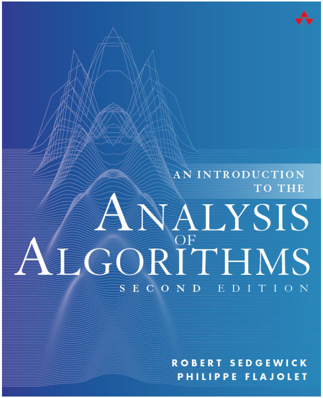
# Asymptotics, Recurrences, and Divide and Conquer

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 $n$

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

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

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

Problem $P$ Algorithm $A$

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

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

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

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

## Average-case Time Complexity (Problem 1.8)

$$\text{Input} : r \in [1, n], \;\; r \in \mathbb{Z}^+$$

$$P\{r = i\} = \begin{cases} \frac{1}{n}, & 1 \le i \le \frac{n}{4} \\ \frac{2}{n}, & \frac{n}{4} < i \le \frac{n}{2} \\ \frac{1}{2n}, & \frac{n}{2} < i \le n \end{cases}$$

## Average-case Time Complexity (Problem 1.8)

$$\text{Input}: r \in [1, n], \quad r \in \mathbb{Z}^+$$

$$P\{r = i\} = \begin{cases} \frac{1}{n}, & 1 \le i \le \frac{n}{4} \\ \frac{2}{n}, & \frac{n}{4} < i \le \frac{n}{2} \\ \frac{1}{2n}, & \frac{n}{2} < i \le n \end{cases}$$

$$\begin{aligned} A &= \sum_{X \in \mathcal{X}} T(X) \cdot P(X) \\ &= T(1)P(1) + T(2)P(2) + \cdots + T(n)P(n) \\ &= \frac{n}{4} \times 10 \times \frac{1}{n} + \frac{n}{4} \times 20 \times \frac{2}{n} + \frac{n}{4} \times 30 \times \frac{1}{2n} + \frac{n}{4} \times n \times \frac{1}{2n} \\ &= \frac{1}{8}n + \frac{65}{4} \end{aligned}$$

# Average-case Analysis of Quicksort

$$A(n) = n - 1 + \frac{1}{n} \sum_{i=0}^{i=n-1} (A(i) + A(n-i-1))$$

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

# Average-case Analysis of Quicksort

$$A(n) = n - 1 + \frac{1}{n} \sum_{i=0}^{i=n-1} \left(A(i) + A(n-i-1)\right)$$

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

$$\begin{aligned} A(n) &= \mathbb{E}[T(X)] \\ &= \mathbb{E}[\mathbb{E}[T(X)|I]] \\ &= \sum_{i=0}^{i=n-1} P(I = i)\, \mathbb{E}[T(X) \mid I = i] \\ &= \sum_{i=0}^{i=n-1} \frac{1}{n}[n - 1 + A(i) + A(n-i-1)] \end{aligned}$$

## $3$-element Sorting (Problem $1.1$)

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

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

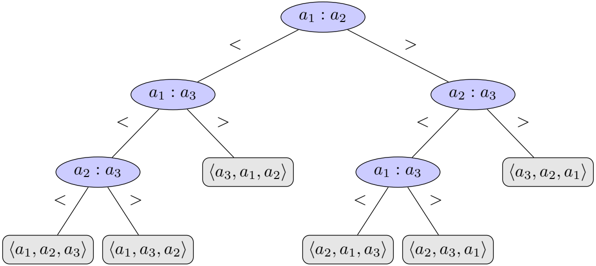
(3) Worst-case lower bound.

## 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.

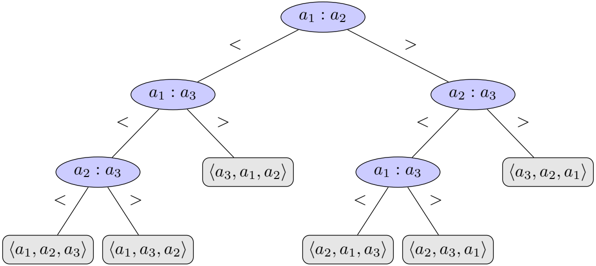


## 3-element Sorting (Problem 1.1)

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

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

(3) Worst-case lower bound.



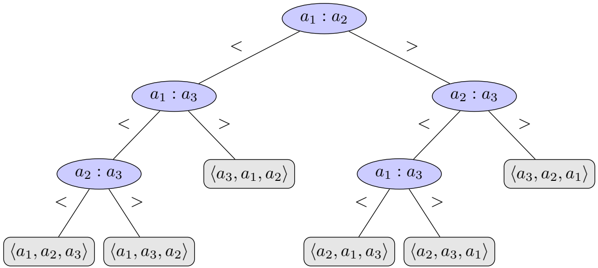

$$W(3) =$$

## 3-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.



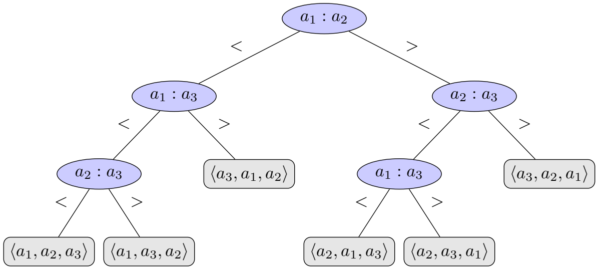

$$W(3) = 3$$

## 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.



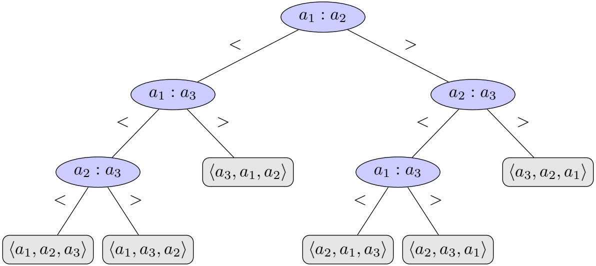

$W(3) = 3 \qquad B(3) =$

## 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.



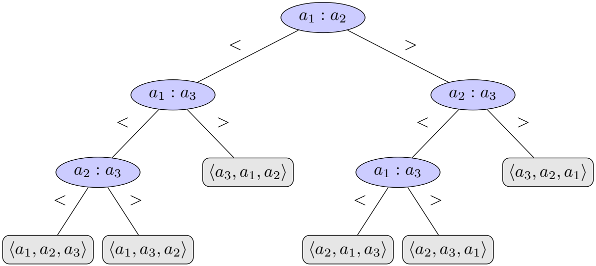

$$W(3) = 3 \qquad B(3) = 2$$

## 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.



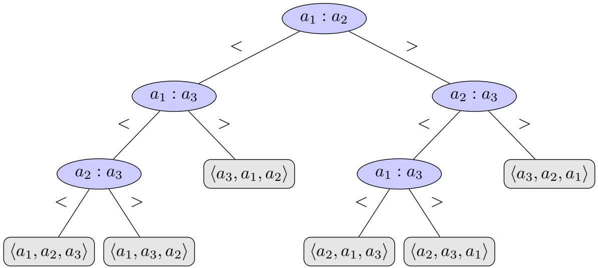

$W(3) = 3 \qquad B(3) = 2 \qquad A(3) =$

## 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.



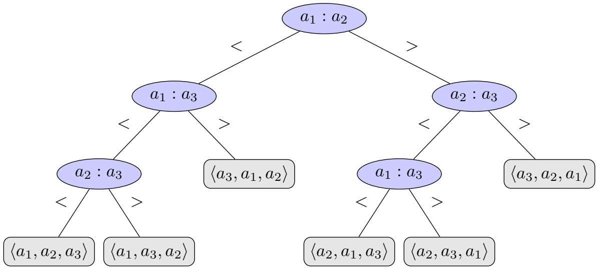

$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{1}{6}(3+3+2+3+3+2) = \frac{8}{3}$$

## 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.




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

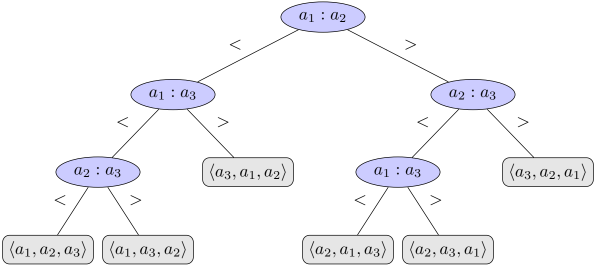
$$\mathsf{LB}(3) =$$

## 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.




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

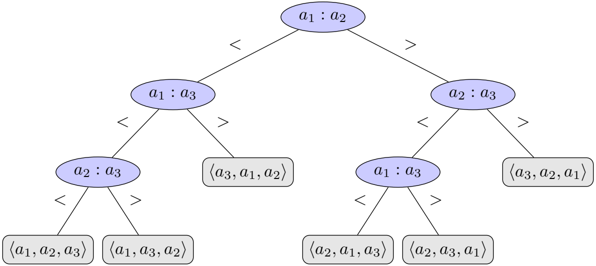
$$\mathsf{LB}(3) = 3$$

## 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.

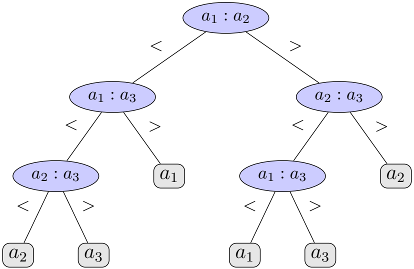
(3) Worst-case lower bound.

## $3$-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.

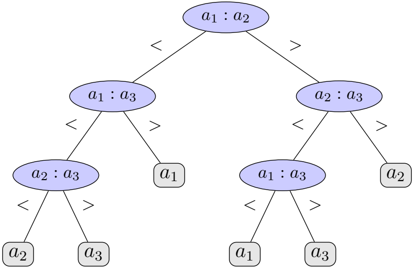


## 3-element Median Seletion (Problem 1.2)

(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.



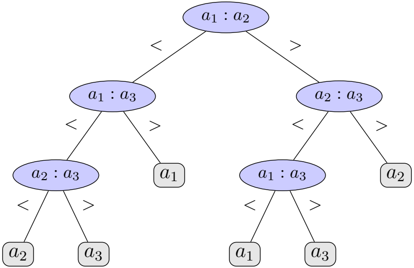

$$W(3) = 3 \qquad B(3) = 2 \qquad A(3) = \frac{8}{3}$$

## 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.




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

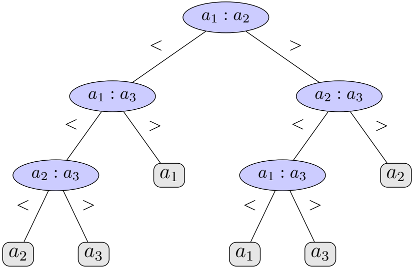
$$\text{LB}(3) =$$

## 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.




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

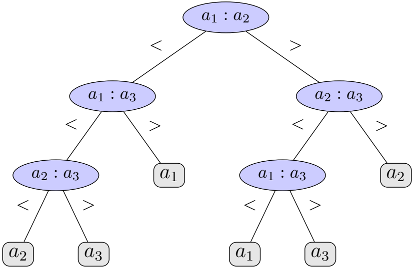
$$\text{LB}(3) = 3$$

## 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.




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

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


$\text{LB} = 2$


$\text{LB} = 2$

```
procedure MEDIAN(a, b, c)
    if (a − b)(a − c) < 0 then
        return a
    if (b − a)(b − c) < 0 then
        return b
    return c
```


**procedure** $\textsc{Median}(a, b, c)$
    **if** $(a-b)(a-c) < 0$ **then**
        **return** $a$
    **if** $(b-a)(b-c) < 0$ **then**
        **return** $b$
    **return** $c$

$\text{LB} = 2$

Not comparison-based!

Exercise

$$n = 5$$

Exercise

$$n = 5$$

Reference

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

$$S(21) = 66$$

# Mathematical Induction

## Horner's rule (Problem 1.5)

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

```
procedure Horner(A[0...n], x)                ▷ A : {a_0 ... a_n}
    p ← A[n]
    for i ← n − 1 downto 0 do
        p ← px + A[i]
    return p
```

## Horner's rule (Problem 1.5)

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

```
procedure HORNER(A[0...n], x)                ▷ A : {a_0 ... a_n}
    p ← A[n]
    for i ← n − 1 downto 0 do
        p ← px + A[i]
    return p
```

Loop invariant (after the $k$-th loop):

## Horner's rule (Problem 1.5)

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

---

**procedure** $\textsc{Horner}(A[0 \ldots n], x)$ $\triangleright\ A : \{a_0 \ldots a_n\}$

    $p \leftarrow A[n]$

    **for** $i \leftarrow n-1$ **downto** $0$ **do**

        $p \leftarrow px + A[i]$

    **return** $p$

---

Loop invariant (after the $k$-th loop):

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

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

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


When you are in an exam:

- $20\%$ : Finding $\mathcal{I}$
- $80\%$ : Proving $\mathcal{I}$ by PMI

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

Proof.

Prove by mathematical induction on non-negative integer $k$, the number of loops.

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

Proof.

Prove by mathematical induction on non-negative integer $k$, the number of loops.

Basis:

$$k = 0 : p = a_n = \mathcal{I}_0$$

Inductive Hypothesis:

Inductive Step:

□

## Integer Multiplication (Problem 1.6)

```
procedure INT-MULT(y, z)
    if z = 0 then
        return 0
    return INT-MULT(cy, ⌊z/c⌋) + y(z mod c)
```

$$\textbf{procedure } \textsc{Int-Mult}(y, z)$$
$$\quad \textbf{if } z = 0 \textbf{ then}$$
$$\qquad \textbf{return } 0$$
$$\quad \textbf{return } \textsc{Int-Mult}(cy, \lfloor \tfrac{z}{c} \rfloor) + y(z \mod c)$$

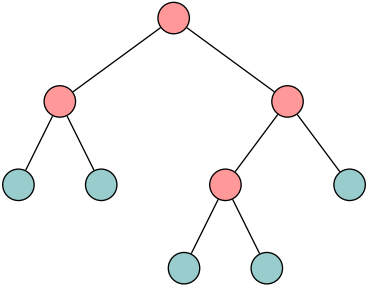
### Integer Multiplication (Problem $1.6$)

```
procedure INT-MULT(y, z)
    if z = 0 then
        return 0
    return INT-MULT(cy, ⌊z/c⌋) + y(z mod c)
```

Proof.

Prove by mathematical induction on non-negative integer $z$. $\square$

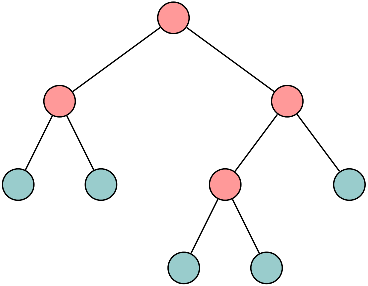
## 2-tree; full binary tree (Problem 2.5)



$$n_0 = n_2 + 1$$

Proof.

2-tree; full binary tree (Problem $2.5$)



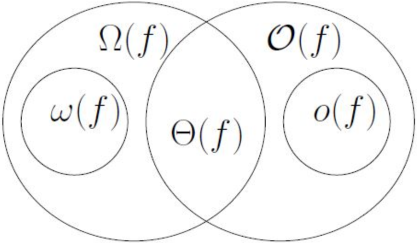
$$n_0 = n_2 + 1$$

Proof.

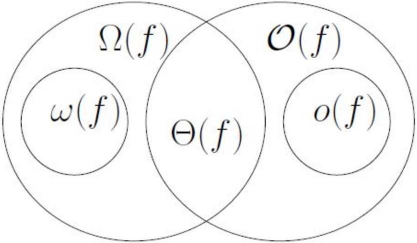
Prove by mathematical induction on *the structure of binary tree.*

$\square$

## Asymptotics



# Asymptotics




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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 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 (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))

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

### Asymptotics (Problem 2.6 (6))

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

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

## 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 $2.6$ $(4)$)

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

## Asymptotics (Problem $2.6$ $(6)$)

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

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

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

## Reference:

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

Asymptotics (Problem 2.7 (2))

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

## Asymptotics (Problem 2.7 (2))

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

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

## 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*):

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

## Asymptotics (Problem 2.10)

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

## Stirling Formula (by *James Stirling*):

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

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

Asymptotics (Problem 2.10)

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

Stirling Formula (by *James Stirling*):

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

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

$$\log(n!) \le n \log n$$

## Asymptotics (Problem $2.10$)

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

## Stirling Formula (by *James Stirling*):

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

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

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

## Summation (Problem 2.20)

```
procedure CONUNDRUM(n)
    r ← 0
    for i ← 1 to n do
        for j ← i + 1 to n do
            for k ← i + j − 1 to n do
                r ← r + 1
    return r
```

## Summation (Problem 2.20)

```
procedure CONUNDRUM(n)
    r ← 0
    for i ← 1 to n do
        for j ← i + 1 to n do
            for k ← i + j − 1 to n do
                r ← r + 1
    return r
```

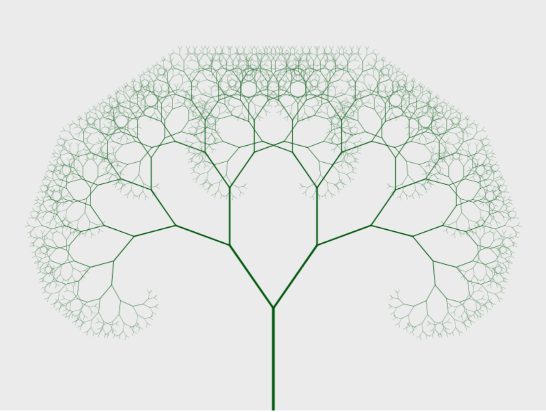
$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=i+j-1}^{n} 1 =$$

## Summation (Problem 2.20)

```
procedure CONUNDRUM(n)
    r ← 0
    for i ← 1 to n do
        for j ← i + 1 to n do
            for k ← i + j − 1 to n do
                r ← r + 1
    return r
```

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=i+j-1}^{n} 1 = \frac{n^2 - n}{2} = \Theta(n^2)$$

# 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 $n$.

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

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

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

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

## 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:

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

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

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

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

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

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

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

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

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

By recursion-tree.

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

## 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)

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

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

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

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

$$\begin{aligned}\mathsf{T}(n) &= n^{\sum_{i=1}^{k} \frac{1}{2^i}}\ \mathsf{T}\left(n^{\frac{1}{2^k}}\right) + kn \\ &= n^{\sum_{i=1}^{\log\log n} \frac{1}{2^i}}\ \mathsf{T}(2) + n\log\log n\end{aligned}$$

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

$$\begin{aligned} \mathsf{T}(n) &= n^{\sum_{i=1}^{k} \frac{1}{2^i}} \ \mathsf{T}\left(n^{\frac{1}{2^k}}\right) + kn \\ &= n^{\sum_{i=1}^{\log\log n} \frac{1}{2^i}} \ \mathsf{T}(2) + n \log\log n \end{aligned}$$

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

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

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

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

Exercise: Prove it by mathematical induction.

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

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

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

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

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

$$n \leftrightarrow 2^m$$

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

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

$$n \leftrightarrow 2^m$$

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

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

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

$$n \leftrightarrow 2^m$$

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

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

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

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

$$n \leftrightarrow 2^m$$

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

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

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

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

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

$$n \leftrightarrow 2^m$$

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

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

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

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

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

$$n \leftrightarrow 2^m$$

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

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

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

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