

Program Design and Algorithms

Lecture 2: Asymptotic Notations and Recurrences



Yongxin Tong (童咏昕)

School of CSE, Beihang University

yxtong@buaa.edu.cn

Outline

- Asymptotic Notations (渐近记号)
 - Big-Oh
 - Big-Omega
 - Big-Theta
 - Algorithm Design and Algorithm Turing
- Solving Recurrences
 - Recursion-tree Method (递归树法)
 - Substitution Method (代入法/替代法)
 - Master Method and Master Theorem (主方法)

Outline

- Asymptotic Notations (渐近记号)
 - Big-Oh
 - Big-Omega
 - Big-Theta
 - Algorithm Design and Algorithm Turing
- Solving Recurrences
 - Recursion-tree Method (递归树法)
 - Substitution Method (代入法/替代法)
 - Master Method and Master Theorem (主方法)

Big-Oh

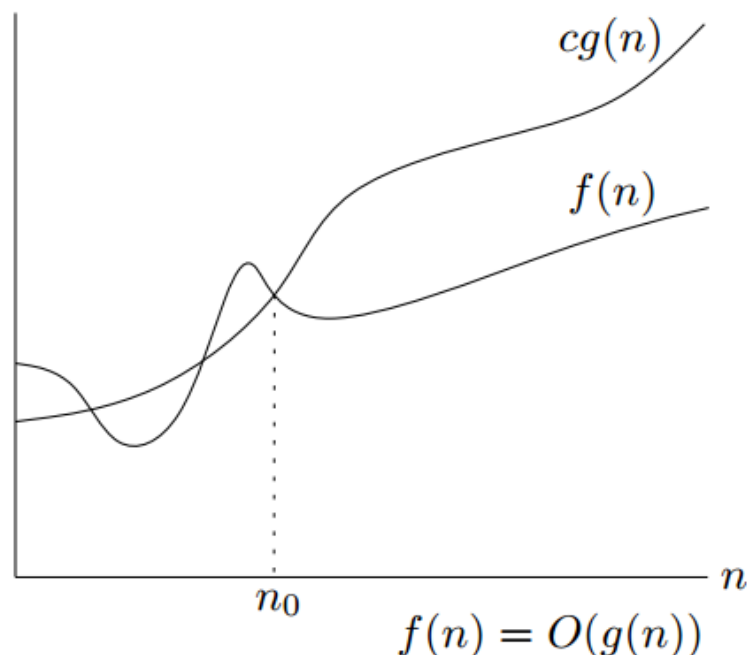
Asymptotic upper bound

Definition (big-Oh)

$f(n) = O(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

When estimating the growth rate of $T(n)$ using big-Oh:

- ignore the low order terms
- ignore the constant coefficient of the most significant term



Big-Oh: Example

Definition (big-Oh)

$f(n) = O(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

Example

Let $T(n) = 3n^2 + 4n + 5$. Prove that $T(n) = O(n^2)$.

Big-Oh: Example

Definition (big-Oh)

$f(n) = O(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

Example

Let $T(n) = 3n^2 + 4n + 5$. Prove that $T(n) = O(n^2)$.

Proof.

$$\begin{aligned} T(n) &= 3n^2 + 4n + 5 \\ &\leq 3n^2 + 4n^2 + 5n^2 \\ &= 12n^2. \end{aligned}$$

Thus, $T(n) \leq 12n^2$ for all $n \geq 1$. Setting $n_0 = 1$ and $c = 12$ in the definition, we have that $T(n) = O(n^2)$. \square

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} = O(1)$
- $\sum_{i=1}^n i^2$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} = O(1)$
- $\sum_{i=1}^n i^2 \leq n \cdot n^2 =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} = O(1)$
- $\sum_{i=1}^n i^2 \leq n \cdot n^2 = O(n^3)$
- $\sum_{i=1}^n i$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} = O(1)$
- $\sum_{i=1}^n i^2 \leq n \cdot n^2 = O(n^3)$
- $\sum_{i=1}^n i \leq n \cdot n =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} = O(1)$
- $\sum_{i=1}^n i^2 \leq n \cdot n^2 = O(n^3)$
- $\sum_{i=1}^n i \leq n \cdot n = O(n^2)$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} = O(1)$
- $\sum_{i=1}^n i^2 \leq n \cdot n^2 = O(n^3)$
- $\sum_{i=1}^n i \leq n \cdot n = O(n^2)$
- $\log(n!) =$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} = O(1)$
- $\sum_{i=1}^n i^2 \leq n \cdot n^2 = O(n^3)$
- $\sum_{i=1}^n i \leq n \cdot n = O(n^2)$
- $\log(n!) = \log(n) + \dots + \log 1 = O(n \log n)$

Big-Oh: More Examples

- $\frac{n^2}{2} - 3n = O(n^2)$
- $1 + 4n = O(n)$
- $\log_{10} n = \frac{\log_2 n}{\log_2 10} = O(\log_2 n) = O(\log n)$
- $\sin n = O(1), 10 = O(1), 10^{10} = O(1)$
- $\sum_{i=1}^n i^2 \leq n \cdot n^2 = O(n^3)$
- $\sum_{i=1}^n i \leq n \cdot n = O(n^2)$
- $\log(n!) = \log(n) + \dots + \log 1 = O(n \log n)$
- $\sum_{i=1}^n \frac{1}{i} = O(\log n)$ (Harmonic Series, 调和级数)

Big-Oh: More Examples

The Asymptotic Upper Bound of Harmonic Series:

$$\sum_{i=1}^n \frac{1}{i} = O(\log n)$$

Big-Oh: More Examples

Proof: Assume that n is a power of two, then

$$\sum_{i=1}^n \frac{1}{n}$$

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

Big-Oh: More Examples

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{i} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \dots + \frac{1}{n/2} + \frac{1}{n}
 \end{aligned}$$

The diagram illustrates the proof by grouping terms in the harmonic series. Red dashed boxes are used to group terms in the second row, showing how they are bounded by a geometric series. The first box groups $\frac{1}{2} + \frac{1}{3}$ and is bounded by $\frac{1}{2}$. The second box groups $\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7}$ and is bounded by $\frac{1}{4}$. The third box groups $\frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots$ and is bounded by $\frac{1}{8}$. The final terms $\frac{1}{n/2}$ and $\frac{1}{n}$ are shown separately.

Big-Oh: More Examples

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{n} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \dots + \frac{1}{n/2} + \frac{1}{n} \\
 &= \frac{1}{1} + 2 \cdot \left(\frac{1}{2}\right) + 4 \cdot \left(\frac{1}{4}\right) + 8 \cdot \left(\frac{1}{8}\right) + \dots + \frac{n}{2} \left(\frac{1}{n/2}\right) + \frac{1}{n}
 \end{aligned}$$

Big-Oh: More Examples

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{n} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &< \frac{1}{1} + \frac{1}{2} + \frac{1}{\textcolor{red}{2}} + \frac{1}{4} + \frac{1}{\textcolor{red}{4}} + \frac{1}{\textcolor{red}{4}} + \frac{1}{\textcolor{red}{4}} + \frac{1}{8} + \frac{1}{\textcolor{red}{8}} + \frac{1}{\textcolor{red}{8}} + \dots + \frac{1}{n/2} + \frac{1}{n} \\
 &= \frac{1}{1} + 2 \cdot \left(\frac{1}{2}\right) + 4 \cdot \left(\frac{1}{4}\right) + 8 \cdot \left(\frac{1}{8}\right) + \dots + \frac{n}{2} \left(\frac{1}{n/2}\right) + \frac{1}{n} \\
 &= 1/n + \sum_{j=0}^{\log n - 1} 1
 \end{aligned}$$

Big-Oh: More Examples

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{i} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \dots + \frac{1}{n/2} + \frac{1}{n} \\
 &= \frac{1}{1} + 2 \cdot \left(\frac{1}{2}\right) + 4 \cdot \left(\frac{1}{4}\right) + 8 \cdot \left(\frac{1}{8}\right) + \dots + \frac{n}{2} \left(\frac{1}{n/2}\right) + \frac{1}{n} \\
 &= 1/n + \sum_{j=0}^{\log n - 1} 1 \\
 &= \log n + \frac{1}{n}
 \end{aligned}$$

Big-Oh: More Examples

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{n} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \dots + \frac{1}{n/2} + \frac{1}{n} \\
 &= \frac{1}{1} + 2 \cdot \left(\frac{1}{2}\right) + 4 \cdot \left(\frac{1}{4}\right) + 8 \cdot \left(\frac{1}{8}\right) + \dots + \frac{n}{2} \left(\frac{1}{n/2}\right) + \frac{1}{n} \\
 &= 1/n + \sum_{j=0}^{\log n - 1} 1 \\
 &= \log n + \frac{1}{n}
 \end{aligned}$$

Thus, $\sum_{i=1}^n \frac{1}{n} = O(\log n)$

Big-Oh: Examples of Complexity Analysis

algorithm scan(v)

1. **for** $i = 1$ **to** n **do**
2. **if** $S[i] = v$ **then**
3. **return** *yes*
4. **return** *no*

Big-Oh: Examples of Complexity Analysis

algorithm `scan(v)`

```
1. for i = 1 to n do  
2.   if S[i] = v then  
3.     return yes  
4. return no
```

} $O(1)$

Big-Oh: Examples of Complexity Analysis

algorithm scan(v)

1. **for** $i = 1$ **to** n **do**
2. **if** $S[i] = v$ **then**
3. **return** *yes*
4. **return** *no*

} $O(1)$

} $n \cdot O(1) = O(n)$

Big-Oh: Examples of Complexity Analysis

algorithm scan(v)

1. **for** $i = 1$ **to** n **do**

2. **if** $S[i] = v$ **then**

3. **return** *yes*

4. **return** *no*

$$\left. \begin{array}{l} \text{Lines 2-3} \end{array} \right\} O(1) \quad \left. \begin{array}{l} \text{Loop} \end{array} \right\} n \cdot O(1) = O(n)$$

Although Lines 2-3 **may be executed less than n times**, we are considering the **worst-case** complexity

Big-Oh: Examples of Complexity Analysis

algorithm CountingInversedPairs($A[1..n]$)

```
1.  $ans = 0$   
2. for  $i = 1$  to  $n$  do  
3.     for  $j = i + 1$  to  $n$  do  
4.         if  $A[i] > A[j]$  then  
5.              $ans = ans + 1$   
6. return  $ans$ 
```

What's the worst-case complexity of this program?

Big-Oh: Examples of Complexity Analysis

algorithm CountingInversedPairs($A[1..n]$)

1. $ans = 0$
2. **for** $i = 1$ **to** n **do**
3. **for** $j = i + 1$ **to** n **do**
4. **if** $A[i] > A[j]$ **then**
5. $ans = ans + 1$
6. **return** ans

Big-Oh: Examples of Complexity Analysis

algorithm CountingInversedPairs($A[1..n]$)

```
1.  $ans = 0$   
2. for  $i = 1$  to  $n$  do  
3.   for  $j = i + 1$  to  $n$  do  
4.     if  $A[i] > A[j]$  then  
5.        $ans = ans + 1$   
6. return  $ans$ 
```

} $O(1)$

Big-Oh: Examples of Complexity Analysis

algorithm CountingInversedPairs($A[1..n]$)

```
1.  $ans = 0$   
2. for  $i = 1$  to  $n$  do  
3.   for  $j = i + 1$  to  $n$  do  
4.     if  $A[i] > A[j]$  then  
5.        $ans = ans + 1$   
6. return  $ans$ 
```

$\left. \begin{array}{l} \text{ } \end{array} \right\} O(1) \left. \vphantom{\begin{array}{l} \text{ } \end{array}} \right\} (n-i)O(1)$

Big-Oh: Examples of Complexity Analysis

algorithm CountingInversedPairs($A[1..n]$)

```

1.  $ans = 0$ 
2. for  $i = 1$  to  $n$  do
3.     for  $j = i + 1$  to  $n$  do
4.         if  $A[i] > A[j]$  then
5.              $ans = ans + 1$ 
6. return  $ans$ 

```

$\left. \begin{array}{l} \text{ } \end{array} \right\} O(1)$
 $\left. \begin{array}{l} \text{ } \end{array} \right\} (n - i)O(1)$
 $\left. \begin{array}{l} \text{ } \end{array} \right\} ??$

Big-Oh: Examples of Complexity Analysis

algorithm CountingInversedPairs($A[1..n]$)

```

1. ans = 0
2. for  $i = 1$  to  $n$  do
3.     for  $j = i + 1$  to  $n$  do
4.         if  $A[i] > A[j]$  then
5.              $ans = ans + 1$ 
6. return ans

```

$\left. \begin{array}{l} \text{4.} \\ \text{5.} \end{array} \right\} O(1)$
 $\left. \begin{array}{l} \text{3.} \\ \text{4.} \\ \text{5.} \end{array} \right\} (n - i)O(1)$
 $\left. \begin{array}{l} \text{2.} \\ \text{3.} \\ \text{4.} \\ \text{5.} \end{array} \right\} ??$

$$\begin{aligned}
 ?? &= (n - 1)O(1) + (n - 2)O(1) + \cdots + (n - n)O(1) \\
 &= n(n - 1)O(1) \\
 &= O(n^2)
 \end{aligned}$$

Outline

- Asymptotic Notations (渐近记号)
 - Big-Oh
 - Big-Omega
 - Big-Theta
 - Algorithm Design and Algorithm Turing
- Solving Recurrences
 - Recursion-tree Method (递归树法)
 - Substitution Method (代入法/替代法)
 - Master Method and Master Theorem (主方法)

Big-Omega

Asymptotic lower bound

Definition (big-Omega)

$f(n) = \Omega(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

Big-Omega

Asymptotic lower bound

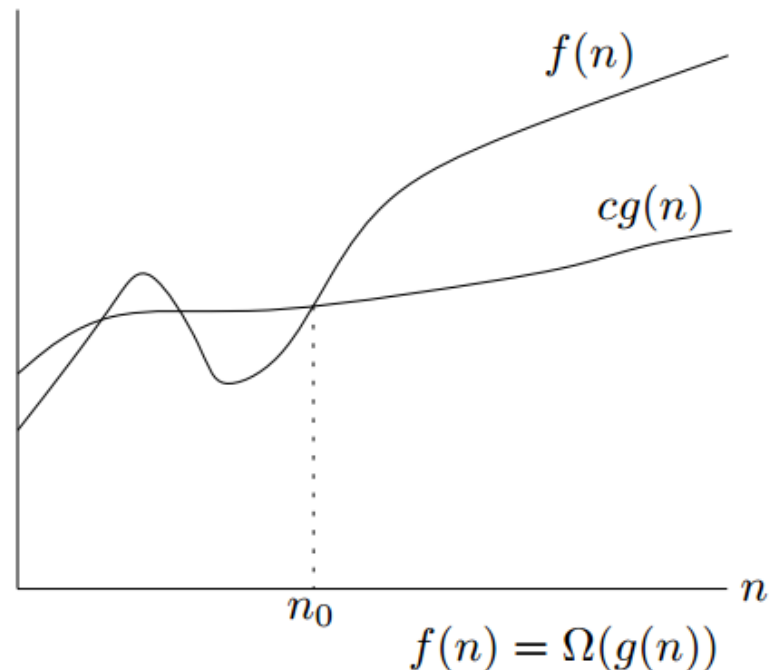
Definition (big-Omega)

$f(n) = \Omega(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

It is easy to show that

$$\frac{n^2}{2} - 3n \geq \frac{n^2}{4} \quad \text{for all } n \geq 12.$$

Thus, $n^2/2 - 3n = \Omega(n^2)$.



Big-Omega

Asymptotic lower bound

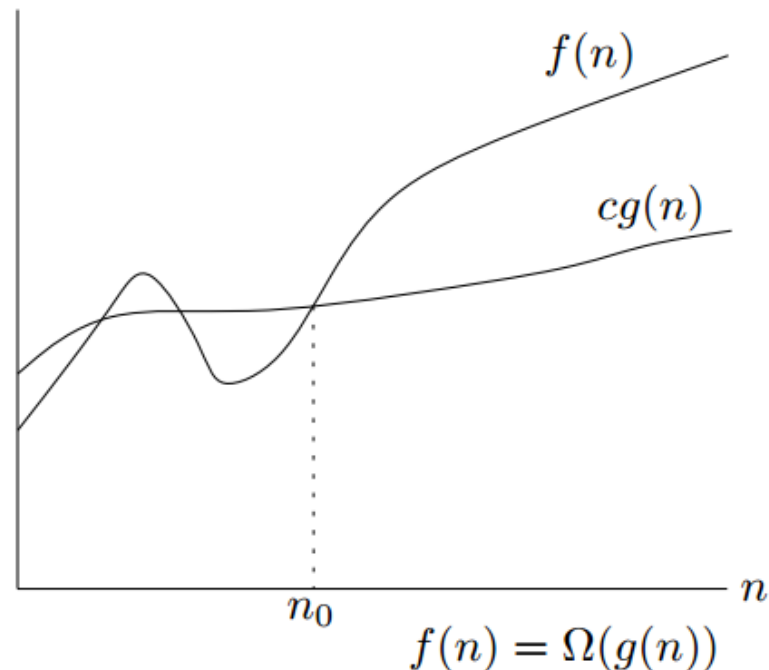
Definition (big-Omega)

$f(n) = \Omega(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

It is easy to show that

$$\frac{n^2}{2} - 3n \geq \frac{n^2}{4} \quad \text{for all } n \geq 12.$$

Thus, $n^2/2 - 3n = \Omega(n^2)$.



Example

$$\log(n!) = \log(n) + \log(n-1) + \dots + \log 1$$

Big-Omega

Asymptotic lower bound

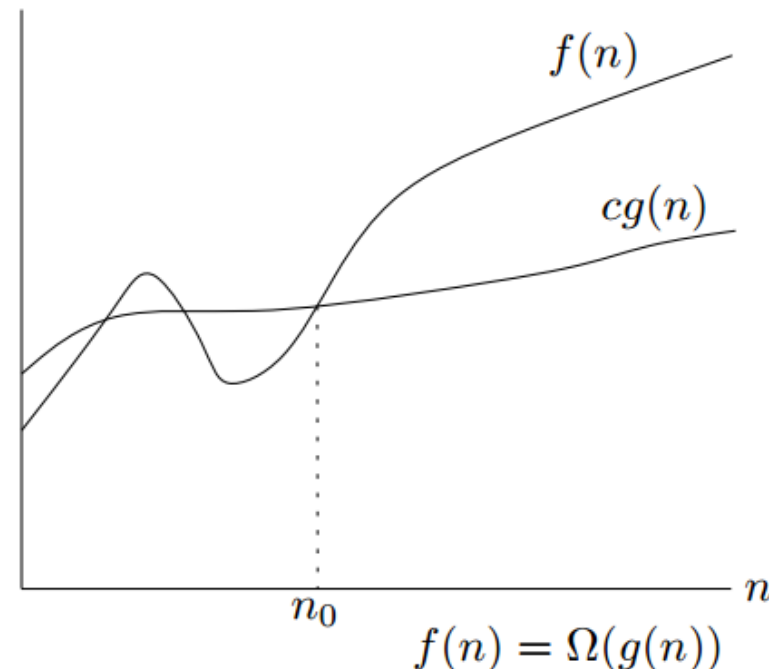
Definition (big-Omega)

$f(n) = \Omega(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

It is easy to show that

$$\frac{n^2}{2} - 3n \geq \frac{n^2}{4} \quad \text{for all } n \geq 12.$$

Thus, $n^2/2 - 3n = \Omega(n^2)$.



Example

$$\begin{aligned} \log(n!) &= \log(n) + \log(n-1) + \dots + \log 1 \\ &\geq \log(n) + \log(n-1) + \dots + \log(n/2) \end{aligned}$$

Big-Omega

Asymptotic lower bound

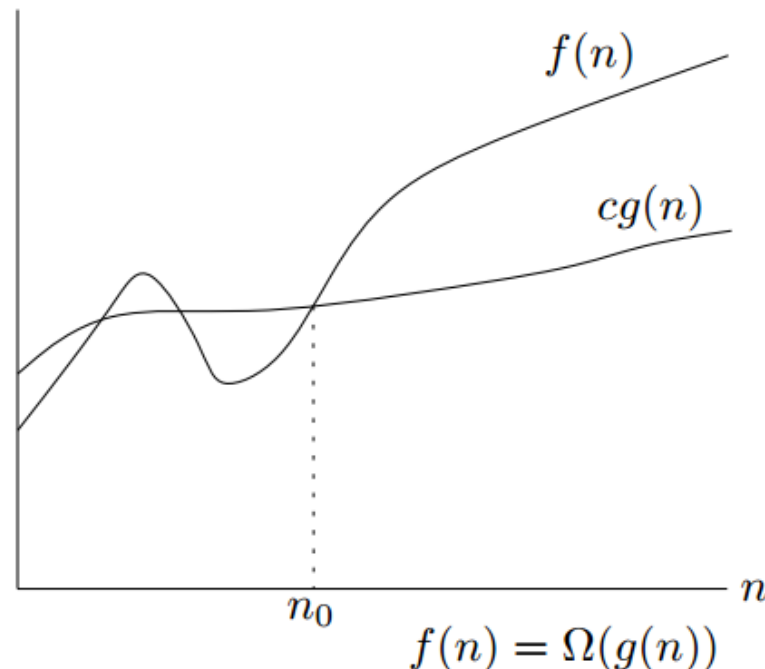
Definition (big-Omega)

$f(n) = \Omega(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

It is easy to show that

$$\frac{n^2}{2} - 3n \geq \frac{n^2}{4} \quad \text{for all } n \geq 12.$$

Thus, $n^2/2 - 3n = \Omega(n^2)$.



Example

$$\begin{aligned} \log(n!) &= \log(n) + \log(n-1) + \dots + \log 1 \\ &\geq \log(n) + \log(n-1) + \dots + \log(n/2) \\ &\geq n/2 \cdot \log(n/2) \end{aligned}$$

Big-Omega

Asymptotic lower bound

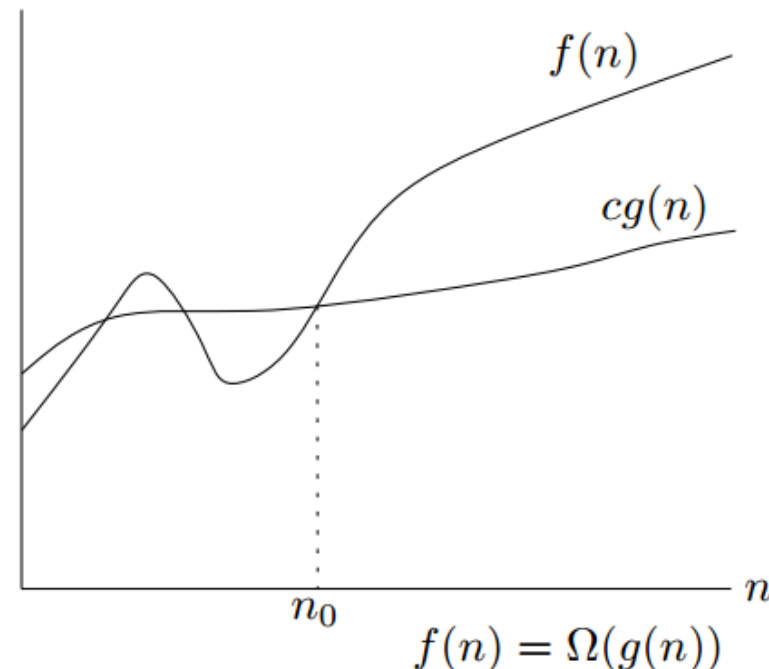
Definition (big-Omega)

$f(n) = \Omega(g(n))$: There exists constant $c > 0$ and n_0 such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$.

It is easy to show that

$$\frac{n^2}{2} - 3n \geq \frac{n^2}{4} \quad \text{for all } n \geq 12.$$

Thus, $n^2/2 - 3n = \Omega(n^2)$.



Example

$$\begin{aligned} \log(n!) &= \log(n) + \log(n-1) + \dots + \log 1 \\ &\geq \log(n) + \log(n-1) + \dots + \log(n/2) \\ &\geq n/2 \cdot \log(n/2) \\ &= n/2 \cdot (\log n - 1) = \Omega(n \log n). \end{aligned}$$

Big-Omega: Harmonic Series

The Asymptotic Lower Bound of Harmonic Series:

$$\sum_{i=1}^n \frac{1}{i} = \Omega(\log n)$$

Big-Omega: Harmonic Series

Proof: Assume that n is a power of two, then

$$\sum_{i=1}^n \frac{1}{i}$$

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

Big-Omega: Harmonic Series

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{i} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \dots + \frac{1}{n}
 \end{aligned}$$

Big-Omega: Harmonic Series

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{i} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \dots + \frac{1}{n} \\
 &= \frac{1}{1} + \frac{1}{2} + 2 \cdot \left(\frac{1}{4}\right) + 4 \cdot \left(\frac{1}{8}\right) + 8 \cdot \left(\frac{1}{16}\right) + \dots + \frac{n}{2} \left(\frac{1}{n}\right)
 \end{aligned}$$

Big-Omega: Harmonic Series

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{i} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \dots + \frac{1}{n} \\
 &= \frac{1}{1} + \frac{1}{2} + 2 \cdot \left(\frac{1}{4}\right) + 4 \cdot \left(\frac{1}{8}\right) + 8 \cdot \left(\frac{1}{16}\right) + \dots + \frac{n}{2} \left(\frac{1}{n}\right) \\
 &= 1 + \sum_{j=1}^{\log n} \frac{1}{2}
 \end{aligned}$$

Big-Omega: Harmonic Series

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{i} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \dots + \frac{1}{n} \\
 &= \frac{1}{1} + \frac{1}{2} + 2 \cdot \left(\frac{1}{4}\right) + 4 \cdot \left(\frac{1}{8}\right) + 8 \cdot \left(\frac{1}{16}\right) + \dots + \frac{n}{2} \left(\frac{1}{n}\right) \\
 &= 1 + \sum_{j=1}^{\log n} \frac{1}{2} \\
 &= 1 + \frac{1}{2} \log n
 \end{aligned}$$

Big-Omega: Harmonic Series

Proof: Assume that n is a power of two, then

$$\begin{aligned}
 & \sum_{i=1}^n \frac{1}{i} \\
 &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \\
 &> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \dots + \frac{1}{n} \\
 &= \frac{1}{1} + \frac{1}{2} + 2 \cdot \left(\frac{1}{4}\right) + 4 \cdot \left(\frac{1}{8}\right) + 8 \cdot \left(\frac{1}{16}\right) + \dots + \frac{n}{2} \left(\frac{1}{n}\right) \\
 &= 1 + \sum_{j=1}^{\log n} \frac{1}{2^j} \\
 &= 1 + \frac{1}{2} \log n
 \end{aligned}$$

Thus, $\sum_{i=1}^n \frac{1}{i} = \Omega(\log n)$

Outline

- Asymptotic Notations (渐近记号)
 - Big-Oh
 - Big-Omega
 - Big-Theta
 - Algorithm Design and Algorithm Turing
- Solving Recurrences
 - Recursion-tree Method (递归树法)
 - Substitution Method (代入法/替代法)
 - Master Method and Master Theorem (主方法)

Big-Theta

Asymptotic tight bound

Definition (big-Theta)

$f(n) = \Theta(g(n))$: $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Big-Theta

Asymptotic tight bound

Definition (big-Theta)

$f(n) = \Theta(g(n))$: $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

We have shown that

$$n^2/2 - 3n = O(n^2),$$

and

$$n^2/2 - 3n = \Omega(n^2).$$

Therefore, we have that $n^2/2 - 3n = \Theta(n^2)$.

Big-Theta

Asymptotic tight bound

Definition (big-Theta)

$f(n) = \Theta(g(n))$: $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

We have shown that

$$n^2/2 - 3n = O(n^2),$$

and

$$n^2/2 - 3n = \Omega(n^2).$$

Therefore, we have that $n^2/2 - 3n = \Theta(n^2)$.

Usually (and in this course), it is sufficient to show only upper bounds (big-Oh), though we should try to make these as tight as we can.

Asymptotic Notations

Upper bounds. $T(n)=O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $T(n) \leq c \cdot f(n)$.

Equivalent definition: $\lim_{n \rightarrow \infty} \frac{T(n)}{f(n)} < \infty$.

Lower bounds. $T(n)=\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $T(n) \geq c \cdot f(n)$.

Equivalent definition: $\lim_{n \rightarrow \infty} \frac{T(n)}{f(n)} > 0$.

Tight bounds. $T(n) = \Theta(f(n))$ if $T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$.

Note: Here “=” means “is”, not equal. The more mathematically correct way should be $T(n) \in O(f(n))$.

For example, for the harmonic series,

we have: $\sum_{i=1}^n \frac{1}{i} = O(\log n) = \Omega(\log n) = \Theta(\log n)$

Examples

- $100n^2 = O(n^3)$?
- $100n^2 = \Omega(n^3)$?
- $10n^2 - 100n = O(n^2)$?
- $10n^2 - 100n = \Omega(n^2)$?
- $10n^2 - 100n = \Theta(n^2)$?
- $\log(2n) = O(\log n)$?
- $(2n)^{10} = O(n^{10})$?
- $2^{2n} = O(2^n)$?

Examples

- $100n^2 = O(n^3)$?

Answer: Yes. $C = 1$ and $n_0 = 100$. Then $\forall n \geq n_0$,
 $100n^2 \leq C \cdot n^3$.

- $100n^2 = \Omega(n^3)$?

- $10n^2 - 100n = O(n^2)$?

- $10n^2 - 100n = \Omega(n^2)$?

- $10n^2 - 100n = \Theta(n^2)$?

- $\log(2n) = O(\log n)$?

- $(2n)^{10} = O(n^{10})$?

- $2^{2n} = O(2^n)$?

Examples

- $100n^2 = O(n^3)$?

Answer: Yes. $C = 1$ and $n_0 = 100$. Then $\forall n \geq n_0$, $100n^2 \leq C \cdot n^3$.

- $100n^2 = \Omega(n^3)$?

Answer: No. $\forall C > 0$, $n_0 > 0$, there exists $n > n_0$ ($n = n_0 + 100/C$) such that $100n^2 < C \cdot n^3$.

- $10n^2 - 100n = O(n^2)$?
- $10n^2 - 100n = \Omega(n^2)$?
- $10n^2 - 100n = \Theta(n^2)$?
- $\log(2n) = O(\log n)$?
- $(2n)^{10} = O(n^{10})$?
- $2^{2n} = O(2^n)$?

Examples

- $100n^2 = O(n^3)$?

Answer: Yes. $C = 1$ and $n_0 = 100$. Then $\forall n \geq n_0$, $100n^2 \leq C \cdot n^3$.

- $100n^2 = \Omega(n^3)$?

Answer: No. $\forall C > 0$, $n_0 > 0$, there exists $n > n_0$ ($n = n_0 + 100/C$) such that $100n^2 < C \cdot n^3$.

- $10n^2 - 100n = O(n^2)$? ✓

- $10n^2 - 100n = \Omega(n^2)$? ✓

- $10n^2 - 100n = \Theta(n^2)$? ✓

- $\log(2n) = O(\log n)$? ✓

- $(2n)^{10} = O(n^{10})$? ✓

- $2^{2n} = O(2^n)$? ✗

Solutions

- $10n^2 - 100n = O(n^2)$?
- $10n^2 - 100n = \Omega(n^2)$?
- $10n^2 - 100n = \Theta(n^2)$?

Solutions

- $10n^2 - 100n = O(n^2)$?

Answer: Yes. $\forall n > 0, 10n^2 - 100n \leq 10n^2$.

- $10n^2 - 100n = \Omega(n^2)$?

- $10n^2 - 100n = \Theta(n^2)$?

Solutions

- $10n^2 - 100n = O(n^2)$?

Answer: Yes. $\forall n > 0, 10n^2 - 100n \leq 10n^2$.

- $10n^2 - 100n = \Omega(n^2)$?

Answer: Yes. $\forall n \geq 20, 10n^2 - 100n \geq 5n^2$.

- $10n^2 - 100n = \Theta(n^2)$?

Solutions

- $10n^2 - 100n = O(n^2)$?

Answer: Yes. $\forall n > 0, 10n^2 - 100n \leq 10n^2$.

- $10n^2 - 100n = \Omega(n^2)$?

Answer: Yes. $\forall n \geq 20, 10n^2 - 100n \geq 5n^2$.

- $10n^2 - 100n = \Theta(n^2)$?

Answer: Yes. Because

$$10n^2 - 100n = O(n^2)$$

and

$$10n^2 - 100n = \Omega(n^2)$$

Solutions

- $\log(2n) = O(\log n)$?
- $(2n)^{10} = O(n^{10})$?
- $2^{2n} = O(2^n)$?

Solutions

- $\log(2n) = O(\log n)$?

Answer: Yes. $\forall n \geq 2, \log(2n) = \log n + 1 \leq 2 \log n$.

- $(2n)^{10} = O(n^{10})$?

- $2^{2n} = O(2^n)$?

Solutions

- $\log(2n) = O(\log n)$?

Answer: Yes. $\forall n \geq 2, \log(2n) = \log n + 1 \leq 2 \log n$.

- $(2n)^{10} = O(n^{10})$?

Answer: **Yes.** $\forall n > 0, (2n)^{10} = (2^{10})(n^{10})$.

- $2^{2n} = O(2^n)$?

Solutions

- $\log(2n) = O(\log n)$?

Answer: Yes. $\forall n \geq 2, \log(2n) = \log n + 1 \leq 2 \log n$.

- $(2n)^{10} = O(n^{10})$?

Answer: Yes. $\forall n > 0, (2n)^{10} = (2^{10})(n^{10})$.

- $2^{2n} = O(2^n)$?

Answer: No. $\forall C > 0, n_0 > 0$, let $n = n_0 + \log C > n_0$.
Then $2^{2n} = 2^n \cdot 2^n > C \cdot 2^n$.

An interesting fact about logarithm

$$\log_{b_1} n = O(\log_{b_2} n)$$

For any constant $b_1 > 1$ and $b_2 > 1$.

For example, let us verify $\log_2 n = O(\log_3 n)$.

Notice that

$$\log_3 n = \frac{\log_2 n}{\log_2 3} \Rightarrow \log_2 n = \log_2 3 \cdot \log_3 n$$

Hence, we can set $c_1 = \log_2 3$ and $c_2 = 1$, which makes

$$\log_2 n \leq c_1 \log_3 n$$

Hold for all $n \geq c_2$.

An interesting fact about logarithm

$$\log_{b_1} n = O(\log_{b_2} n)$$

For any constant $b_1 > 1$ and $b_2 > 1$.

Because of the above, in computer science, we omit all the constant logarithm bases in big-O. For example, instead of $O(\log_2 n)$, we will simply write $O(\log n)$

- Essentially, this says that "you are welcome to put any constant base there, and it will be the same asymptotically".
- Obviously, Ω , Θ also have this property.

Outline

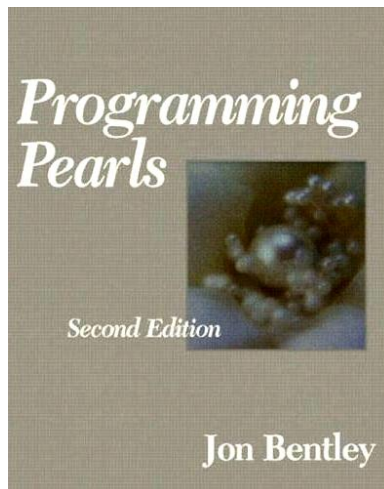
- Asymptotic Notations (渐近记号)
 - Big-Oh
 - Big-Omega
 - Big-Theta
 - Algorithm Design and Algorithm Turing
- Solving Recurrences
 - Recursion-tree Method (递归树法)
 - Substitution Method (代入法/替代法)
 - Master Method and Master Theorem (主方法)

Some Thoughts on Algorithm Design

- **Algorithm Design**, as taught in this class, is mainly about designing algorithms that have **small big-Oh running times**.
- As n gets larger and larger, $O(n \log n)$ algorithms will run faster than $O(n^2)$ ones and $O(n)$ algorithms will beat $O(n \log n)$ ones.
- Good algorithm design & analysis allows you to identify the **hard parts** of your problem and deal with them effectively.
- Too often, programmers try to solve problems using brute force techniques and end up with slow complicated code!
- A few hours of abstract thought devoted to algorithm design often results in **faster**, **simpler**, and **more general** solutions.

Algorithm Tuning

- After algorithm design one can continue on to **Algorithm tuning**
 - concentrate on improving algorithms by **cutting down on the constants** in the big $O()$ bounds.
 - needs a good understanding of both **algorithm design principles** and efficient use of **data structures**.
- In this course we will not go further into algorithm tuning
 - For a good introduction, see chapter 9 in **Programming Pearls, 2nd ed** by Jon Bentley



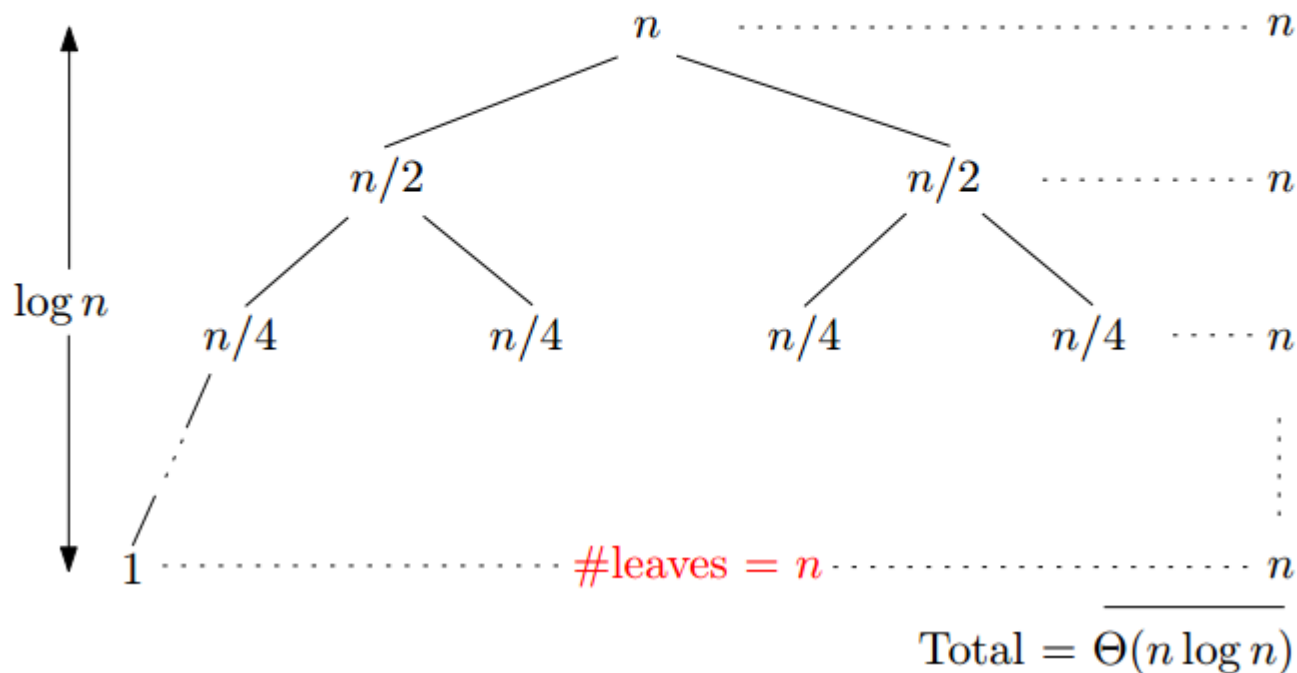
Outline

- Asymptotic Notations (渐近记号)
 - Big-Oh
 - Big-Omega
 - Big-Theta
 - Algorithm Design and Algorithm Turing
- Solving Recurrences
 - Recursion-tree Method (递归树法)
 - Substitution Method (代入法/替代法)
 - Master Method and Master Theorem (主方法)

Solving recurrences: Recursion-tree method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
 - Each node represents the cost of a single subproblem.

$$T(n) = \begin{cases} 2T(n/2) + n, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$



Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

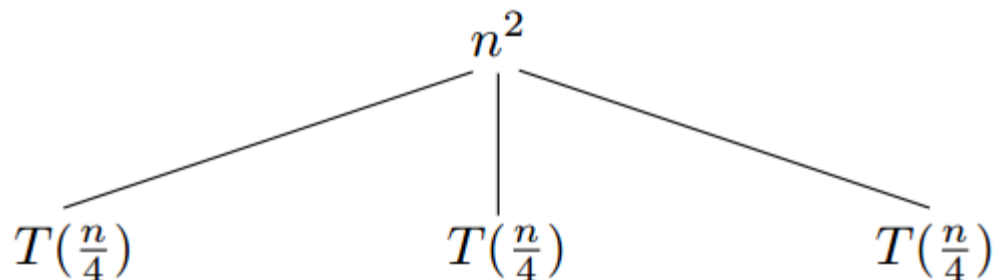
Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

$$T(n)$$

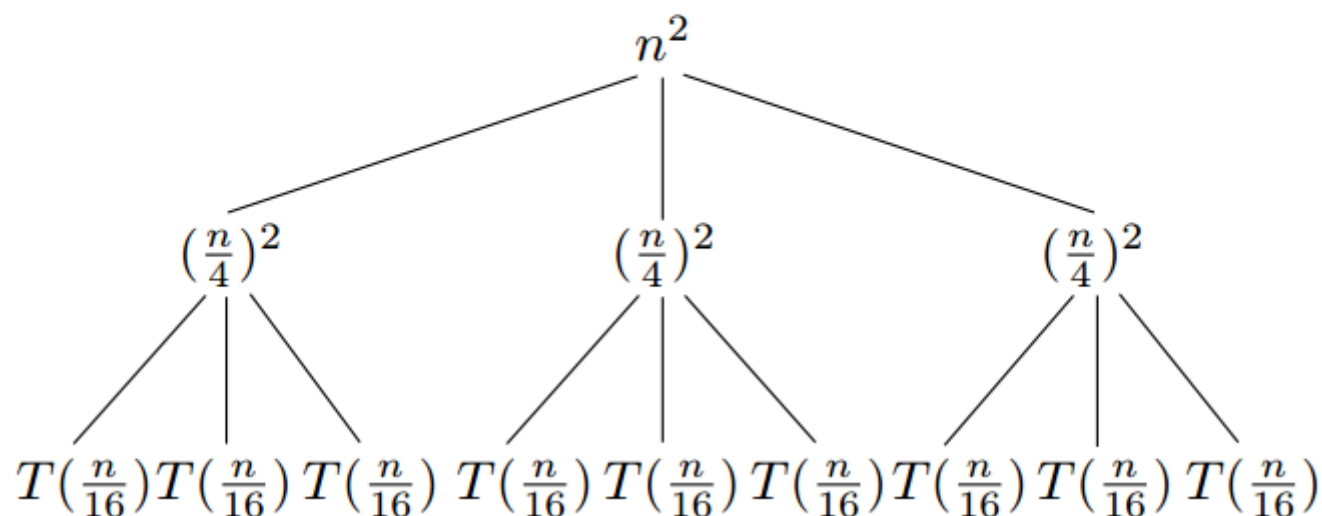
Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$



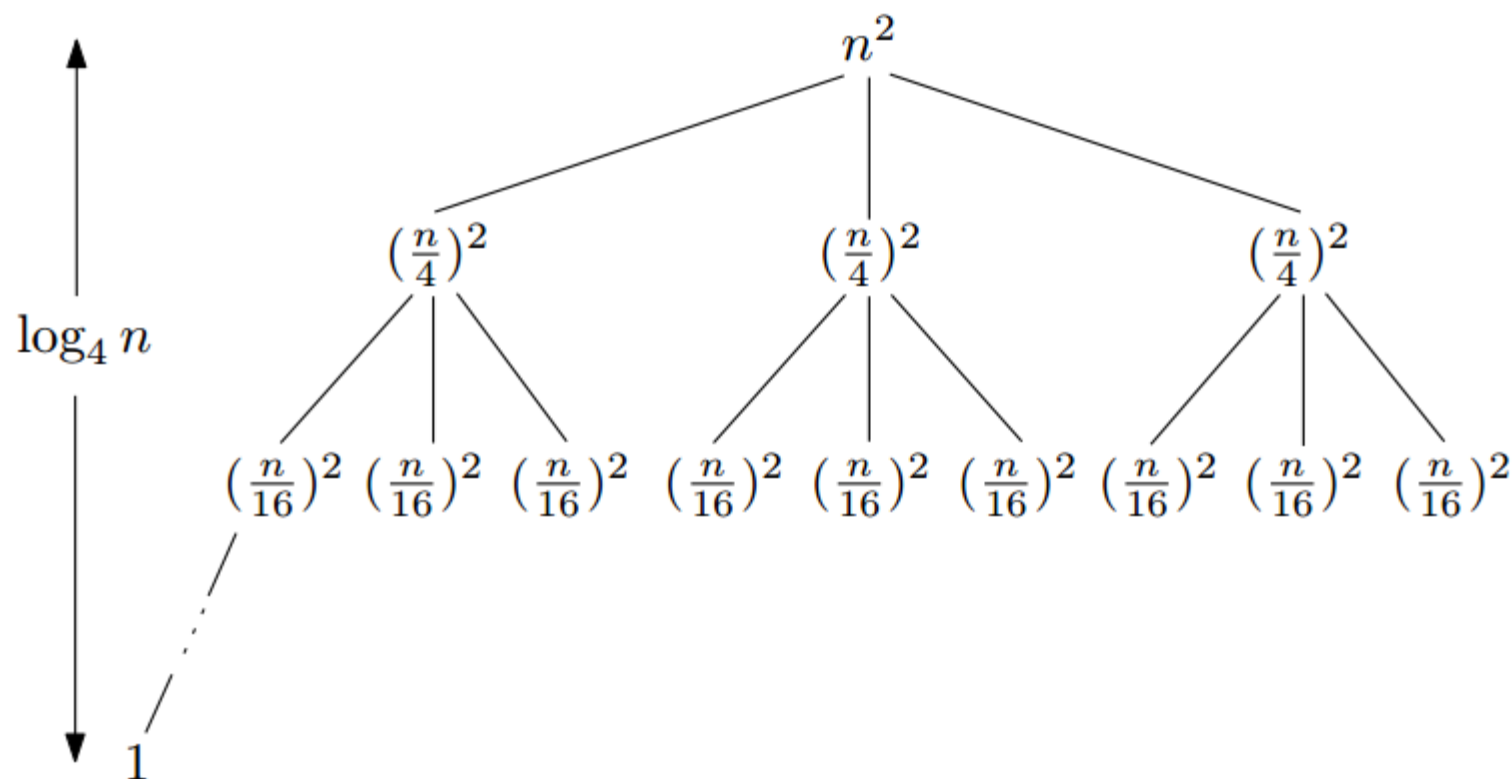
Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$



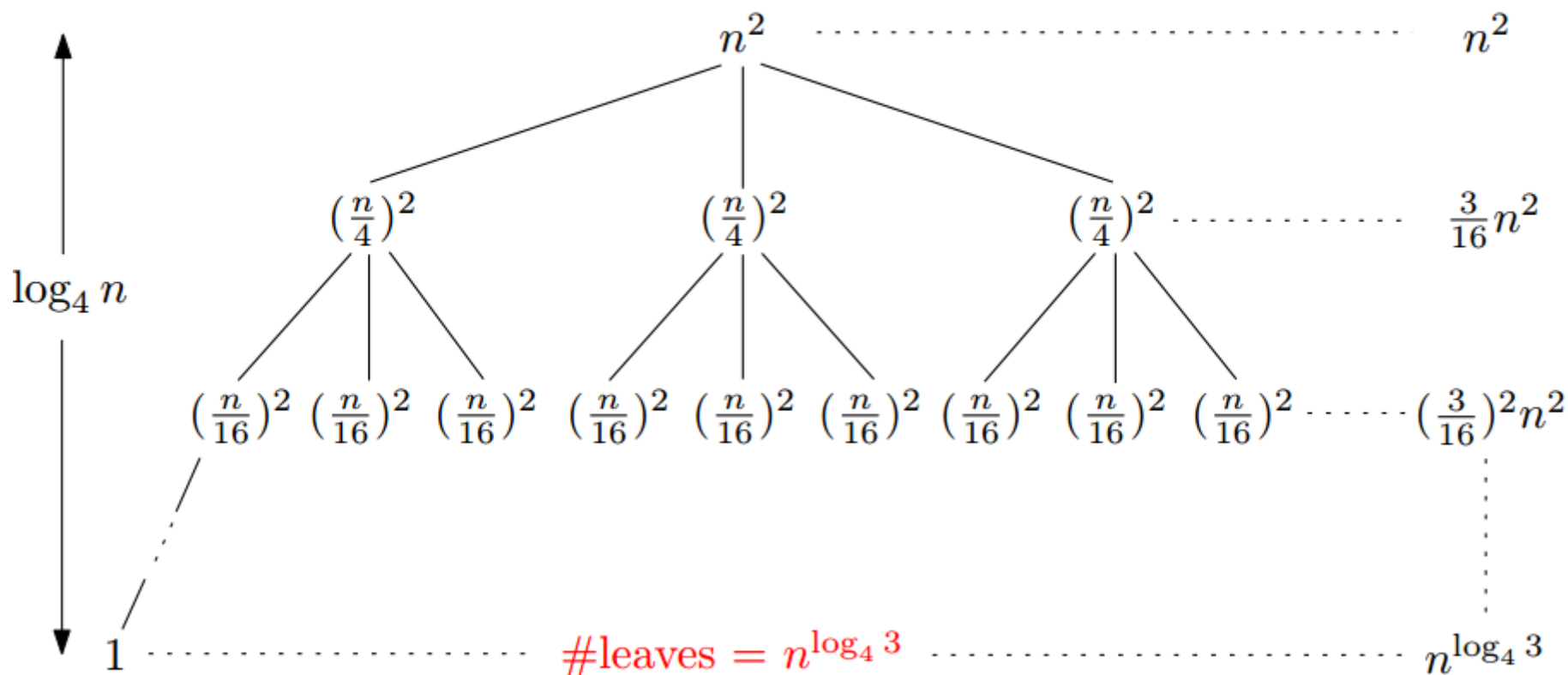
Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$



Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$



对数技巧: $a^{\log_b n} = n^{\log_b a}$,
 $3^{\log_4 n} = n^{\log_4 3}$

Total = $\Theta(n^2)$

Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

$$\begin{aligned} T(n) &\leq n^2 + \frac{3}{16}n^2 + \left(\frac{3}{16}\right)^2 n^2 + \dots \\ &= O(n^2). \quad \text{geometric series} \end{aligned}$$



几何级数(又称为等比级数)

Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

$$\begin{aligned} T(n) &\leq n^2 + \frac{3}{16}n^2 + \left(\frac{3}{16}\right)^2 n^2 + \dots \\ &= O(n^2). \quad \text{geometric series} \end{aligned}$$

- Since $T(n) = 3T(n/4) + n^2$, it follows that $T(n) \geq n^2$

Recursion-tree method: Example

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

$$\begin{aligned} T(n) &\leq n^2 + \frac{3}{16}n^2 + \left(\frac{3}{16}\right)^2 n^2 + \dots \\ &= O(n^2). \quad \text{geometric series} \end{aligned}$$

- Since $T(n) = 3T(n/4) + n^2$, it follows that $T(n) \geq n^2$
- So, $T(n) = \Omega(n^2)$.
- Thus, $T(n) = \Theta(n^2)$

Outline

- Asymptotic Notations (渐近记号)
 - Big-Oh
 - Big-Omega
 - Big-Theta
 - Algorithm Design and Algorithm Turing
- Solving Recurrences
 - Recursion-tree Method (递归树法)
 - Substitution Method (代入法/替代法)
 - Master Method and Master Theorem (主方法)

Substitution method: Example 1

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

Prove $T(n) \leq cn^2$ by induction, where c is a large constant.

Substitution method: Example 1

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

Prove $T(n) \leq cn^2$ by induction, where c is a large constant.

Proof.

- Base ($n=1$) : obviously holds for any $c \geq 1$



Substitution method: Example 1

$$T(n) = \begin{cases} 3T(n/4) + n^2, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

Prove $T(n) \leq cn^2$ by induction, where c is a large constant.

Proof.

- Base ($n=1$) : obviously holds for any $c \geq 1$
- Induction:

$$\begin{aligned} T(n) &= 3T(n/4) + n^2 \\ &\leq 3c(n/4)^2 + n^2 \\ &= cn^2 - (13c/16 - 1)n^2 \\ &\leq cn^2, \end{aligned}$$

whenever $13c/16 - 1 \geq 0$, or $c \geq 16/13$.



Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$

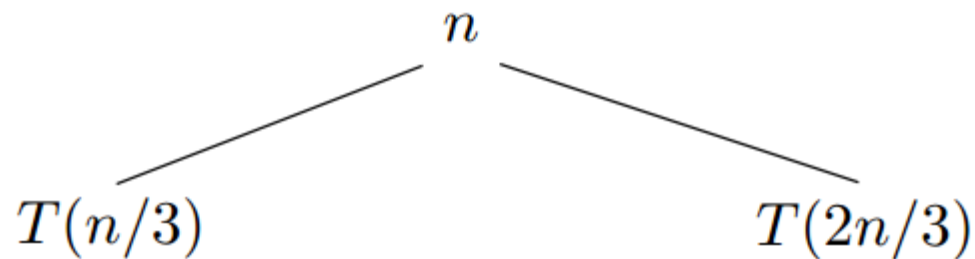
Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$

$$T(n)$$

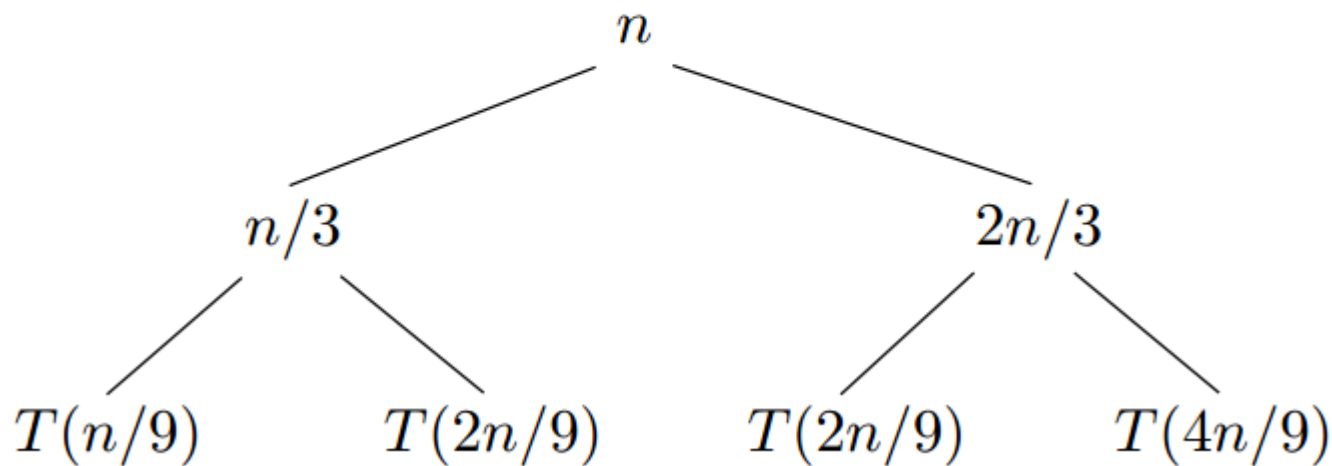
Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$



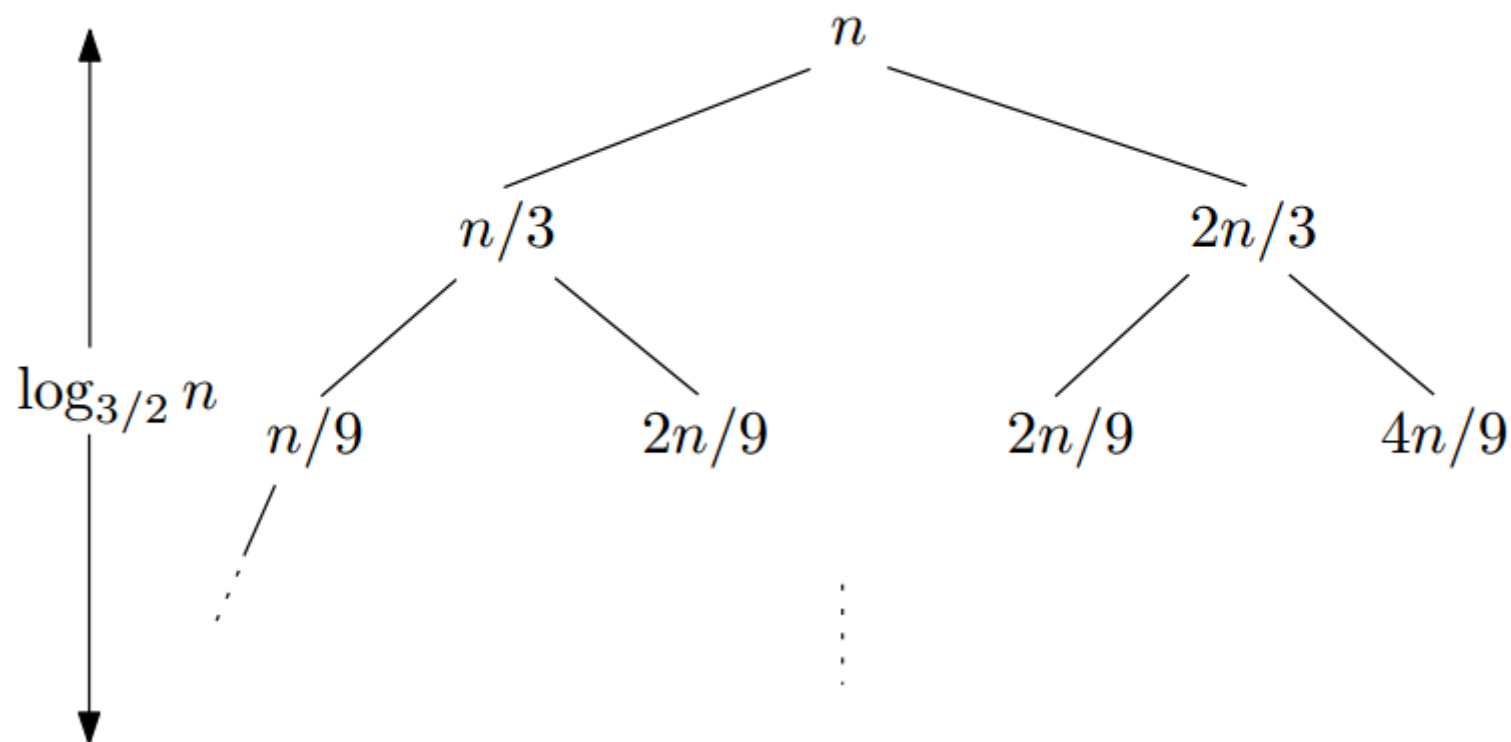
Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$



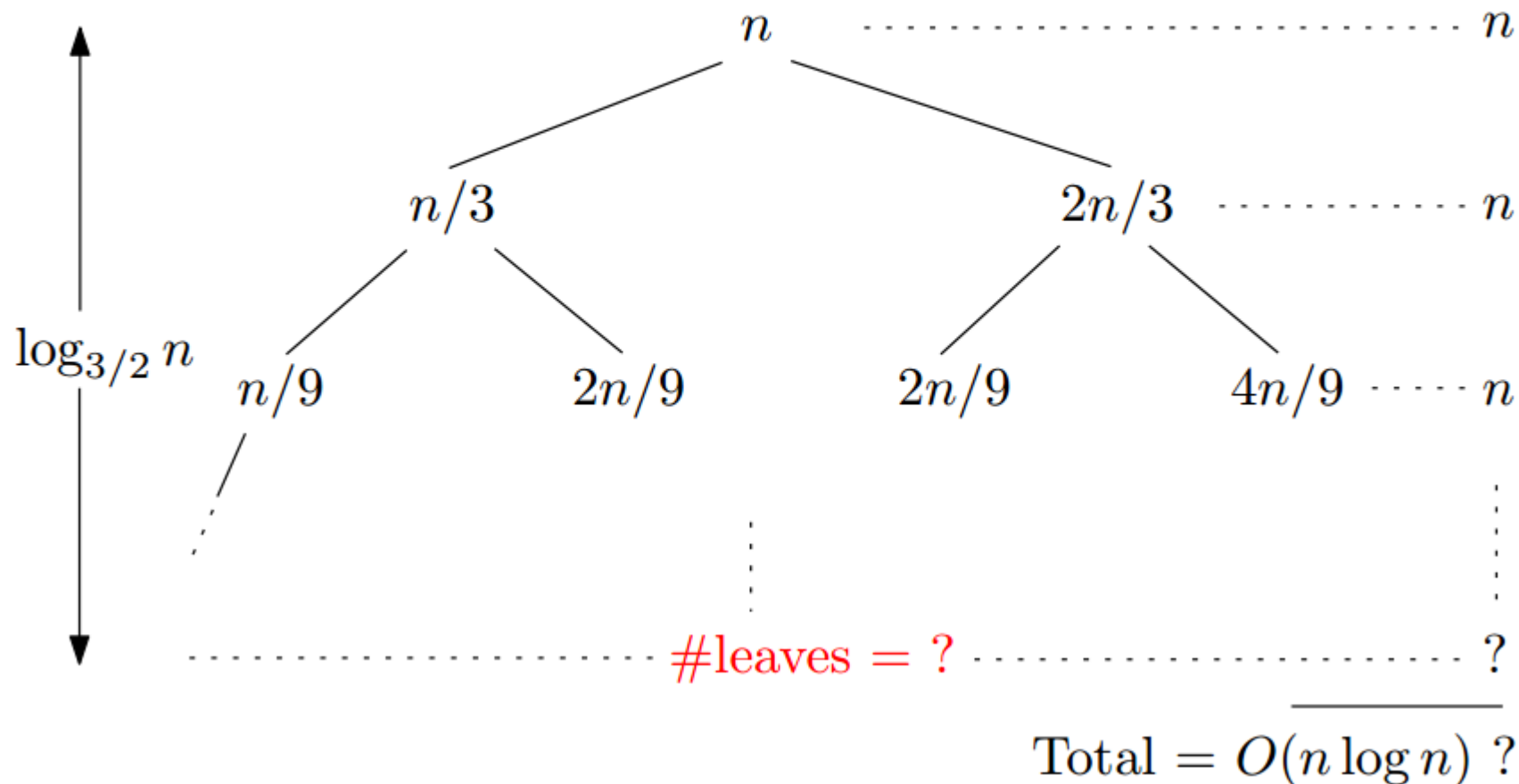
Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$



Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$



Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$

Prove $T(n) \leq cn \log n$ by induction, where c is a large constant.

Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$

Prove $T(n) \leq cn \log n$ by induction, where c is a large constant.

Proof.

- Base ($n=2$) : obviously holds for any $c \geq 1/2$



Substitution method: Example 2

$$T(n) = \begin{cases} T(n/3) + T(2n/3) + n, & \text{if } n > 2, \\ 1, & \text{if } n = 1, 2. \end{cases}$$

Prove $T(n) \leq cn \log n$ by induction, where c is a large constant.

Proof.

- Base ($n=2$) : obviously holds for any $c \geq 1/2$
- Induction:

$$\begin{aligned} T(n) &= T(n/3) + T(2n/3) + n \\ &\leq c(n/3) \log(n/3) + c(2n/3) \log(2n/3) + n \\ &= cn \log n - c((n/3) \log 3 + (2n/3) \log(3/2)) + n \\ &= cn \log n - cn(\log 3 - 2/3) + n \\ &\leq cn \log n, \end{aligned}$$

as long as $c \geq 1/(\log 3 - 2/3)$.



Outline

- Asymptotic Notations (渐近记号)
 - Big-Oh
 - Big-Omega
 - Big-Theta
 - Algorithm Design and Algorithm Turing
- Solving Recurrences
 - Recursion-tree Method (递归树法)
 - Substitution Method (代入法/替代法)
 - Master Method and Master Theorem (主方法)

Master Theorem

If $T(n) = aT\left(\left\lceil\frac{n}{b}\right\rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

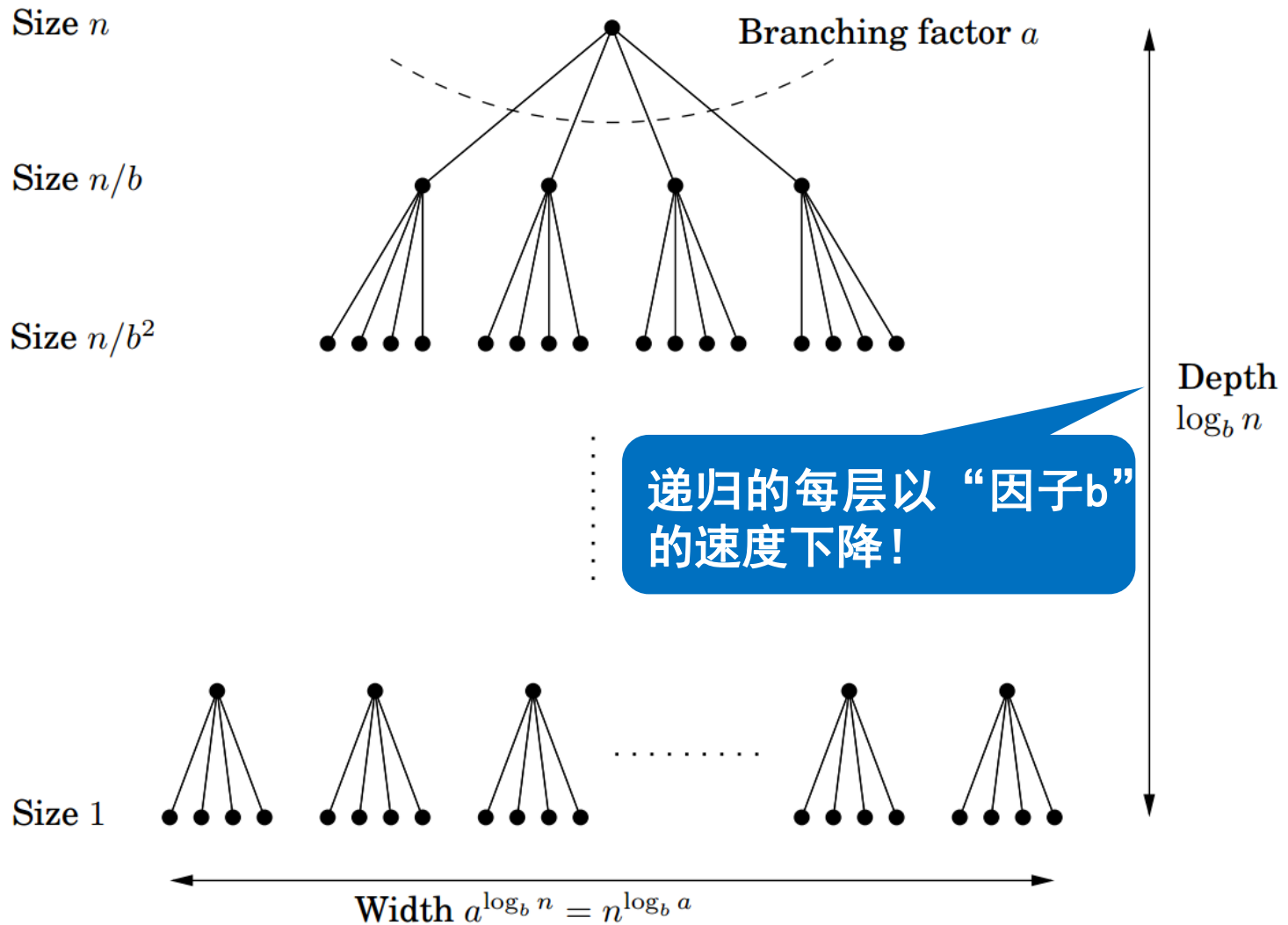
Proof of the Master Theorem

If $T(n) = aT\left(\left\lceil\frac{n}{b}\right\rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

For the sake of convenience, we assume that n is a power of b . This will not influence the final bound in any important way— n is **at most a multiplicative factor of b** away from some power of b —and it will allow us to ignore the rounding effect in $\lceil\frac{n}{b}\rceil$.

Proof of the Master Theorem



$$a^k \times O\left(\frac{n}{b^k}\right)^d = O(n^d) \times \left(\frac{a}{b^d}\right)^k$$

Proof of the Master Theorem

If $T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

- The size of the subproblems decreases by a factor of b with each level of recursion, and therefore reaches the base case after **$\log_b n$ levels**. This is the height of the recursion tree.
- The k -th level of the tree is made up of **a^k subproblems**, each of size **n/b^k**
- The total work done at this level is

$$a^k \times O\left(\frac{n}{b^k}\right)^d = O(n^d) \times \left(\frac{a}{b^d}\right)^k$$

Proof of the Master theorem

It comes down to the following three cases.

- **The ratio a/b^d is less than 1 ($a/b^d < 1$).**
Then the series is decreasing, and its sum is just given by its first term, $O(n^d)$.

Proof of the Master theorem

It comes down to the following three cases.

- **The ratio a/b^d is less than 1 ($a/b^d < 1$).**

Then the series is decreasing, and its sum is just given by its first term, $O(n^d)$.

- **The ratio a/b^d is greater than 1 ($a/b^d > 1$).**

The series is increasing and its sum is given by its last term, $O(n^{\log_b a})$:

$$n^d \left(\frac{a}{b^d} \right)^{\log_b n} =$$

Proof of the Master theorem

It comes down to the following three cases.

- **The ratio a/b^d is less than 1 ($a/b^d < 1$).**

Then the series is decreasing, and its sum is just given by its first term, $O(n^d)$.

- **The ratio a/b^d is greater than 1 ($a/b^d > 1$).**

The series is increasing and its sum is given by its last term, $O(n^{\log_b a})$:

$$n^d \left(\frac{a}{b^d} \right)^{\log_b n} = n^d \left(\frac{a^{\log_b n}}{(b^{\log_b n})^d} \right) =$$

Proof of the Master theorem

It comes down to the following three cases.

- **The ratio a/b^d is less than 1 ($a/b^d < 1$).**

Then the series is decreasing, and its sum is just given by its first term, $O(n^d)$.

- **The ratio a/b^d is greater than 1 ($a/b^d > 1$).**

The series is increasing and its sum is given by its last term, $O(n^{\log_b a})$:

$$n^d \left(\frac{a}{b^d} \right)^{\log_b n} = n^d \left(\frac{a^{\log_b n}}{(b^{\log_b n})^d} \right) = a^{\log_b n} = a^{(\log_a n)(\log_b a)}$$

=

Proof of the Master theorem

It comes down to the following three cases.

- **The ratio a/b^d is less than 1 ($a/b^d < 1$).**

Then the series is decreasing, and its sum is just given by its first term, $O(n^d)$.

- **The ratio a/b^d is greater than 1 ($a/b^d > 1$).**

The series is increasing and its sum is given by its last term, $O(n^{\log_b a})$:

$$\begin{aligned} n^d \left(\frac{a}{b^d} \right)^{\log_b n} &= n^d \left(\frac{a^{\log_b n}}{(b^{\log_b n})^d} \right) = a^{\log_b n} = a^{(\log_a n)(\log_b a)} \\ &= n^{\log_b a} \end{aligned}$$

Proof of the Master theorem

It comes down to the following three cases.

- **The ratio a/b^d is less than 1 ($a/b^d < 1$).**

Then the series is decreasing, and its sum is just given by its first term, $O(n^d)$.

- **The ratio a/b^d is greater than 1 ($a/b^d > 1$).**

The series is increasing and its sum is given by its last term, $O(n^{\log_b a})$:

$$\begin{aligned} n^d \left(\frac{a}{b^d} \right)^{\log_b n} &= n^d \left(\frac{a^{\log_b n}}{(b^{\log_b n})^d} \right) = a^{\log_b n} = a^{(\log_a n)(\log_b a)} \\ &= n^{\log_b a} \end{aligned}$$

- **The ratio a/b^d is exactly 1 ($a/b^d = 1$).**

In this case all $O(\log n)$ terms of the series are equal to $O(n^d)$.

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

Example 1: $T(n) = 3T\left(\frac{n}{2}\right) + n$

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

Example 1: $T(n) = 3T\left(\frac{n}{2}\right) + n$

- $a = 3$, $b = 2$, $d = 1$, $d < \log_b a$

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

Example 1: $T(n) = 3T\left(\frac{n}{2}\right) + n$

- $a = 3$, $b = 2$, $d = 1$, $d < \log_b a$
- $T(n) = O(n^{\log_b a}) = O(n^{\log 3})$

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

Example 2: $T(n) = 3T\left(\frac{n}{4}\right) + n^5$

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

Example 2: $T(n) = 3T\left(\frac{n}{4}\right) + n^5$

- $a = 3$, $b = 4$, $d = 5$, $d > \log_b a$

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

Example 2: $T(n) = 3T\left(\frac{n}{4}\right) + n^5$

- $a = 3$, $b = 4$, $d = 5$, $d > \log_b a$
- $T(n) = O(n^d) = O(n^5)$

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

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

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

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

- $a = 4$, $b = 2$, $d = 2$, $d = \log_b a$

Master theorem: Example

If $T(n) = aT\left(\lceil \frac{n}{b} \rceil\right) + O(n^d)$ for some constant $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

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

- $a = 4$, $b = 2$, $d = 2$, $d = \log_b a$
- $T(n) = O(n^d \log n) = O(n^2 \log n)$

dank u
ju faleminderit
Tack
Asante 谢谢 Tak mulțumesc
kiitos
Salamat! Gracias
Terima kasih Aliquam
Merci
Dankie Obrigado
ありがとう köszönöm grazie
Aliquam Go raibh maith agat
děkuii Thank you