
3: $\quad$ **for** $i \leftarrow n - 1$ **downto** 0 **do**
4: $\qquad p \leftarrow px + A[i]$
5: $\quad$ **return** p

Loop invariant (after the k-th loop):

$$\mathcal{I}: \ p = \sum_{i=n}^{i=n-k} a_i x^{k-(n-i)}$$

$$\mathcal{I} : \ p = \sum_{i=n}^{i=n-k} a_i x^{k-(n-i)}$$

$$\mathcal{I}: \quad p = \sum_{i=n}^{i=n-k} a_i x^{k-(n-i)}$$

When you are in an exam:

20% : Finding $\mathcal{I}$

80% : Proving $\mathcal{I}$ by PMI

Prove by mathematical induction on the number k of loops.

Prove by mathematical induction on the number k of loops.

Base Case: $k = 0$.

Prove by mathematical induction on the number k of loops.

Base Case: $k = 0$.

Inductive Hypothesis: $\mathcal{I}$ is valid after the k-th ($k \geq 0$) loop.

Prove by mathematical induction on the number k of loops.

Base Case: $k = 0$.

Inductive Hypothesis: $\mathcal{I}$ is valid after the k-th ($k \geq 0$) loop.

Inductive Step: $\mathcal{I}$ maintains for the $(k+1)$-th loop:

Base Case: $k = 0$.

Inductive Hypothesis: $\mathcal{I}$ is valid after the k-th ($k \geq 0$) loop.

Inductive Step: $\mathcal{I}$ maintains for the $(k+1)$-th loop:

Prove by mathematical induction on the number k of loops.

Base Case: $k = 0$.

Inductive Hypothesis: $\mathcal{I}$ is valid after the k-th ($k \geq 0$) loop.

Inductive Step: $\mathcal{I}$ maintains for the $(k+1)$-th loop:

$$\left(\sum_{i=n}^{i=n-k} a_i x^{k-(n-i)} \right) \cdot x + A[n-k-1] = \sum_{i=n}^{i=n-(k+1)} a_i x^{(k+1)-(n-i)}$$

Prove by mathematical induction on the number k of loops.

Base Case: $k = 0$.

Inductive Hypothesis: $\mathcal{I}$ is valid after the k-th ($k \geq 0$) loop.

Inductive Step: $\mathcal{I}$ maintains for the $(k+1)$-th loop:

$$\left(\sum_{i=n}^{i=n-k} a_i x^{k-(n-i)} \right) \cdot x + A[n-k-1] = \sum_{i=n}^{i=n-(k+1)} a_i x^{(k+1)-(n-i)}$$

Termination

Prove by mathematical induction on the number k of loops.

Base Case: $k = 0$.

Inductive Hypothesis: $\mathcal{I}$ is valid after the k-th ($k \geq 0$) loop.

Inductive Step: $\mathcal{I}$ maintains for the $(k+1)$-th loop:

$$\left(\sum_{i=n}^{i=n-k} a_i x^{k-(n-i)} \right) \cdot x + A[n-k-1] = \sum_{i=n}^{i=n-(k+1)} a_i x^{(k+1)-(n-i)}$$

Termination

(a) $i \leftarrow n-1$ **downto** 0

(b) $k = n \implies p = \sum_{i=0}^{i=n} a_i x^i$

Integer Multiplication (Problem 1.6)

1: **procedure** INT-MULT(y, z)   ▷ $y, z \geq 0$; $y, z \in \mathbb{Z}$
2:     **if** $z = 0$ **then**
3:         **return** 0
4:     **return** INT-MULT($cy, \lfloor \frac{z}{c} \rfloor$) + $y(z \bmod c)$ ▷ $c \geq 2$

Integer Multiplication (Problem 1.6)

1:	**procedure** INT-MULT(y, z)	$\triangleright\ y, z \geq 0;\ y, z \in \mathbb{Z}$
2:	**if** $z = 0$ **then**	
3:	**return** 0	
4:	**return** INT-MULT$(cy, \lfloor \frac{z}{c} \rfloor) + y(z \bmod c)$	$\triangleright\ c \geq 2$

Prove by mathematical induction on

Integer Multiplication (Problem 1.6)

1:	**procedure** INT-MULT(y, z) $\hspace{2cm} \triangleright\ y, z \geq 0;\ y, z \in \mathbb{Z}$
2:	$\quad$ **if** $z = 0$ **then**
3:	$\quad\quad$ **return** 0
4:	$\quad$ **return** INT-MULT$(cy, \lfloor \frac{z}{c} \rfloor) + y(z \bmod c)$ $\hspace{2cm} \triangleright\ c \geq 2$

Prove by mathematical induction on the non-negative integer z.

Integer Multiplication (Problem 1.6)

1: **procedure** INT-MULT(y, z)	$\triangleright\ y, z \geq 0;\ y, z \in \mathbb{Z}$
2: **if** $z = 0$ **then**	
3: **return** 0	
4: **return** INT-MULT$(cy, \lfloor \frac{z}{c} \rfloor) + y(z \bmod c)$	$\triangleright\ c \geq 2$

Prove by mathematical induction on the non-negative integer z.

$$\text{BC:}\ z = 0: \quad \text{INT-MULT}(y, 0) = 0 = y \cdot 0.$$

1: **procedure** INT-MULT(y, z)	$\triangleright \; y, z \geq 0; \; y, z \in \mathbb{Z}$
2: **if** $z = 0$ **then**	
3: **return** 0	
4: **return** INT-MULT$(cy, \lfloor \frac{z}{c} \rfloor) + y(z \bmod c)$	$\triangleright \; c \geq 2$

Prove by mathematical induction on the non-negative integer z.

$$\text{BC: } z = 0 : \quad \text{INT-MULT}(y, 0) = 0 = y \cdot 0.$$
$$\text{I.H.: } < z \; (z > 0) : \quad \text{INT-MULT}(y, z) = yz.$$

1: **procedure** INT-MULT(y, z)	$\triangleright\ y, z \geq 0;\ y, z \in \mathbb{Z}$
2: **if** $z = 0$ **then**	
3: **return** 0	
4: **return** INT-MULT$(cy, \lfloor \frac{z}{c} \rfloor) + y(z \bmod c)$	$\triangleright\ c \geq 2$

Prove by mathematical induction on the non-negative integer z.

$$\text{BC:}\ z = 0:\quad \text{INT-MULT}(y, 0) = 0 = y \cdot 0.$$
$$\text{I.H.:}\ < z\ (z > 0):\quad \text{INT-MULT}(y, z) = yz.$$
$$\text{I.S.:}\ = z.$$

1: **procedure** INT-MULT(y, z)	$\triangleright\ y, z \geq 0;\ y, z \in \mathbb{Z}$
2: **if** $z = 0$ **then**	
3: **return** 0	
4: **return** INT-MULT$(cy, \lfloor \frac{z}{c} \rfloor) + y(z \bmod c)$	$\triangleright\ c \geq 2$

Prove by mathematical induction on the non-negative integer z.

$$\text{BC: } z = 0: \quad \text{INT-MULT}(y, 0) = 0 = y \cdot 0.$$

$$\text{I.H.: } < z\ (z > 0): \quad \text{INT-MULT}(y, z) = yz.$$

$$\text{I.S.: } = z.$$

Integer Multiplication (Problem 1.6)

1: **procedure** INT-MULT(y, z)	$\triangleright\ y, z \geq 0;\ y, z \in \mathbb{Z}$
2: **if** $z = 0$ **then**	
3: **return** 0	
4: **return** INT-MULT$(cy, \lfloor \frac{z}{c} \rfloor) + y(z \bmod c)$	$\triangleright\ c \geq 2$

Prove by mathematical induction on the non-negative integer z.

$$\text{BC: } z = 0: \quad \text{INT-MULT}(y, 0) = 0 = y \cdot 0.$$
$$\text{I.H.: } < z\ (z > 0): \quad \text{INT-MULT}(y, z) = yz.$$
$$\text{I.S.: } = z.$$

$$\text{INT-MULT}(y, z) = \text{INT-MULT}(cy, \lfloor \frac{z}{c} \rfloor) + y(z \bmod c)$$
$$= cy \cdot \lfloor \frac{z}{c} \rfloor + y(z \bmod c) = yz$$

2-tree; Full Binary Tree (Problem 2.5)
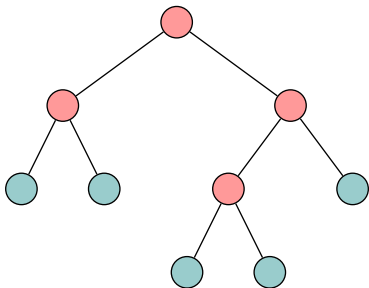
$$n_0 = n_2 + 1$$

Prove by mathematical induction on

2-tree; Full Binary Tree (Problem 2.5)
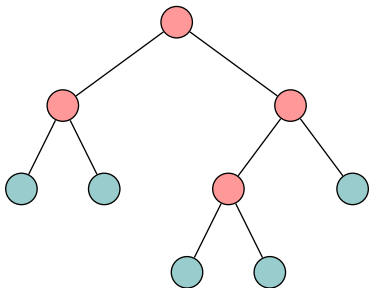
$$n_0 = n_2 + 1$$

Prove by mathematical induction on *the size of binary tree*.
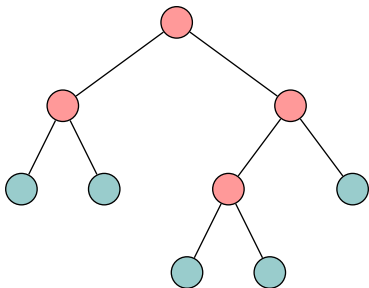
2-tree; Full Binary Tree (Problem 2.5)
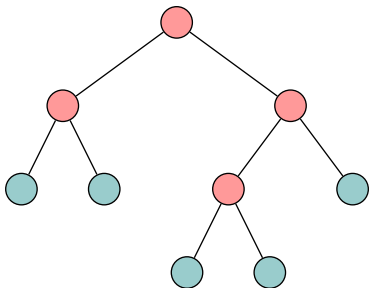
$$n_{0L} = n_{2L} + 1$$

$$n_{0R} = n_{2R} + 1$$

2-tree; Full Binary Tree (Problem 2.5)
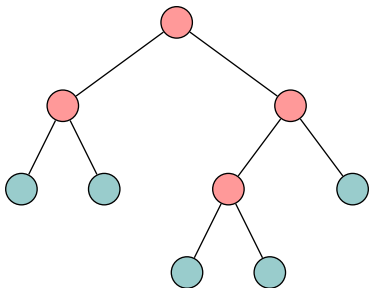
$$n_{0L} = n_{2L} + 1$$

$$n_{0R} = n_{2R} + 1$$

$$n_{0L} + n_{0R} \text{ vs. } n_{2L} + n_{2R} + 1$$

Asymptotics
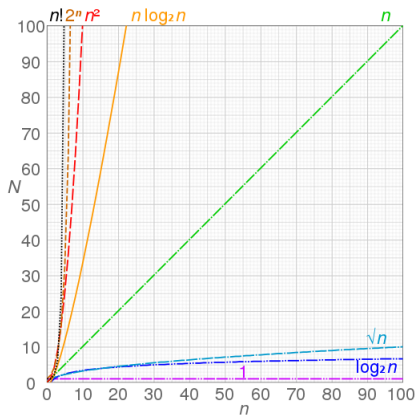
Asymptotics

$$Q : \theta(f) \ ?$$

$$O(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq f(n) \leq cg(n)\}$$

$$\Omega(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq cg(n) \leq f(n)\}$$

$$O(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq f(n) \leq cg(n)\}$$
$$\Omega(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq cg(n) \leq f(n)\}$$

$$\Theta(g(n)) = \{f(n) \mid \exists c_1 > 0, \exists c_2 > 0, \exists n_0, \forall n \geq n_0 :$$
$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

$$O(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq f(n) \leq cg(n)\}$$

$$\Omega(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq cg(n) \leq f(n)\}$$

$$\Theta(g(n)) = \{f(n) \mid \exists c_1 > 0, \exists c_2 > 0, \exists n_0, \forall n \geq n_0 :$$
$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

$$o(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq f(n) < cg(n)\}$$

$$\omega(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq cg(n) < f(n)\}$$

$$O(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq f(n) \leq cg(n)\}$$

$$\Omega(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq cg(n) \leq f(n)\}$$

$$\Theta(g(n)) = \{f(n) \mid \exists c_1 > 0, \exists c_2 > 0, \exists n_0, \forall n \geq n_0 :$$
$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

$$o(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq f(n) < cg(n)\}$$

$$\omega(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0, \forall n \geq n_0 : 0 \leq cg(n) < f(n)\}$$

$$f(n) \sim g(n) \iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = 1$$

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \land f(n) = \Omega(g(n))$$

Asymptotics (Problem $2.6\ (4)$)

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \land f(n) = \Omega(g(n))$$

Asymptotics (Problem $2.6\ (6)$)

$$\Theta(g(n)) \cap o(g(n)) = \emptyset$$

Asymptotics (Problem 2.6 (4))

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$

Asymptotics (Problem 2.6 (6))

$$\Theta(g(n)) \cap o(g(n)) = \emptyset$$

$$Q : f(n) = O(g(n)) \vee g(n) = \Omega(f(n)) \ ?$$

Asymptotics (Problem 2.6 (4))

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \land f(n) = \Omega(g(n))$$

Asymptotics (Problem 2.6 (6))

$$\Theta(g(n)) \cap o(g(n)) = \emptyset$$

$$Q : f(n) = O(g(n)) \lor g(n) = \Omega(f(n)) ?$$

$$f(n) = n, \quad g(n) = n^{1+\sin n}$$

Asymptotics (Problem 2.7 (2))

$$(\log n)^2 \ \text{vs.} \ \sqrt{n}$$

Asymptotics (Problem 2.7 (2))

$$(\log n)^2 \quad \text{vs.} \quad \sqrt{n}$$

$$(\log n)^{c_1} = O(n^{c_2}) \quad c_1, c_2 > 0$$

Asymptotics (Problem 2.10)

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

Asymptotics (Problem 2.10)

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

Stirling Formula (by *James Stirling*):

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Asymptotics (Problem 2.10)

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

Stirling Formula (by *James Stirling*):

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$\log(n!) = \log 1 + \log 2 + \cdots + \log n$$

Asymptotics (Problem 2.10)

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

Stirling Formula (by *James Stirling*):

$$n! \sim \sqrt{2\pi n} \Big(\frac{n}{e}\Big)^n$$

$$\log(n!) = \log 1 + \log 2 + \cdots + \log n$$

$$\log(n!) \leq n \log n$$

Asymptotics (Problem 2.10)

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

Stirling Formula (by *James Stirling*):

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$\log(n!) = \log 1 + \log 2 + \cdots + \log n$$

$$\log(n!) \leq n \log n \qquad \log(n!) \geq \frac{n}{2} \log \frac{n}{2}$$

Summary (Problem 2.20)

1: **procedure** CONUNDRUM(n)
2:     $r \leftarrow 0$

3:     **for** $i \leftarrow 1$ **to** n **do**
4:         **for** $j \leftarrow i + 1$ **to** n **do**
5:             **for** $k \leftarrow i + j - 1$ **to** n **do**
6:                 $r \leftarrow r + 1$

7:     **return** r

1: **procedure** Conundrum(n)
2: $\quad r \leftarrow 0$
3: $\quad$ **for** $i \leftarrow 1$ **to** n **do**
4: $\quad\quad$ **for** $j \leftarrow i + 1$ **to** n **do**
5: $\quad\quad\quad$ **for** $k \leftarrow i + j - 1$ **to** n **do**
6: $\quad\quad\quad\quad r \leftarrow r + 1$
7: $\quad$ **return** r

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=i+j-1}^{n} 1 = \cdots$$

Reference:

"Big Omicron and Big Omega and Big Theta" by Donald E. Knuth, 1976.

Recurrences
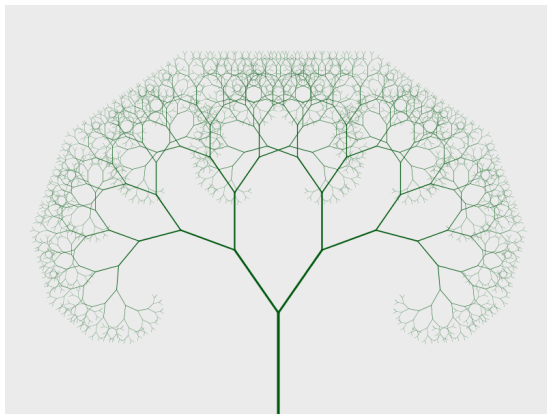
$$T(n) = aT(n/b) + f(n) \quad (a > 0, b > 1)$$

$$T(n) = aT(n/b) + f(n) \quad (a > 0, b > 1)$$

Assume that $T(n)$ is constant for sufficiently small n.

$$T(n) = aT(n/b) + f(n) \quad (a > 0, b > 1)$$

Assume that $T(n)$ is constant for sufficiently small n.

$$\left.\begin{array}{r} f(n) \\ af(\dfrac{n}{b}) \\ a^2 f(\dfrac{n}{b^2}) \\ \vdots \\ a^{\log_b n} f(c) = \Theta(n^{\log_b a}) \end{array}\right\}$$

$$T(n) = aT(n/b) + f(n) \quad (a > 0, b > 1)$$

Assume that $T(n)$ is constant for sufficiently small n.

$$
\left.
\begin{array}{c}
f(n) \\
af(\dfrac{n}{b}) \\
a^2 f(\dfrac{n}{b^2}) \\
\vdots \\
a^{\log_b n} f(c) = \Theta(n^{\log_b a})
\end{array}
\right\}
\sum \;\; f(n) \stackrel{\text{vs.}}{=} n^E
\left\{
\begin{array}{cl}
n^{\log_b a} & f(n) = O(n^{E-\epsilon}) \\
n^{\log_b a} \log n & f(n) = \Theta(n^E) \\
f(n) & f(n) = \Omega(n^{E+\epsilon})
\end{array}
\right.
$$

Solving Recurrences
(Problem 2.15)

(1) $\Theta(n^{\log_3 2})$

(2) $\Theta(\log^2 n)$

(3) $\Theta(n)$

(4) $\Theta(n \log n)$

(5) $\Theta(n \log^2 n)$

(6) $\Theta(n^2)$

(7) $\Theta(n^{\frac{3}{2}} \log n)$

(8) $\Theta(n)$

(9) $\Theta(n^{c+1})$

(10) $\Theta(c^{n+1})$

(11) $\cdots$

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

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

Solving Recurrences
(Problem 2.15)

(1) $\Theta(n^{\log_3 2})$

(2) $\Theta(\log^2 n)$

(3) $\Theta(n)$

(4) $\Theta(n \log n)$

(5) $\Theta(n \log^2 n)$

(6) $\Theta(n^2)$

(7) $\Theta(n^{\frac{3}{2}} \log n)$

(8) $\Theta(n)$

(9) $\Theta(n^{c+1})$

(10) $\Theta(c^{n+1})$

(11) $\cdots$

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

Reference:

$$f(n) = \Theta(n^{\log_b a} \log^k n) \implies \Theta(n^{\log_b a} \log^{k+1} n)$$

Solving Recurrences
(Problem 2.15)

(1) $\Theta(n^{\log_3 2})$

(2) $\Theta(\log^2 n)$

(3) $\Theta(n)$

(4) $\Theta(n \log n)$

(5) $\Theta(n \log^2 n)$

(6) $\Theta(n^2)$

(7) $\Theta(n^{\frac{3}{2}} \log n)$

(8) $\Theta(n)$

(9) $\Theta(n^{c+1})$

(10) $\Theta(c^{n+1})$

(11) $\cdots$

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

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

Reference:

$$f(n) = \Theta(n^{\log_b a} \log^k n) \implies \Theta(n^{\log_b a} \log^{k+1} n)$$

Gaps in Master Theorem (Problem 2.18)

Solving Recurrences (Problem 2.15)

(1) $\Theta(n^{\log_3 2})$

(2) $\Theta(\log^2 n)$

(3) $\Theta(n)$

(4) $\Theta(n \log n)$

(5) $\Theta(n \log^2 n)$

(6) $\Theta(n^2)$

(7) $\Theta(n^{\frac{3}{2}} \log n)$

(8) $\Theta(n)$

(9) $\Theta(n^{c+1})$

(10) $\Theta(c^{n+1})$

(11) $\cdots$

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

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

Reference:

$$f(n) = \Theta(n^{\log_b a} \log^k n) \implies \Theta(n^{\log_b a} \log^{k+1} n)$$

Gaps in Master Theorem (Problem 2.18)

$$T(n) = 2T(n/2) + \frac{n}{\log n}$$

Solving Recurrences
(Problem 2.15)

(1) $\Theta(n^{\log_3 2})$

(2) $\Theta(\log^2 n)$

(3) $\Theta(n)$

(4) $\Theta(n \log n)$

(5) $\Theta(n \log^2 n)$

(6) $\Theta(n^2)$

(7) $\Theta(n^{\frac{3}{2}} \log n)$

(8) $\Theta(n)$

(9) $\Theta(n^{c+1})$

(10) $\Theta(c^{n+1})$

(11) $\cdots$

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

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

Reference:

$$f(n) = \Theta(n^{\log_b a} \log^k n) \implies \Theta(n^{\log_b a} \log^{k+1} n)$$

Gaps in Master Theorem (Problem 2.18)

$$T(n) = 2T(n/2) + \frac{n}{\log n} = \Theta(n \log \log n)$$

Solving Recurrences
(Problem 2.15)

(1) $\Theta(n^{\log_3 2})$

(2) $\Theta(\log^2 n)$

(3) $\Theta(n)$

(4) $\Theta(n \log n)$

(5) $\Theta(n \log^2 n)$

(6) $\Theta(n^2)$

(7) $\Theta(n^{\frac{3}{2}} \log n)$

(8) $\Theta(n)$

(9) $\Theta(n^{c+1})$

(10) $\Theta(c^{n+1})$

(11) $\cdots$

$$T(n) = T(n-1) + n^c \quad c \geq 1$$

$$T(n) = T(n-1) + c^n \quad c > 1$$
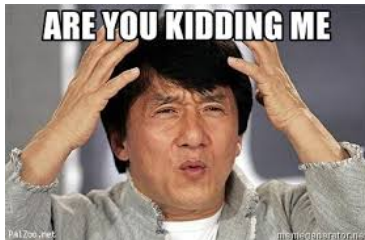
Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8)$$

Solving Recurrences (Problem $2.15\ (11)$)

$$T(n) = T(n/2) + T(n/4) + T(n/8)$$

$$T(n) = T(n/2) + T(n/4) + T(n/8)$$

$$\boxed{T(n) = \Theta(n^{0.879146})}$$

Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8)$$

$$\boxed{T(n) = \Theta(n^{0.879146})}$$

$$\boxed{T(n) = \Theta(n^{\alpha})}$$

Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8)$$

$$\boxed{T(n) = \Theta(n^{0.879146})}$$

$$\boxed{T(n) = \Theta(n^{\alpha})}$$

$$2^{-\alpha} + 4^{-\alpha} + 8^{-\alpha} = 1$$

Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8)$$

$$\boxed{T(n) = \Theta(n^{0.879146})}$$

$$\boxed{T(n) = \Theta(n^{\alpha})}$$

$$2^{-\alpha} + 4^{-\alpha} + 8^{-\alpha} = 1$$

```
Solve[2^{-x} + 4^{-x} + 8^{-x} == 1, x] // N
```

Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n$$

Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n$$

By recursion-tree.

Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n$$

By recursion-tree.

$$\boxed{T(n) = \Theta(n)}$$

Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n$$

By recursion-tree.

$$\boxed{T(n) = \Theta(n)}$$

Exercise: Prove it by mathematical induction.

Solving Recurrences (Problem 2.15 (11))

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n$$

By recursion-tree.

$$\boxed{T(n) = \Theta(n)}$$

Exercise: Prove it by mathematical induction.

Reference:

"On the Solution of Linear Recurrence Equations" by Akra & Bazzi, 1996.

$$T(n) = \sum_{i=1}^{k} a_i T(n/b_i) + f(n)$$

Solving Recurrences (Problem 2.17)

$$\mathsf{T}(n) = \sqrt{n}\ \mathsf{T}(\sqrt{n}) + n$$
$$= n^{\frac{1}{2}}\ \mathsf{T}\left(n^{\frac{1}{2}}\right) + n$$
$$= n^{\frac{1}{2}}\left(n^{\frac{1}{2^2}}\ \mathsf{T}\left(n^{\frac{1}{2^2}}\right) + n^{\frac{1}{2}}\right) + n$$
$$= n^{\frac{1}{2} + \frac{1}{2^2}}\ \mathsf{T}\left(n^{\frac{1}{2^2}}\right) + 2n$$
$$= n^{\frac{1}{2} + \frac{1}{2^2}}\left(n^{\frac{1}{2^3}}\ \mathsf{T}\left(n^{\frac{1}{2^3}}\right) + n^{\frac{1}{2^2}}\right) + 2n$$
$$= n^{\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3}}\ \mathsf{T}\left(n^{\frac{1}{2^3}}\right) + 3n$$
$$= \cdots$$
$$= n^{\sum_{i=1}^{k} \frac{1}{2^i}}\ \mathsf{T}\left(n^{\frac{1}{2^k}}\right) + kn$$

$$n^{\frac{1}{2^k}} = 2$$

$$n^{\frac{1}{2^k}} = 2 \implies k = \log \log n$$

$$n^{\frac{1}{2^k}} = 2 \implies k = \log \log n$$

$$\mathsf{T}(n) = n^{\sum_{i=1}^{k} \frac{1}{2^i}} \, \mathsf{T}\left(n^{\frac{1}{2^k}}\right) + kn$$

$$= n^{\sum_{i=1}^{\log \log n} \frac{1}{2^i}} \, \mathsf{T}(2) + n \log \log n$$

$$n^{\frac{1}{2^k}} = 2 \implies k = \log \log n$$

$$\mathsf{T}(n) = n^{\sum_{i=1}^{k} \frac{1}{2^i}} \; \mathsf{T}\left(n^{\frac{1}{2^k}}\right) + kn$$

$$= n^{\sum_{i=1}^{\log \log n} \frac{1}{2^i}} \; \mathsf{T}(2) + n \log \log n$$

$$\sum_{i=1}^{\log_2 \log_2(n)} \frac{1}{2^i} < 1 \implies T(n) = \Theta(n \log \log n)$$

$$n^{\frac{1}{2^k}} = 2 \implies k = \log \log n$$

$$\mathsf{T}(n) = n^{\sum_{i=1}^{k} \frac{1}{2^i}} \, \mathsf{T}\left(n^{\frac{1}{2^k}}\right) + kn$$

$$= n^{\sum_{i=1}^{\log \log n} \frac{1}{2^i}} \, \mathsf{T}(2) + n \log \log n$$

$$\sum_{i=1}^{\log_2 \log_2(n)} \frac{1}{2^i} < 1 \implies T(n) = \Theta(n \log \log n)$$

Exercise: Prove it by mathematical induction.

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

$$\boxed{T(n) = \sqrt{n}\,T(\sqrt{n}) + n}$$

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

$$\boxed{T(n) = \sqrt{n}\,T(\sqrt{n}) + n}$$

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

$$n \leftrightarrow 2^m$$

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

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

$$n \leftrightarrow 2^m$$

$$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2})}{2^{m/2}} + 1$$

$$\boxed{T(n) = \sqrt{n}\,T(\sqrt{n}) + n}$$

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

$$n \leftrightarrow 2^m$$

$$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2})}{2^{m/2}} + 1$$

$$S(m) \leftrightarrow \frac{T(2^m)}{2^m}$$

$$\boxed{T(n) = \sqrt{n}\,T(\sqrt{n}) + n}$$

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

$$n \leftrightarrow 2^m$$

$$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2})}{2^{m/2}} + 1$$

$$S(m) \leftrightarrow \frac{T(2^m)}{2^m}$$

$$S(m) = S(m/2) + 1 =$$

$$\boxed{T(n) = \sqrt{n}\,T(\sqrt{n}) + n}$$

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

$$n \leftrightarrow 2^m$$

$$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2})}{2^{m/2}} + 1$$

$$S(m) \leftrightarrow \frac{T(2^m)}{2^m}$$

$$S(m) = S(m/2) + 1 = \Theta(\log m)$$

$$\boxed{T(n) = \sqrt{n}\,T(\sqrt{n}) + n}$$

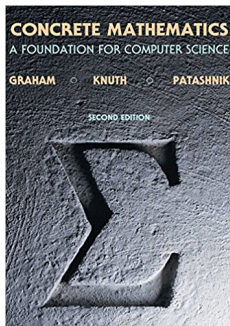$$\frac{T(n)}{n} = \frac{T(\sqrt{n})}{\sqrt{n}} + 1$$

$$n \leftrightarrow 2^m$$

$$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2})}{2^{m/2}} + 1$$

$$S(m) \leftrightarrow \frac{T(2^m)}{2^m}$$

$$S(m) = S(m/2) + 1 = \Theta(\log m)$$

$$\boxed{T(n) = n \log \log n}$$

Maximal Sum Subarray (Problem 3.7)

- array $A[1 \cdots n], a_i >=< 0$
- to find (the sum of) an MS in A

$$A[-2, 1, -3, \boxed{4, -1, 2, 1}, -5, 4]$$

Maximal Sum Subarray (Problem 3.7)

- array $A[1 \cdots n], a_i >=< 0$
- to find (the sum of) an MS in A

$$A[-2, 1, -3, \boxed{4, -1, 2, 1}, -5, 4]$$

Trial and error.

- try subproblem MSS$[i]$: the sum of the MS (MS[i]) in $A[1 \cdots i]$
- goal: mss $=$ MSS$[n]$

Maximal Sum Subarray (Problem 3.7)

- array $A[1 \cdots n], a_i \geq = < 0$
- to find (the sum of) an MS in A

$$A[-2, 1, -3, \boxed{4, -1, 2, 1}, -5, 4]$$

Trial and error.

- try subproblem $MSS[i]$: the sum of the MS (MS[i]) in $A[1 \cdots i]$
- goal: $mss = MSS[n]$
- question: Is $a_i \in MS[i]$?
- recurrence:
$$MSS[i] = \max\{MSS[i-1], ???\}$$

Solution.

- subproblem MSS[i]: the sum of the MS *ending with* a_i or 0
- goal:

$$\mathsf{mss} = \max_{1 \leq i \leq n} \mathsf{MSS}[i]$$

Solution.

- ▶ subproblem MSS[i]: the sum of the MS *ending with* a_i or 0
- ▶ goal:

$$\mathsf{mss} = \max_{1 \le i \le n} \mathsf{MSS}[i]$$

- ▶ question: where does the MS[i] start?
- ▶ recurrence:

$$\mathsf{MSS}[i] = \max\left\{\mathsf{MSS}[i-1] + a_i, 0\right\} \quad \text{(prove it!)}$$

Solution.

- subproblem MSS[i]: the sum of the MS *ending with* a_i or 0
- goal:

$$\mathsf{mss} = \max_{1 \leq i \leq n} \mathsf{MSS}[i]$$

- question: where does the MS[i] start?
- recurrence:

$$\mathsf{MSS}[i] = \max \left\{ \mathsf{MSS}[i-1] + a_i, 0 \right\} \quad \text{(prove it!)}$$

- initialization: MSS[0] = 0

1: **procedure** $\mathrm{MSS}(A[1 \cdots n])$
2:     $\mathsf{MSS}[0] \leftarrow 0$

3:     **for** $i \leftarrow 1$ **to** n **do**
4:         $\mathsf{MSS}[i] \leftarrow \max \{\mathsf{MSS}[i-1] + A[i], 0\}$

5:     **return** $\max\limits_{1 \leq i \leq n} \mathsf{MSS}[i]$

```
1: procedure MSS(A[1 ··· n])
2:     MSS[0] ← 0

3:     for i ← 1 to n do
4:         MSS[i] ← max {MSS[i − 1] + A[i], 0}

5:     return max MSS[i]
             1≤i≤n
```

```
1: procedure MSS(A[1 ··· n])
2:     mss ← 0
3:     MSS ← 0

4:     for i ← 1 to n do
5:         MSS ← max {MSS + A[i], 0}
6:         mss ← max {mss, MSS}

7:     return mss
```

Office 302

Mailbox: H016

hfwei@nju.edu.cn