Experience certainty.

Series and Recurrences

Shirish Karande

10 June 2014

Topics

- Arithmetic Progression
- Geometric Progression
- Harmonic Series and Approximation
- Recurrences
- Master Theorem
  - Statement
  - Examples
  - Proof

- **Example:** A coffee shop in a city usually has 200 visitors between 10:00 a.m. and 6:00 p.m. After a while, its customers kept increasing by 5 every day for 10 days in a row.

200, 205, 210, 215, 220, 225, 230, 235, 240, 245

Arithmetic Progression

- **An Arithmetic Progression (AP)** is a sequence of numbers such that the difference between the consecutive terms is constant.

$$a_n = a_1 + (n-1)d,$$

- **Example:** A coffee shop in a city usually has 200 visitors between 10:00 a.m. and 6:00 p.m. After a while, its customers kept increasing by 5 every day for 10 days in a row.

200, 205, 210, 215, 220, 225, 230, 235, 240, 245

Arithmetic Progression

- **An Arithmetic Progression (AP)** is a sequence of numbers such that the difference between the consecutive terms is constant.

$$a_n = a_1 + (n-1)d,$$

- **Example:** A coffee shop in a city usually has 200 visitors between 10:00 a.m. and 6:00 p.m. After a while, its customers kept increasing by 5 every day for 10 days in a row.

200, 205, 210, 215, 220, 225, 230, 235, 240, 245

Assuming each customer bought 1 cup coffee, what were total coffee cups sold in the 10 days ?

Arithmetic Progression

- **An Arithmetic Progression (AP)** is a sequence of numbers such that the difference between the consecutive terms is constant.

$$a_n = a_1 + (n-1)d,$$

- **Example:** A coffee shop in a city usually has 200 visitors between 10:00 a.m. and 6:00 p.m. After a while, its customers kept increasing by 5 every day for 10 days in a row.

200, 205, 210, 215, 220, 225, 230, 235, 240, 245

Assuming each customer bought 1 cup coffee, what were total coffee cups sold in the 10 days ?

Answer: (200+245) + (205+240) ..... = 5*445 = 2225 cups

Arithmetic Progression

- **An Arithmetic Progression (AP)** is a sequence of numbers such that the difference between the consecutive terms is constant.

$$a_n = a_1 + (n-1)d,$$

- **Example:** A coffee shop in a city usually has 200 visitors between 10:00 a.m. and 6:00 p.m. After a while, its customers kept increasing by 5 every day for 10 days in a row.

  200, 205, 210, 215, 220, 225, 230, 235, 240, 245

  Assuming each customer bought 1 cup coffee, what were total coffee cups sold in the 10 days ?

  <u>Answer:</u> (200+245) + (205+240) ….. = 5*445 = 2225 cups

- **Sum of AP:** $$S_n = \frac{n}{2}[2a_1 + (n-1)d].$$

- **Example:** Suppose you watch a movie on its first day of releases, and recommend it to five friends, who watch it on the second day, and recommend it 5 friends each, ….. for 7 days

1, 5, 25, 125, 625, 3125, 15625

Geometric Progression

- **A Geometric Progression (GP),** also is a sequence of numbers where each term after the first is found by multiplying the previous one by a fixed, non-zero number called the *common ratio*.

$$a_n = a\,r^{n-1}.$$

- **Example:** Suppose you watch a movie on its first day of releases, and recommend it to five friends, who watch it on the second day, and recommend it 5 friends each, ….. for 7 days

1, 5, 25, 125, 625, 3125, 15625

Geometric Progression

- **A Geometric Progression (GP),** also is a sequence of numbers where each term after the first is found by multiplying the previous one by a fixed, non-zero number called the *common ratio*.

$$a_n = a\,r^{n-1}.$$

- **Example:** Suppose you watch a movie on its first day of releases, and recommend it to five friends, who watch it on the second day, and recommend it 5 friends each, ….. for 7 days

1, 5, 25, 125, 625, 3125, 15625

How many watched the movie because of you ?

Geometric Progression

- **A Geometric Progression (GP),** also is a sequence of numbers where each term after the first is found by multiplying the previous one by a fixed, non-zero number called the *common ratio*.

$$a_n = a\,r^{n-1}.$$

- **Example:** Suppose you watch a movie on its first day of releases, and recommend it to five friends, who watch it on the second day, and recommend it 5 friends each, ….. for 7 days

1, 5, 25, 125, 625, 3125, 15625

How many watched the movie because of you ?

- **Sum of GP:**

$$\sum_{k=1}^{n} ar^{k-1} = \frac{a(1-r^n)}{1-r}.$$

Answer: $(1-5^7)/(1-5) = 78124/4 =$ **19531** $(r = 5, a = 1, n = 7)$

- Example: On day 1 a beggar finds a house that donates 1Re a day. On day 2 another beggar joins him and the Re is split. Third day another beggar joins so on…… so how does the first beggars share progress

  1, 1/2, 1/3, 1/4, 1/5, 1/6………………………

Harmonic Series

- Example: On day 1 a beggar finds a house that donates 1Re a day. On day 2 another beggar joins him and the Re is split. Third day another beggar joins so on…… so how does the first beggars share progress

  1, 1/2, 1/3, 1/4, 1/5, 1/6……………………..

  How much will the beggar have made in *n* days? In lifetime ?

- Harmonic Number: $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}.$

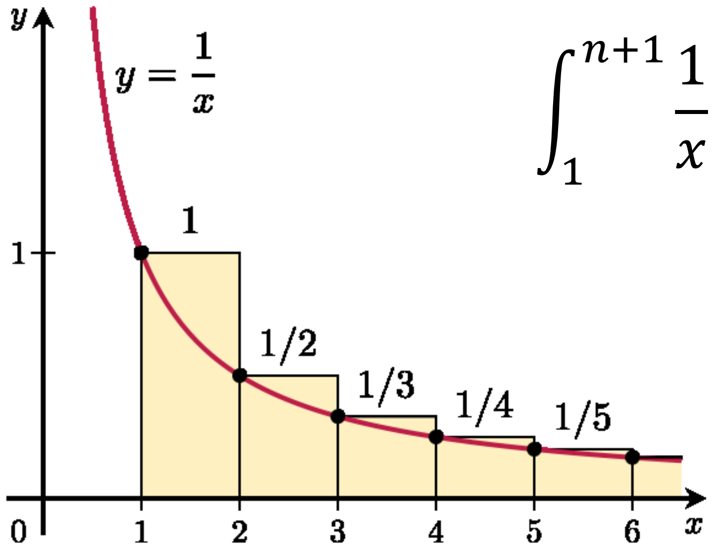
Harmonic Series

- Example: On day 1 a beggar finds a house that donates 1Re a day. On day 2 another beggar joins him and the Re is split. Third day another beggar joins so on…… so how does the first beggars share progress

  1, 1/2, 1/3, 1/4, 1/5, 1/6……………………..

  How much will the beggar have made in *n* days? In lifetime ?

- Harmonic Number: $$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}.$$



$$\int_1^{n+1} \frac{1}{x}\,dx = \ln(n+1) < H_n < \ln(n+1) + 1$$

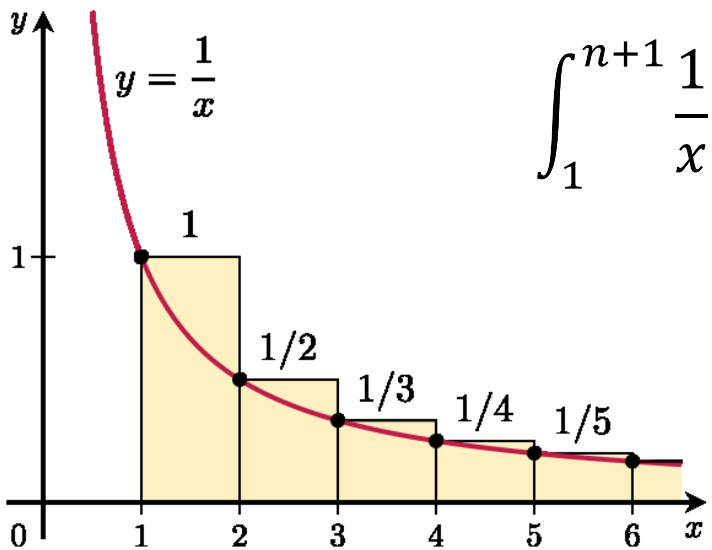
Harmonic Series

- Example: On day 1 a beggar finds a house that donates 1Re a day. On day 2 another beggar joins him and the Re is split. Third day another beggar joins so on…… so how does the first beggars share progress

  1, 1/2, 1/3, 1/4, 1/5, 1/6………………………

  How much will the beggar have made in *n* days? In lifetime ?

- Harmonic Number: $$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}.$$



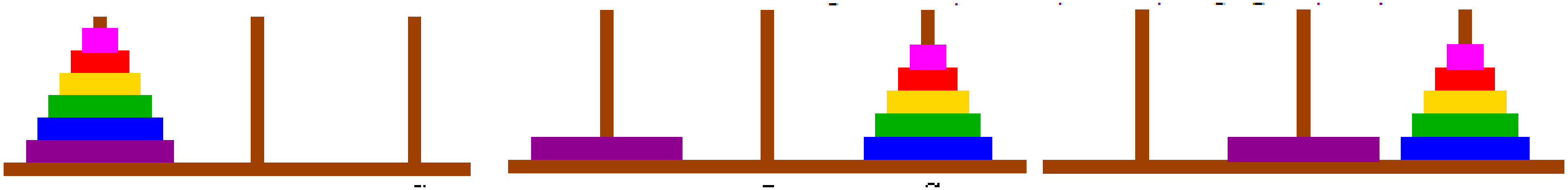
$$\int_1^{n+1} \frac{1}{x}\,dx = \ln(n+1) < H_n < \ln(n+1) + 1$$

So the beggar can make potentially make an unbounded amount of money but the net income grows rather slowly.

1yr ~ 5.9 Re
10yrs ~ 8.2Re
100yrs ~ 10.5Re

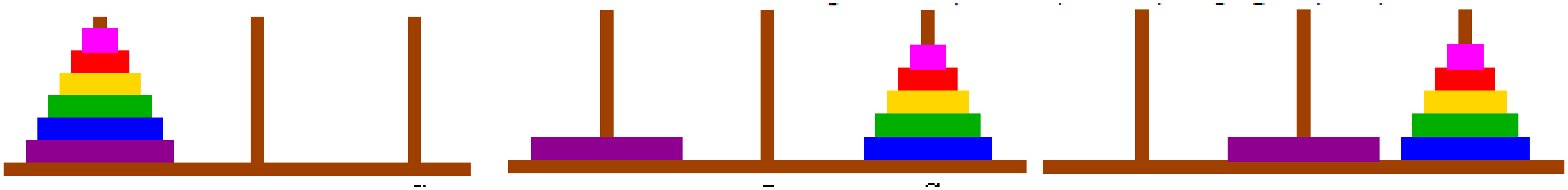
Example: Analysis Of Recursive Algorithms

- Example: Tower of Hanoi



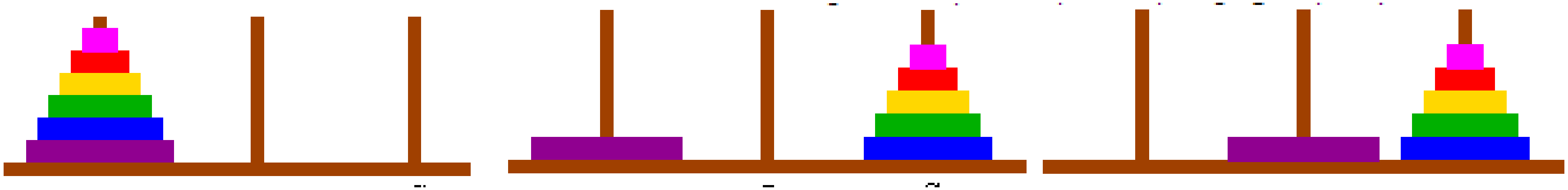
Example: Analysis Of Recursive Algorithms

- Example: Tower of Hanoi

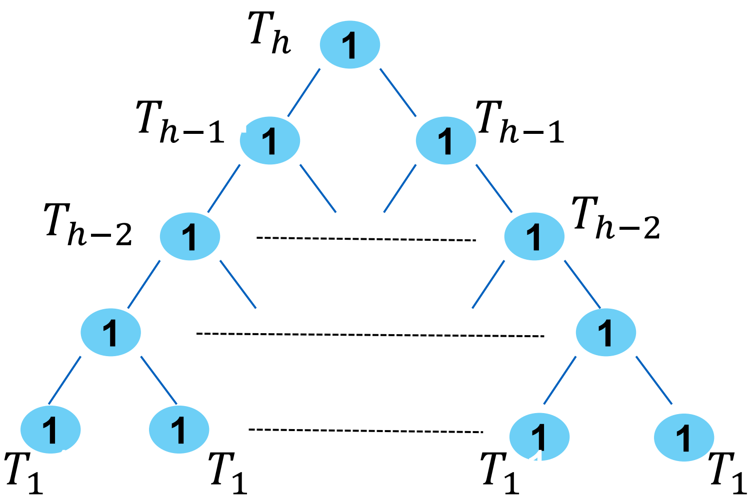


- Recursion: $$T_h = 2T_{h-1} + 1$$

- Example: Tower of Hanoi

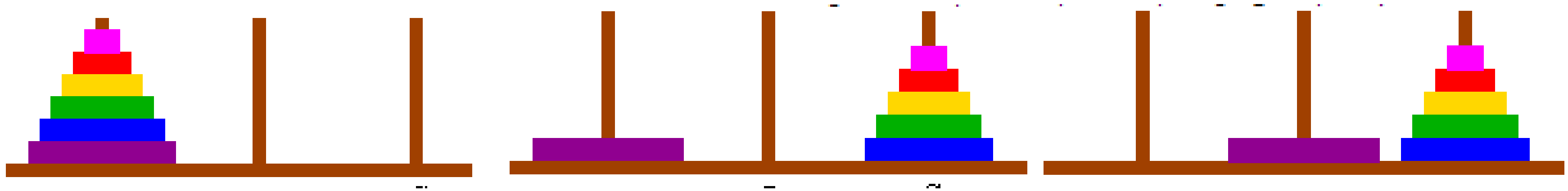


- Recursion: $$T_h = 2T_{h-1} + 1$$

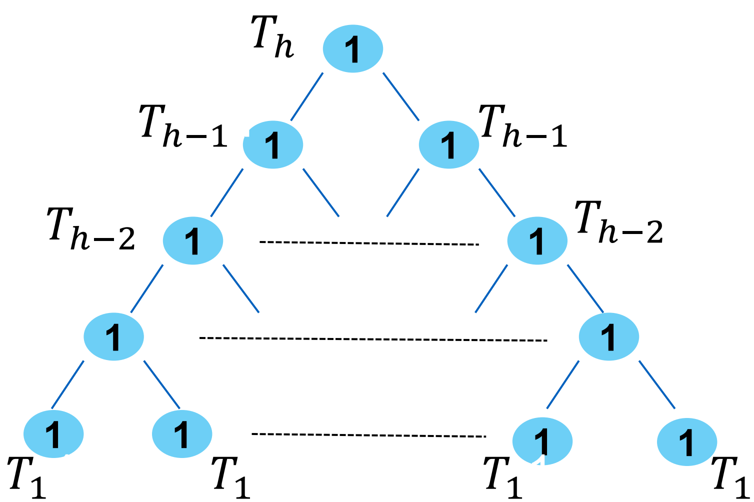


Example: Analysis Of Recursive Algorithms

- Example: Tower of Hanoi



- Recursion: $$T_h = 2T_{h-1} + 1$$



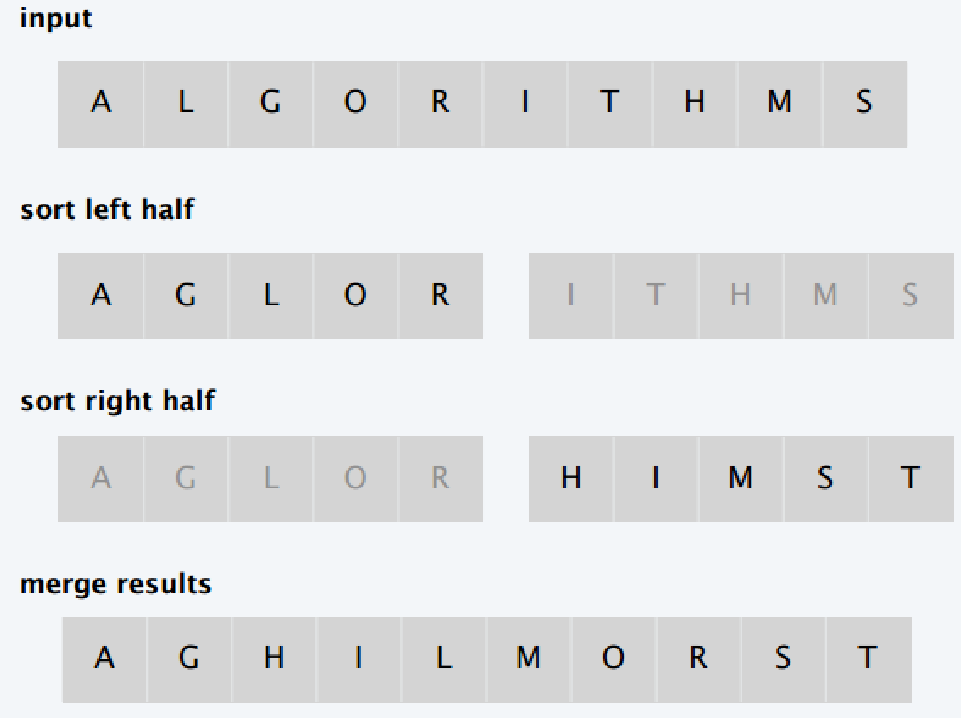
Counting the number of leaves gives us the solution. The number of leaves at each level form a **GP**

$$\mathbf{1 + 2 + 4 .. + 2^{k-1} + \cdots 2^{h-1}}$$

$$T_h = 2^h - 1$$

Merge Sort

- Sort the list of *n* elements recursively
  - Split list into two lists of size n/2
  - Sort the two lists
  - Merge the sorted lists



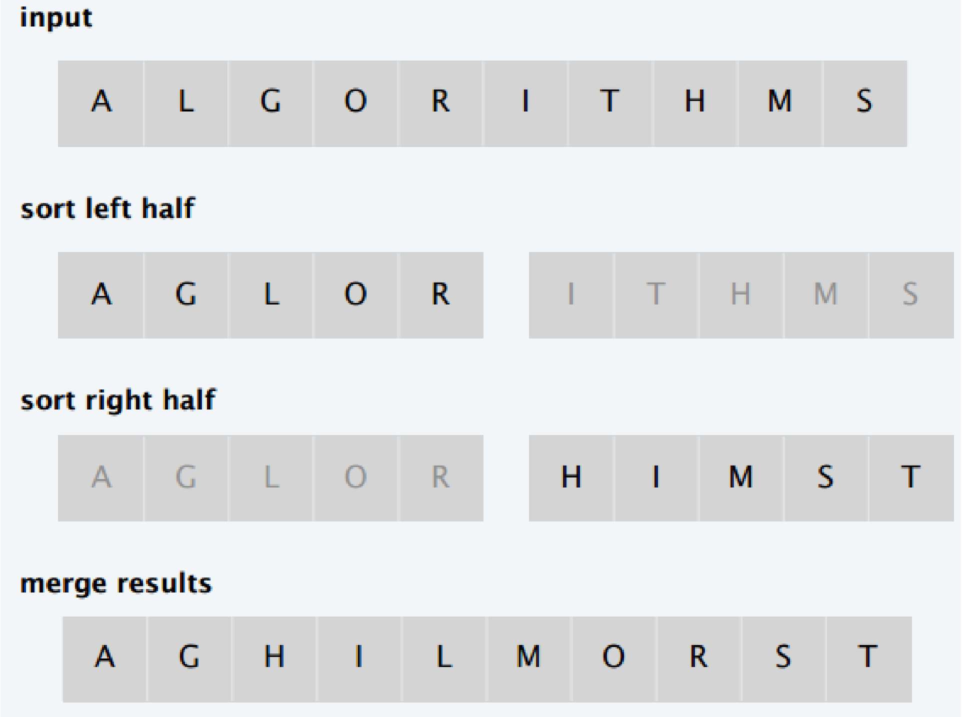
Merge Sort

- Sort the list of *n* elements recursively
  - Split list into two lists of size n/2
  - Sort the two lists
  - Merge the sorted lists

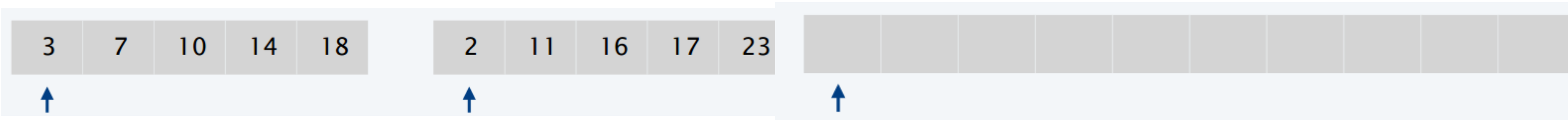
$T(n)$= *Complexity of Sorting a list of size n*

$M\left(\frac{n}{2}\right)$= *Complexity of Merging sorted lists of size n/2*

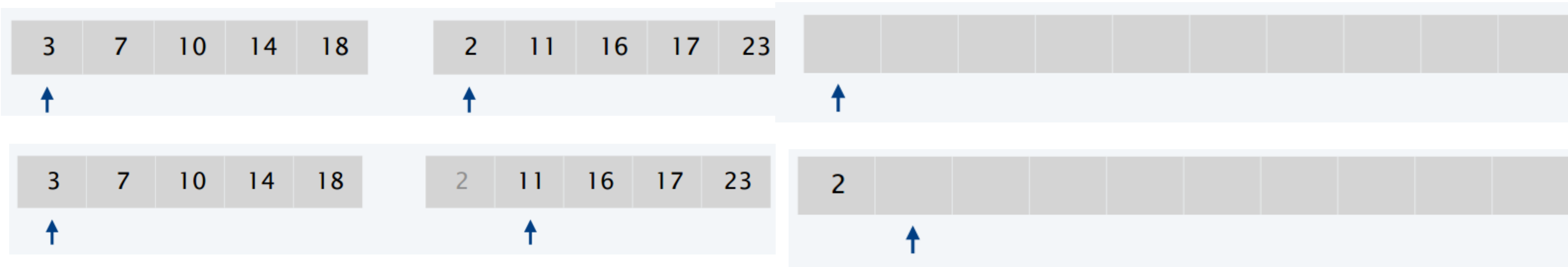
$$\boldsymbol{T(n) = 2T\left(\frac{n}{2}\right) + M\left(\frac{n}{2}\right)}$$



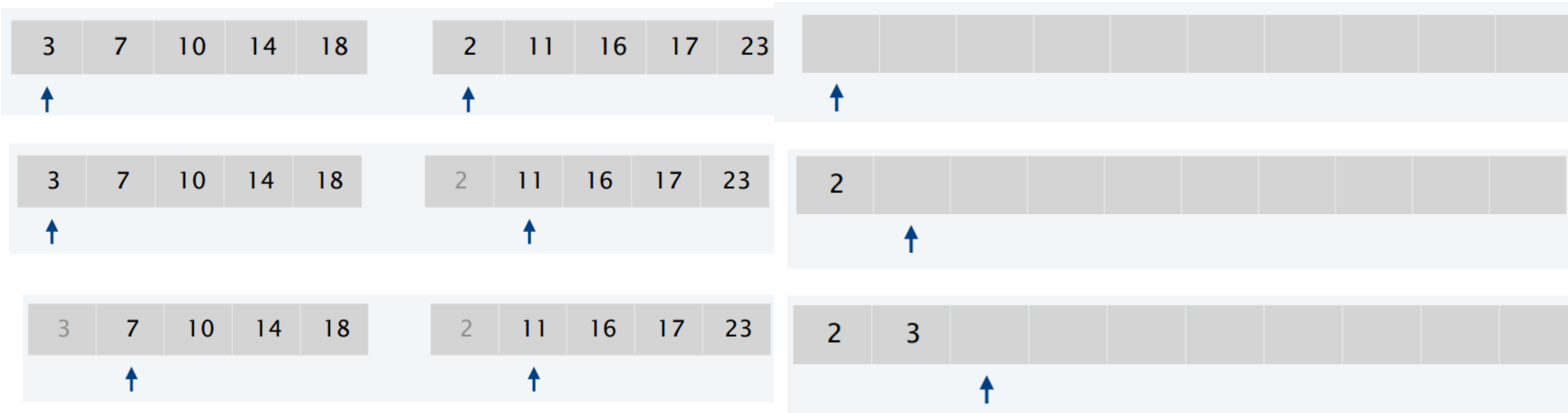
Complexity Of Merging Sorted Lists



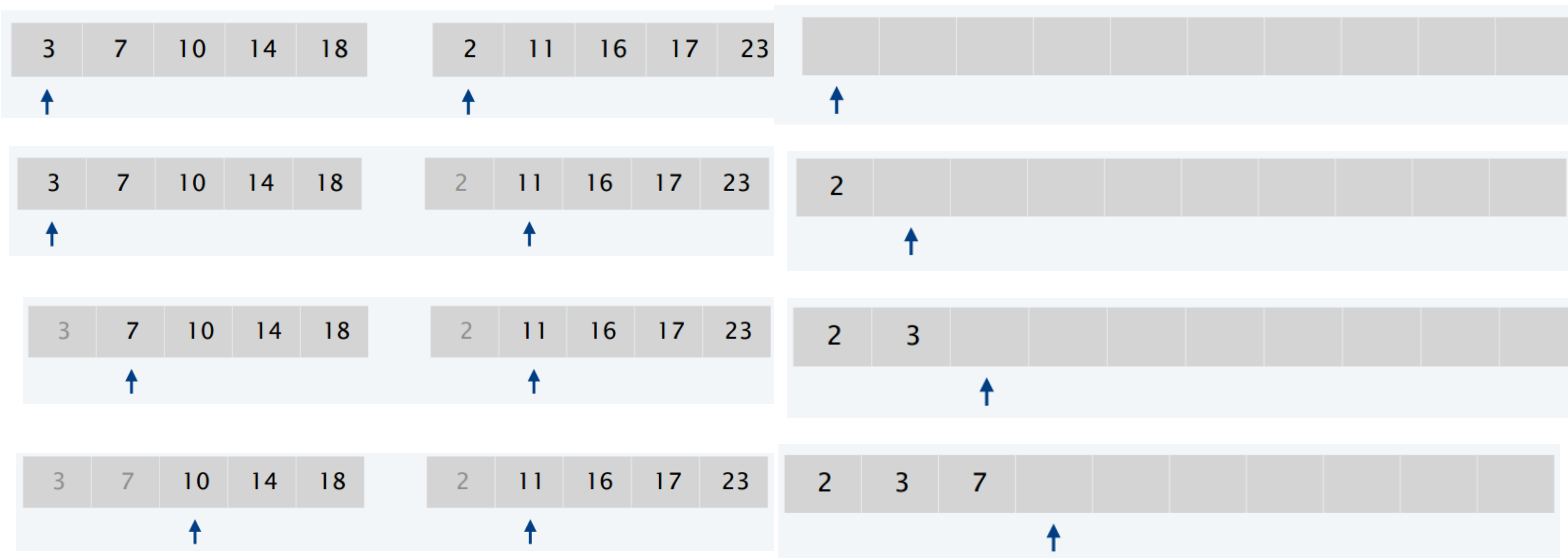
Complexity Of Merging Sorted Lists



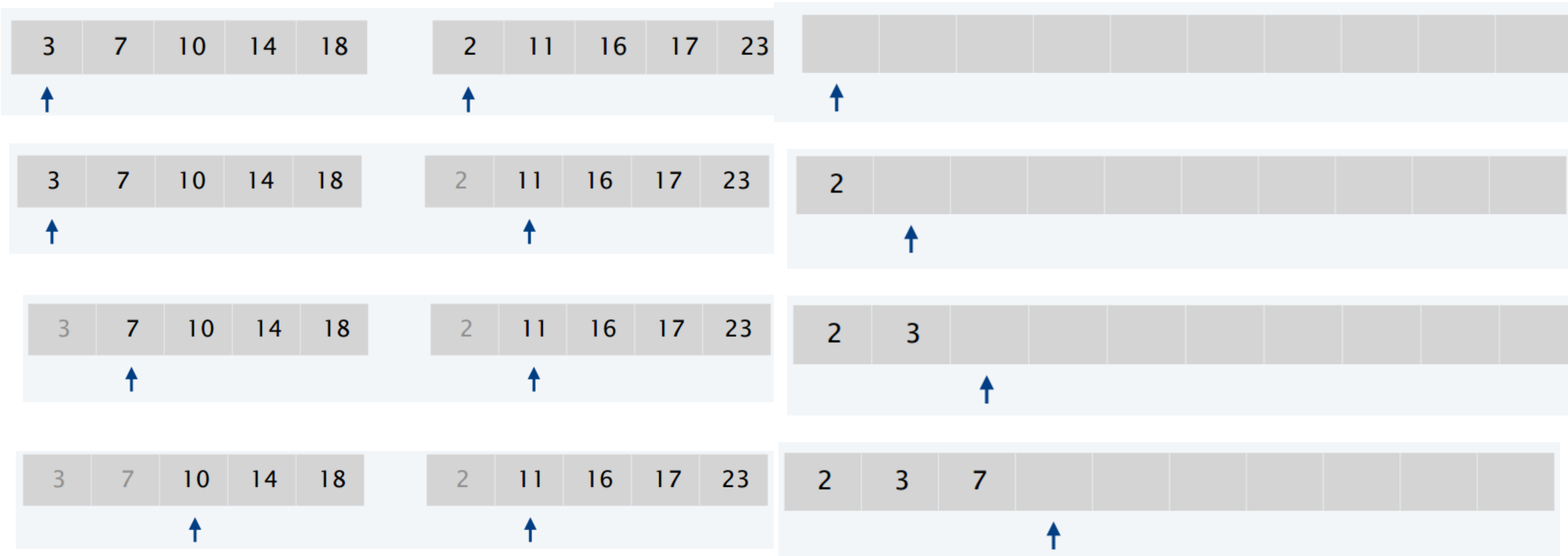
Complexity Of Merging Sorted Lists



Complexity Of Merging Sorted Lists

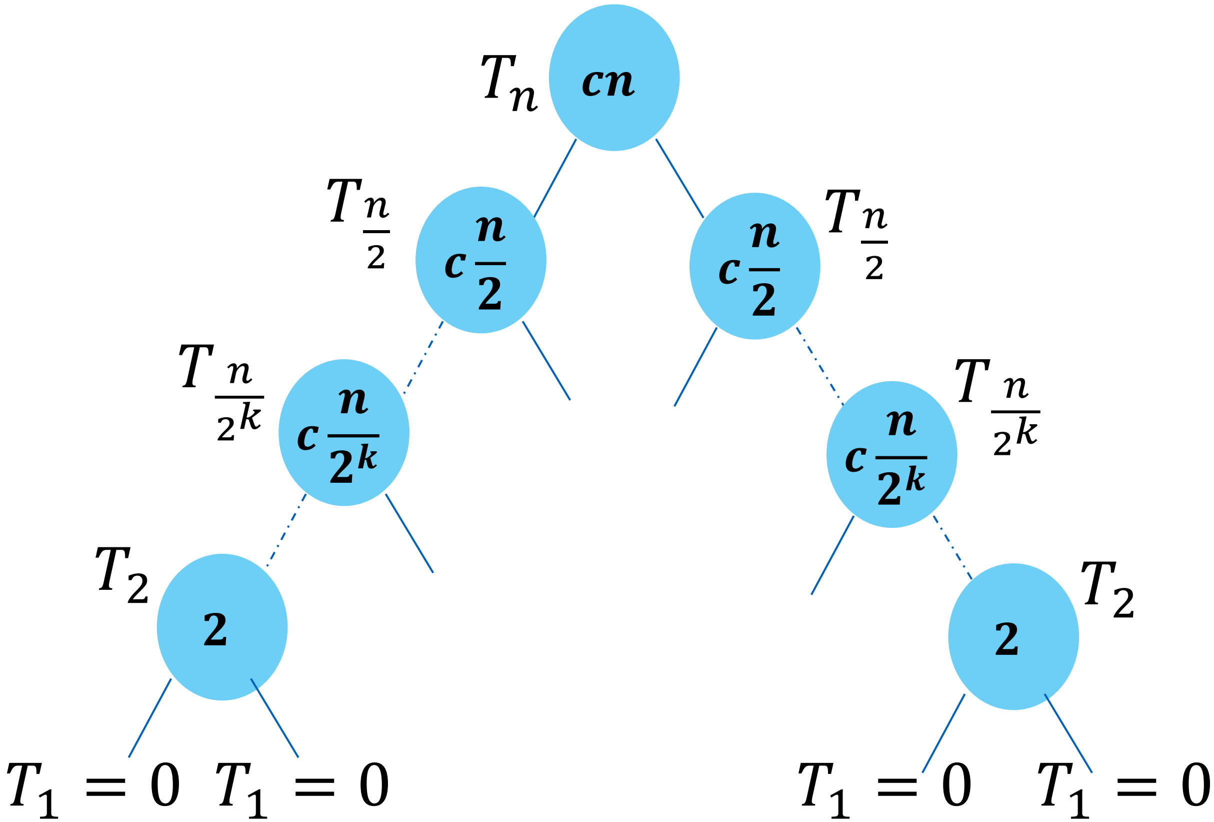


Complexity Of Merging Sorted Lists



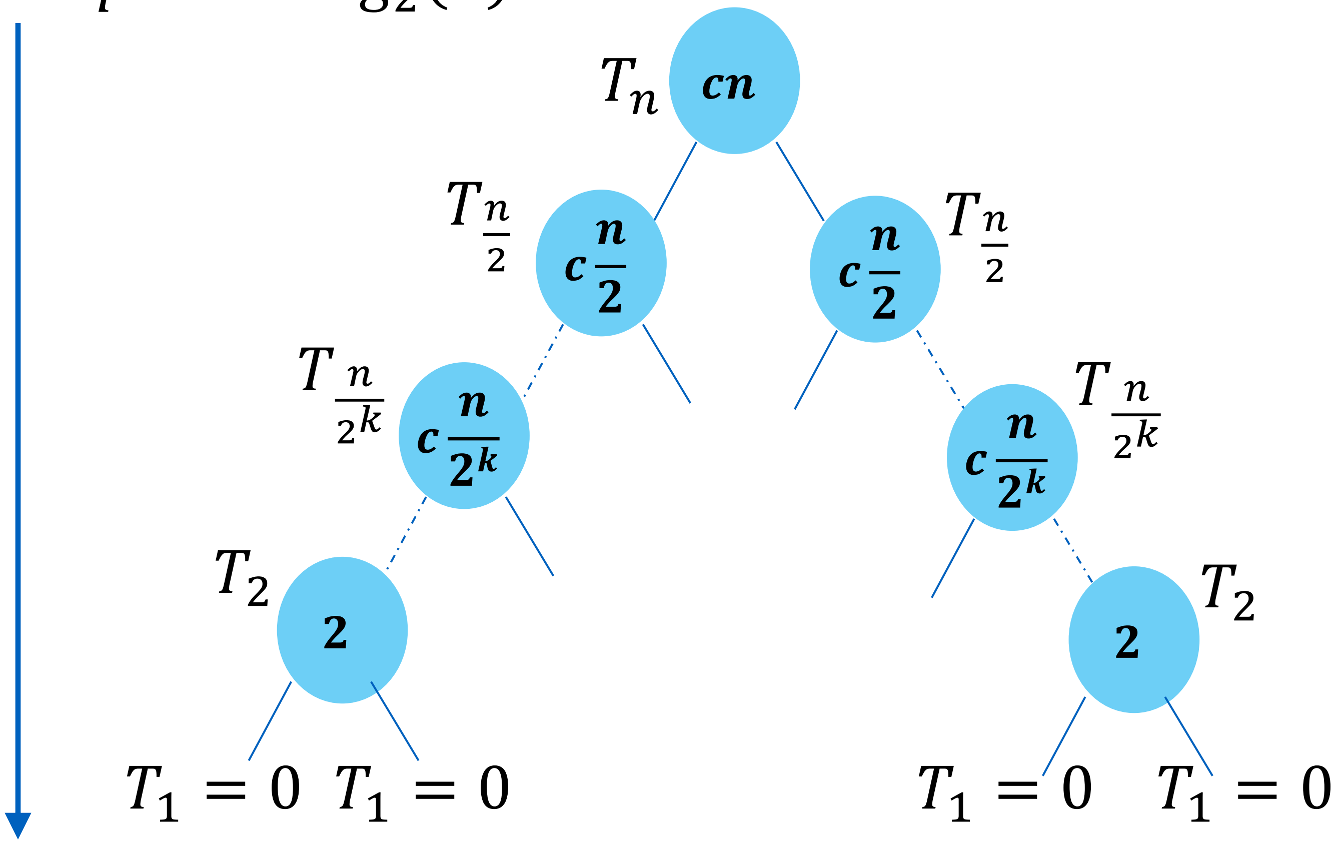
- One of the list pointer moves ahead in each step
- Maximum *n* comparisons to merge two lists of size *n*/2
- **Complexity of merging two list is $\theta(n)$, i.e. $M(n/2) \sim cn$**

Analysis of Merge Sort Recursion



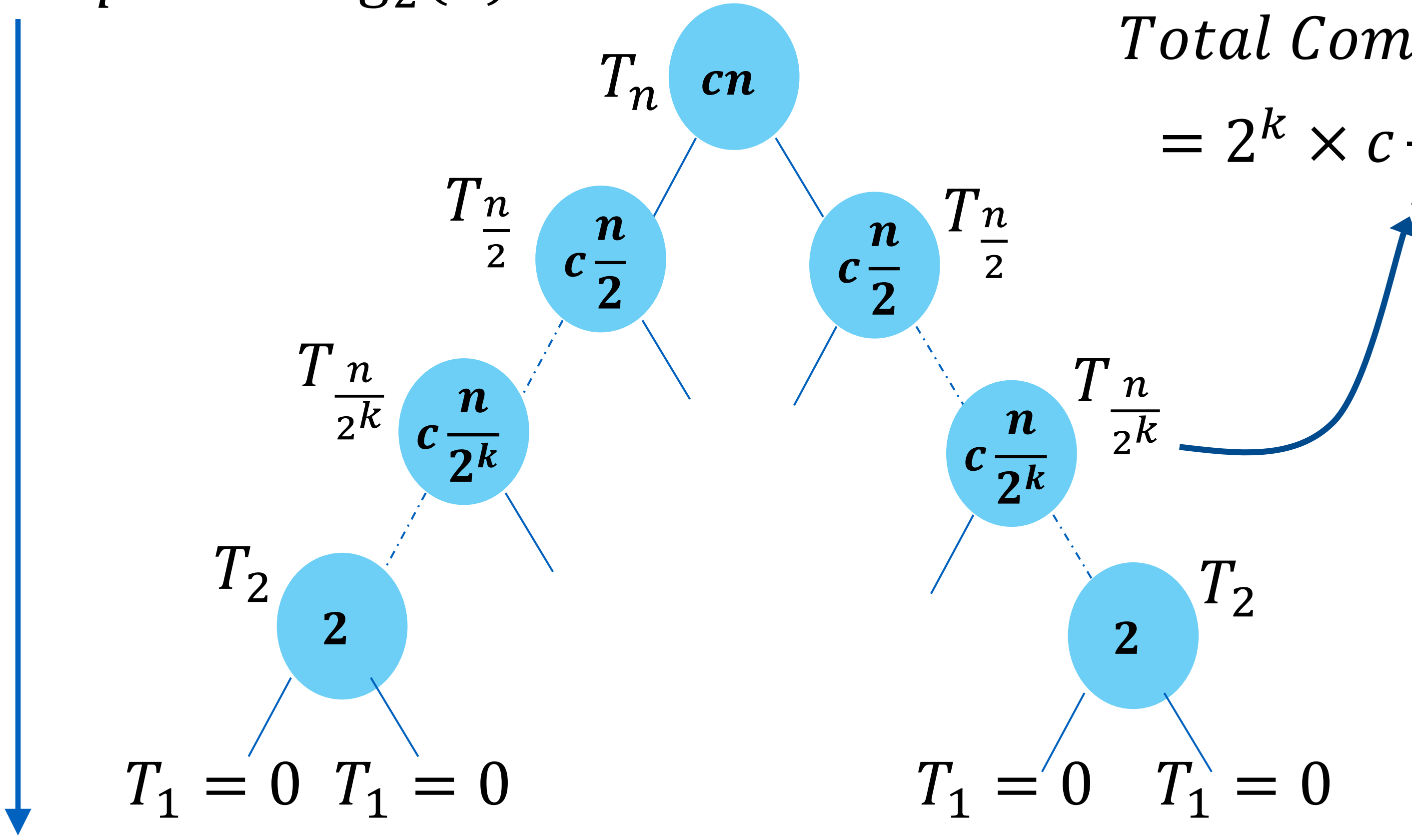
Analysis of Merge Sort Recursion

$$Tree\ Depth = \log_2(n)$$



Analysis of Merge Sort Recursion

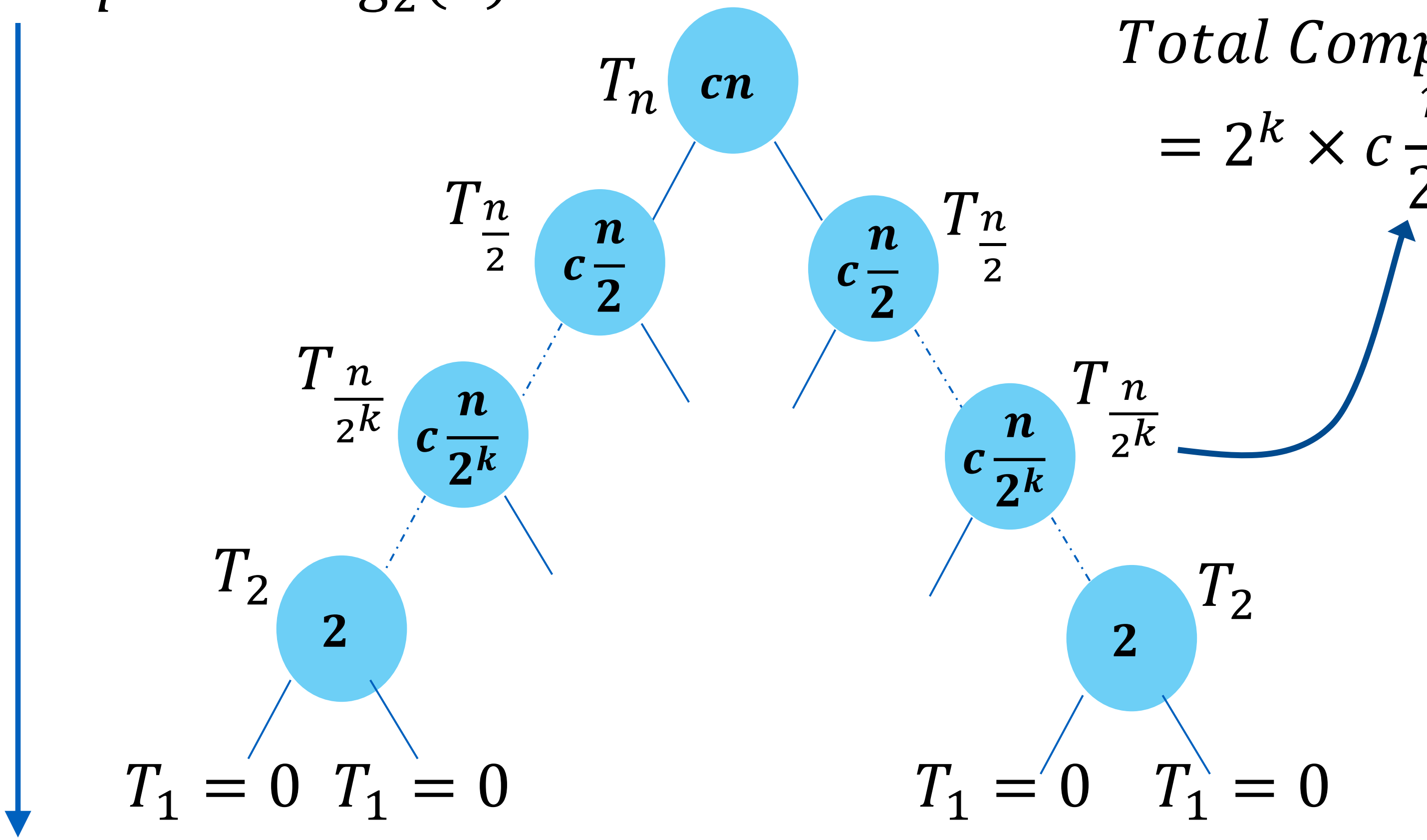
$$Tree\ Depth = \log_2(n)$$



$$Total\ Computation = 2^k \times c\frac{n}{2^k} = cn$$

Analysis of Merge Sort Recursion

$$Tree\ Depth = \log_2(n)$$



$$Total\ Computation = 2^k \times c\frac{n}{2^k} = cn$$

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

General Analysis Of Recursive Algorithms

```
procedure T( n : size of problem ) defined as:
  if n < 1 then exit

  Do work of amount f(n)

  T(n/b)
  T(n/b)
  ...repeat for a total of a times...
  T(n/b)
end procedure
```

General Analysis Of Recursive Algorithms

```
procedure T( n : size of problem ) defined as:
  if n < 1 then exit

  Do work of amount f(n)

  T(n/b)
  T(n/b)
  ...repeat for a total of a times...
  T(n/b)
end procedure
```

Recursion :

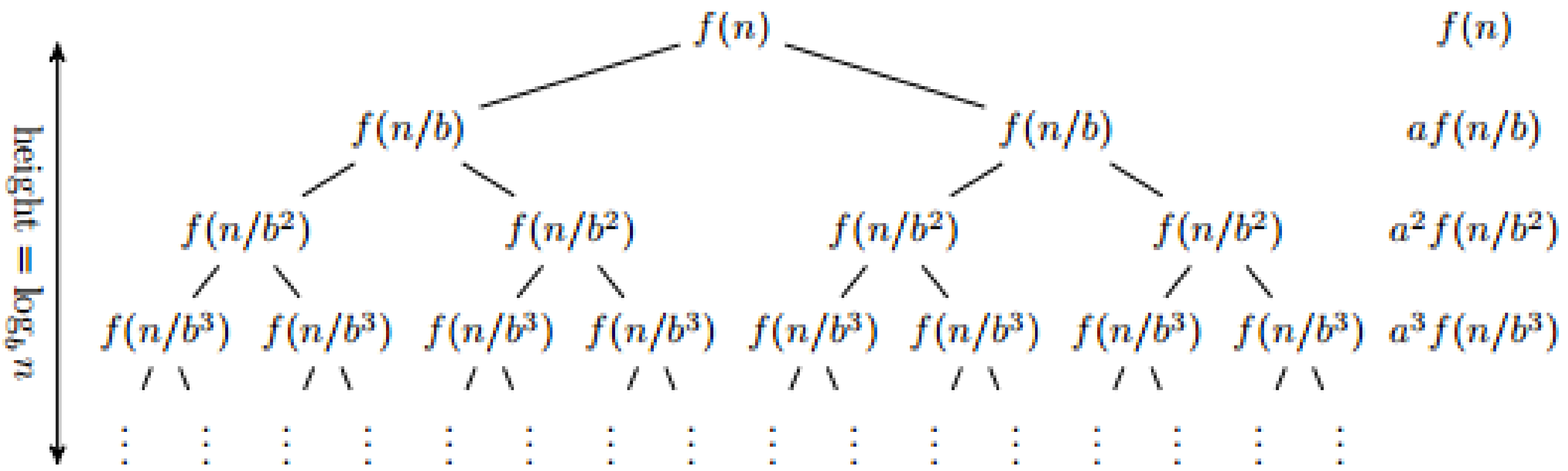
$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \quad \text{where } a \geq 1,\, b > 1$$

General Analysis Of Recursive Algorithms

```
procedure T( n : size of problem ) defined as:
  if n < 1 then exit

  Do work of amount f(n)

  T(n/b)
  T(n/b)
  ...repeat for a total of a times...
  T(n/b)
end procedure
```

Recursion :

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \quad \text{where } a \geq 1,\ b > 1$$



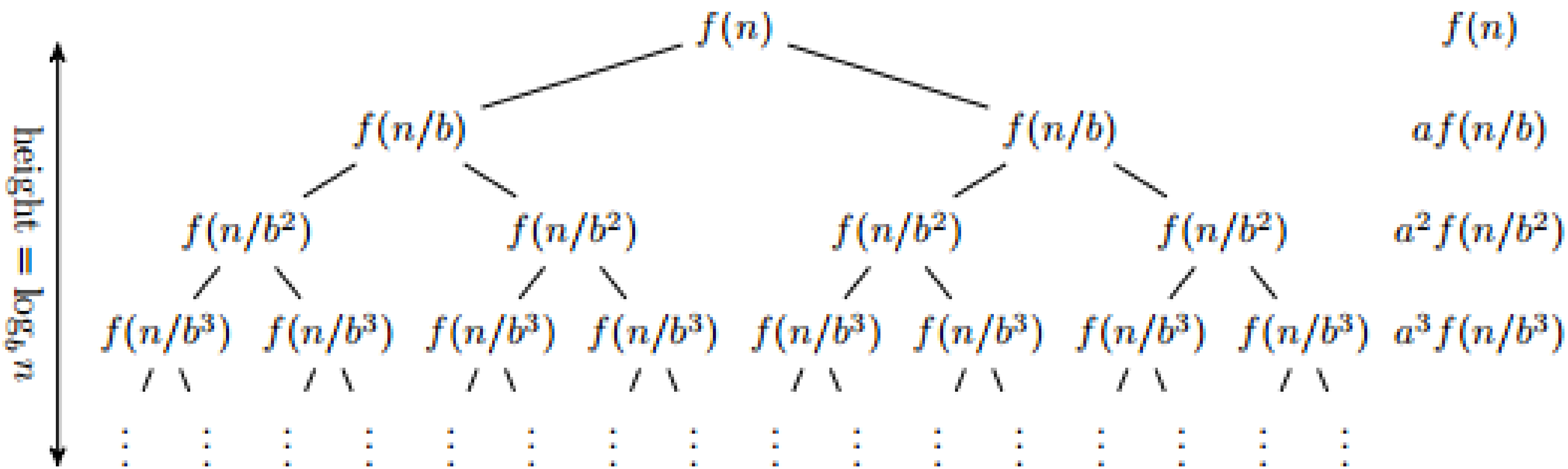
General Analysis Of Recursive Algorithms

```
procedure T( n : size of problem ) defined as:
  if n < 1 then exit

  Do work of amount f(n)

  T(n/b)
  T(n/b)
  ...repeat for a total of a times...
  T(n/b)
end procedure
```

Recursion :

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \quad \text{where } a \geq 1,\ b > 1$$

$$\boxed{T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a})}$$



Statement of Master Theorem

Theorem (Master Method) Consider the recurrence

$$T(n) = aT(n/b) + f(n),$$

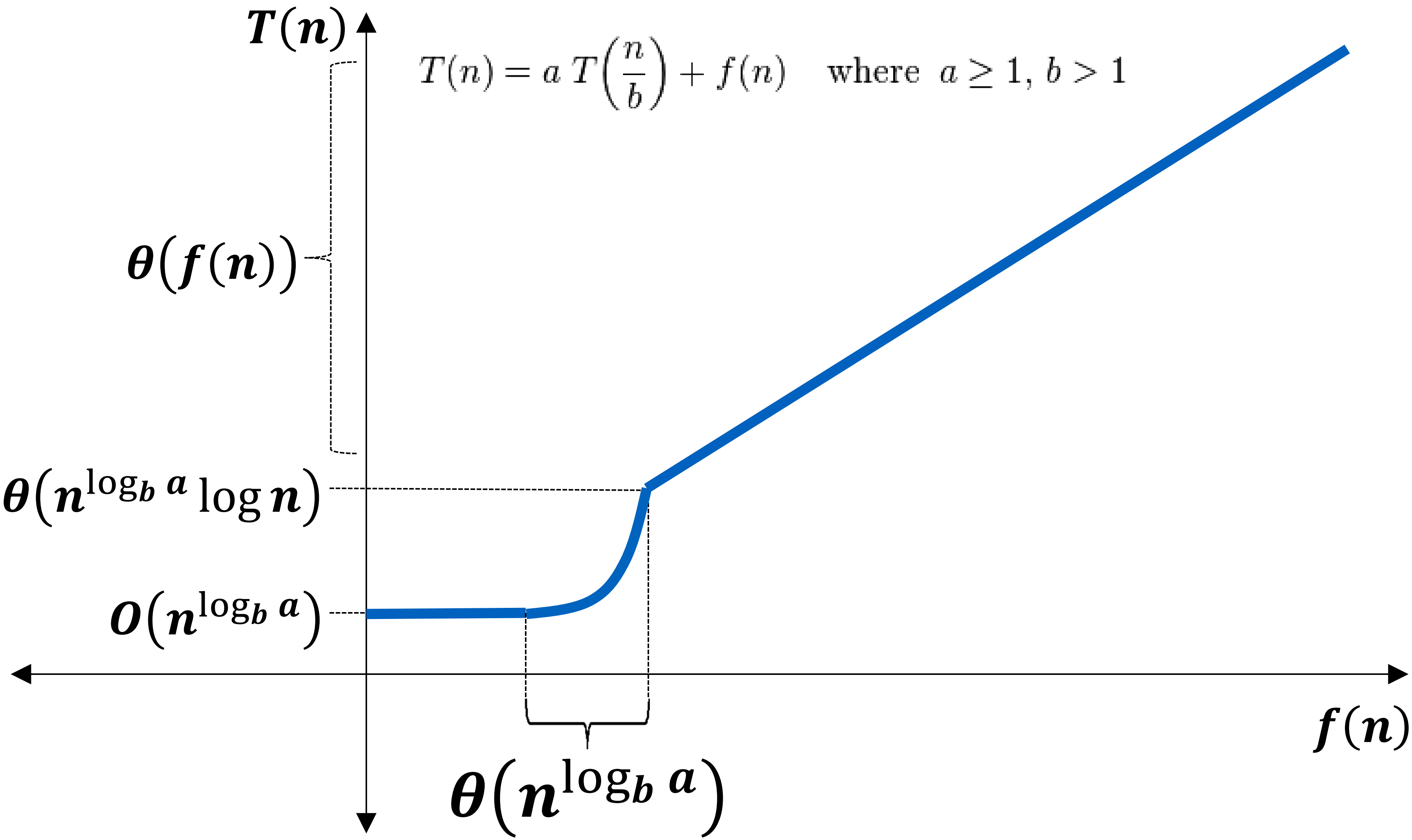
where a, b are constants. Then

(A) If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = O(n^{\log_b a})$.

(B) If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.

(C) If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if f satisfies the smoothness condition $af(n/b) \leq cf(n)$ for some constant $c < 1$, then $T(n) = \Theta(f(n))$.

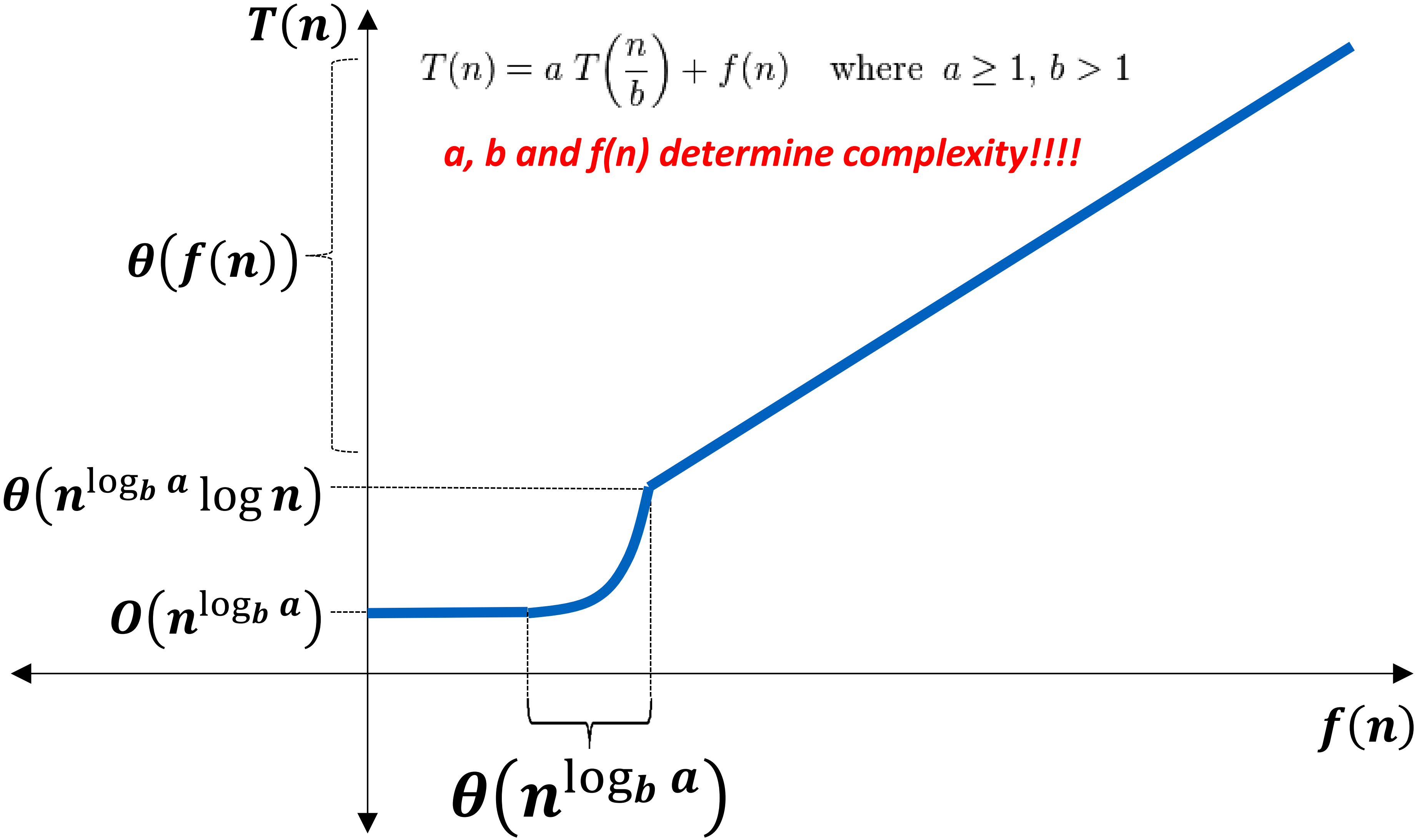
A Way To Remember The Statement



A Way To Remember The Statement

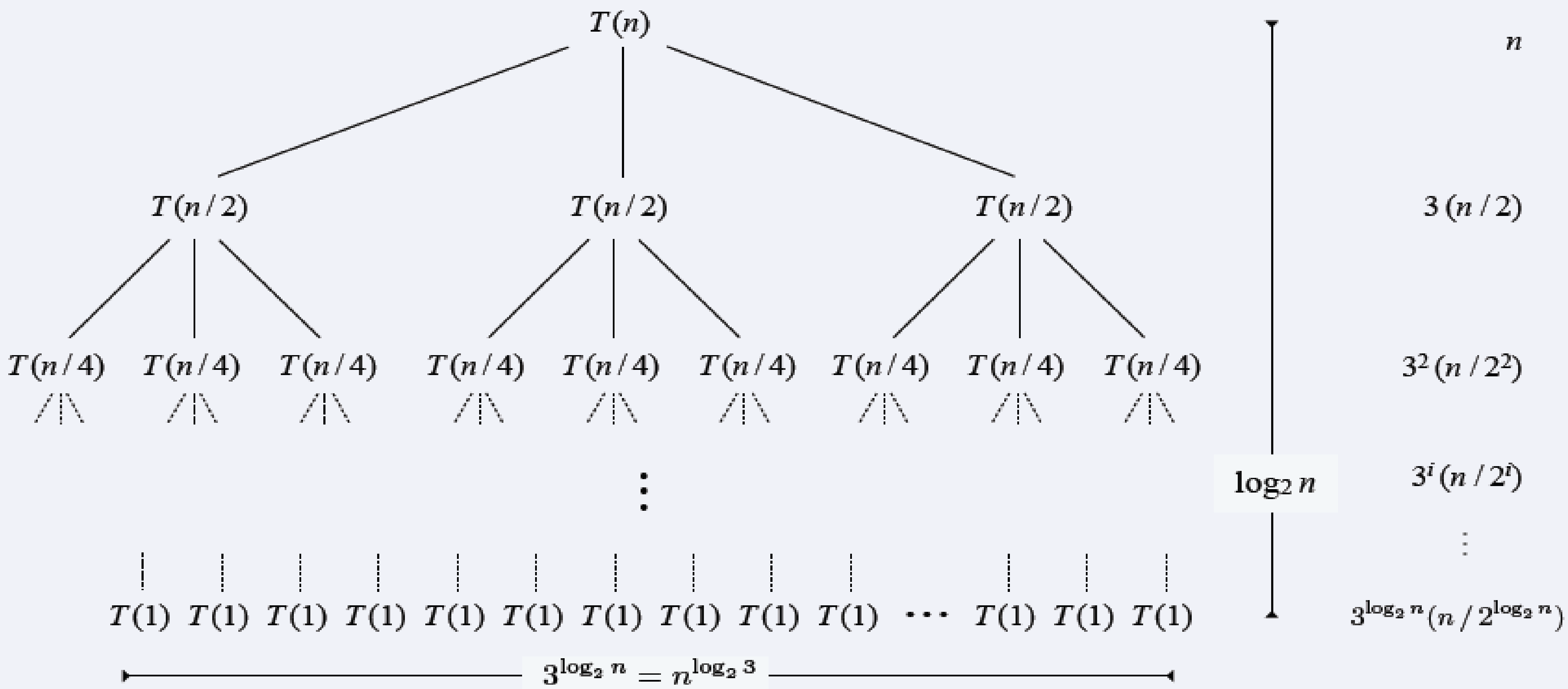
$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \quad \text{where } a \geq 1,\ b > 1$$

a, b and f(n) determine complexity!!!!



Case A: Cost dominated by cost of leaves

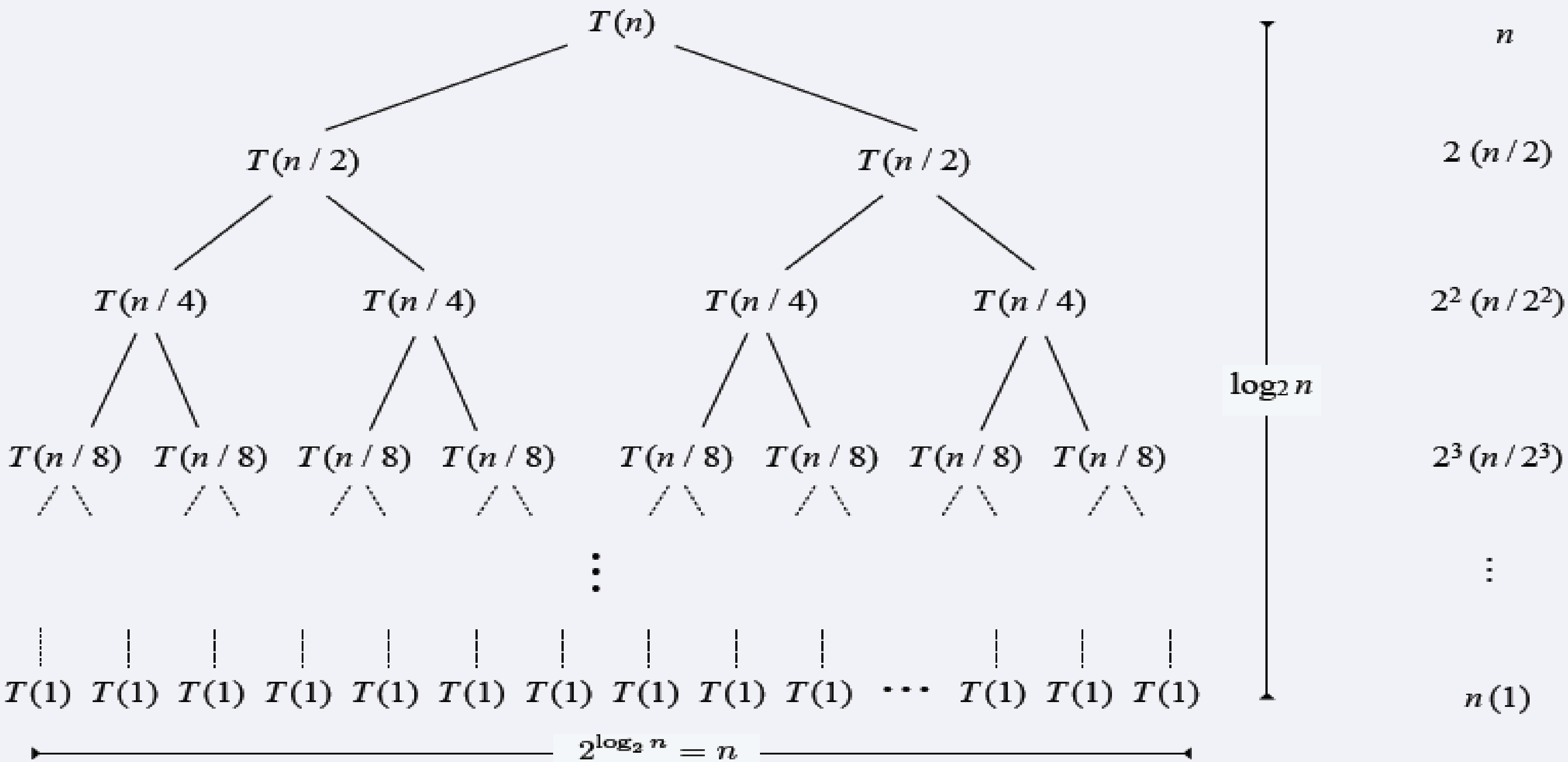
Ex 1. If $T(n)$ satisfies $T(n) = 3\,T(n/2) + n$, with $T(1) = 1$, then $T(n) = \Theta(n^{\lg 3})$.



$$r = 3/2 > 1 \qquad T(n) = (1 + r + r^2 + r^3 + \ldots + r^{\log_2 n})\, n = \frac{r^{1+\log_2 n} - 1}{r - 1}\, n = 3n^{\log_2 3} - 2n$$

Case B: Cost evenly distributed among levels

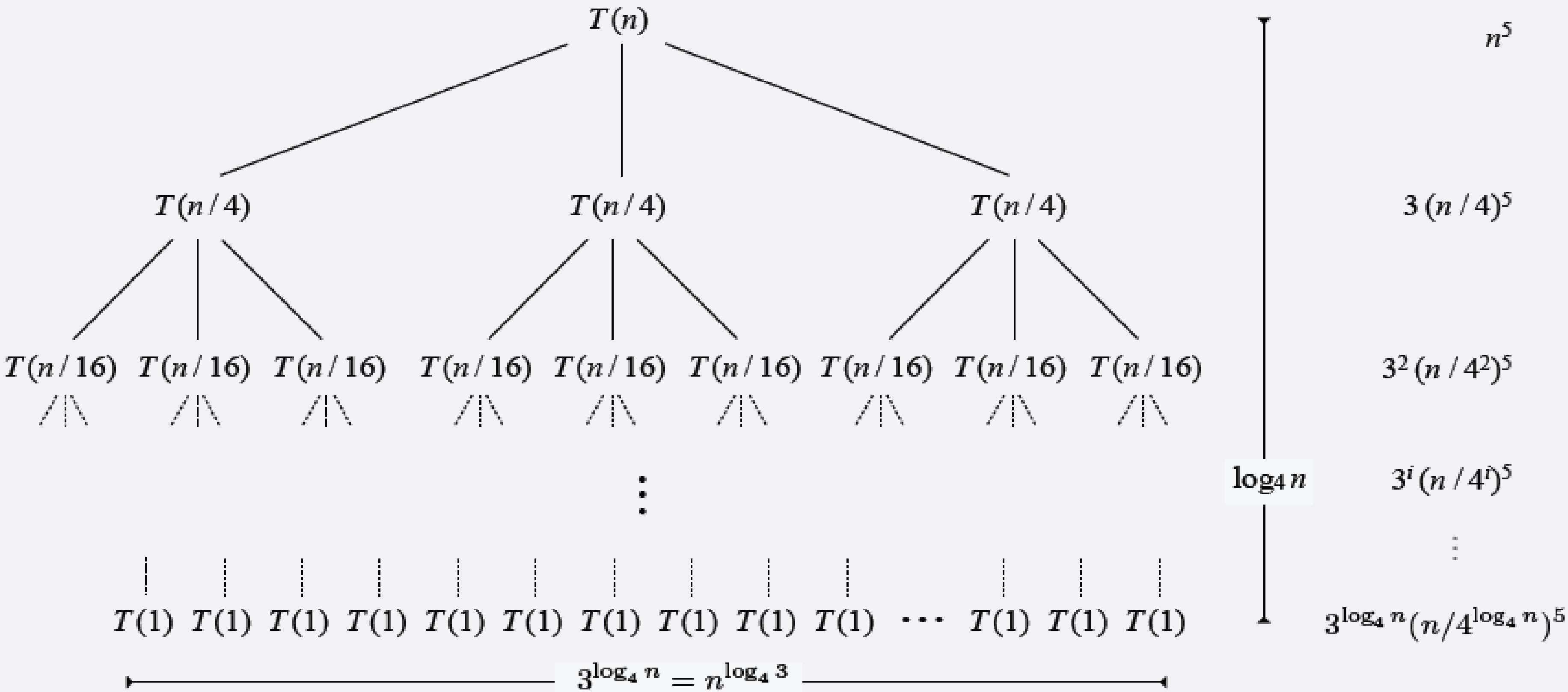
Ex 2. If $T(n)$ satisfies $T(n) = 2\,T(n/2) + n$, with $T(1) = 1$, then $T(n) = \Theta(n \log n)$.



$r = 1$ $$T(n) = (1 + r + r^2 + r^3 + \ldots + r^{\log_2 n})\, n \;=\; n\,(\log_2 n + 1)$$

Case C: Cost dominated by cost of roots

Ex 3. If $T(n)$ satisfies $T(n) = 3\,T(n/4) + n^5$, with $T(1) = 1$, then $T(n) = \Theta(n^5)$.



$r = 3/4^5 < 1 \qquad n^5 \le T(n) \le (1 + r + r^2 + r^3 + \ldots)\, n^5 \le \dfrac{1}{1-r}\, n^5$

Some Well Known Examples

Algorithm	Recurrence Relationship	Run time
Binary search	$T(n) = T\left(\frac{n}{2}\right) + O(1)$	
Binary tree traversal	$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$	
Optimal Sorted Matrix Search	$T(n) = 2T\left(\frac{n}{2}\right) + O(\log n)$	
Merge Sort	$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	

Some Well Known Examples

Algorithm	Recurrence Relationship	Run time
Binary search	$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$
Binary tree traversal	$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$	
Optimal Sorted Matrix Search	$T(n) = 2T\left(\frac{n}{2}\right) + O(\log n)$	
Merge Sort	$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	

Some Well Known Examples

Algorithm	Recurrence Relationship	Run time
Binary search	$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$
Binary tree traversal	$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$	$O(n)$
Optimal Sorted Matrix Search	$T(n) = 2T\left(\frac{n}{2}\right) + O(\log n)$	
Merge Sort	$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	

Some Well Known Examples

Algorithm	Recurrence Relationship	Run time
Binary search	$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$
Binary tree traversal	$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$	$O(n)$
Optimal Sorted Matrix Search	$T(n) = 2T\left(\frac{n}{2}\right) + O(\log n)$	$O(n)$
Merge Sort	$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	

Some Well Known Examples

Algorithm	Recurrence Relationship	Run time
Binary search	$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$
Binary tree traversal	$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$	$O(n)$
Optimal Sorted Matrix Search	$T(n) = 2T\left(\frac{n}{2}\right) + O(\log n)$	$O(n)$
Merge Sort	$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	$O(n \log n)$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a})$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \leq \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \le \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

and

$$\sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i b^{-i \log_b a} b^{i\varepsilon}$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \le \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

and

$$\sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i b^{-i \log_b a} b^{i\varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i a^{-i} b^{i\varepsilon}$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \leq \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

and

$$\sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i b^{-i \log_b a} b^{i\varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i a^{-i} b^{i\varepsilon}$$

$$= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} b^{\varepsilon i}$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \leq \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

and

$$\sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i b^{-i \log_b a} b^{i\varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i a^{-i} b^{i\varepsilon}$$

$$= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} b^{\varepsilon i} = n^{\log_b a - \varepsilon} \frac{b^{\varepsilon(\log_b n + 1)} - 1}{b^{\varepsilon} - 1}$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \leq \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

and

$$\begin{aligned}
\sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} &= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i b^{-i \log_b a} b^{i\varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i a^{-i} b^{i\varepsilon} \\
&= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} b^{\varepsilon i} = n^{\log_b a - \varepsilon} \frac{b^{\varepsilon(\log_b n + 1)} - 1}{b^\varepsilon - 1} \\
&= n^{\log_b a - \varepsilon} \frac{n^\varepsilon b^\varepsilon - 1}{b^\varepsilon - 1}
\end{aligned}$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \leq \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

and

$$\begin{aligned}
\sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} &= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i b^{-i \log_b a} b^{i\varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i a^{-i} b^{i\varepsilon} \\
&= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} b^{\varepsilon i} = n^{\log_b a - \varepsilon} \frac{b^{\varepsilon(\log_b n + 1)} - 1}{b^\varepsilon - 1} \\
&= n^{\log_b a - \varepsilon} \frac{n^\varepsilon b^\varepsilon - 1}{b^\varepsilon - 1} \leq n^{\log_b a - \varepsilon} \frac{n^\varepsilon b^\varepsilon}{b^\varepsilon - 1}
\end{aligned}$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \le \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

and

$$\sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i b^{-i \log_b a} b^{i\varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i a^{-i} b^{i\varepsilon}$$

$$= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} b^{\varepsilon i} = n^{\log_b a - \varepsilon} \frac{b^{\varepsilon(\log_b n + 1)} - 1}{b^\varepsilon - 1}$$

$$= n^{\log_b a - \varepsilon} \frac{n^\varepsilon b^\varepsilon - 1}{b^\varepsilon - 1} \le n^{\log_b a - \varepsilon} \frac{n^\varepsilon b^\varepsilon}{b^\varepsilon - 1} = n^{\log_b a} \frac{b^\varepsilon}{b^\varepsilon - 1}$$

Proof of Master's Theorem (Case A)

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a}) \leq \sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} + O(n^{\log_b a}), \quad (3)$$

and

$$\begin{aligned}
\sum_{i=0}^{\log_b n} a^i (n/b^i)^{\log_b a - \varepsilon} &= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i b^{-i \log_b a} b^{i\varepsilon} = n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} a^i a^{-i} b^{i\varepsilon} \\
&= n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n} b^{\varepsilon i} = n^{\log_b a - \varepsilon} \frac{b^{\varepsilon(\log_b n + 1)} - 1}{b^{\varepsilon} - 1} \\
&= n^{\log_b a - \varepsilon} \frac{n^{\varepsilon} b^{\varepsilon} - 1}{b^{\varepsilon} - 1} \leq n^{\log_b a - \varepsilon} \frac{n^{\varepsilon} b^{\varepsilon}}{b^{\varepsilon} - 1} = n^{\log_b a} \frac{b^{\varepsilon}}{b^{\varepsilon} - 1} \\
&= O(n^{\log_b a}).
\end{aligned}$$

Combining this with (3), we get $T(n) = O(n^{\log_b a})$.

Examples: Master's method is not applicable

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

  a is not a constant; the number of subproblems should be fixed

Examples: Master's method is not applicable

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

  a is not a constant; the number of subproblems should be fixed

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

  non-polynomial difference between f(n) and $n^{\log_b a}$

$$\boxed{\frac{f(n)}{n^{\log_b a}} = \frac{\frac{n}{\log n}}{n^{log_2 2}} = \frac{n}{n \log n} = \frac{1}{\log n}}$$

Examples: Master's method is not applicable

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

  a is not a constant; the number of subproblems should be fixed

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

  non-polynomial difference between f(n) and $n^{\log_b a}$ $\boxed{\frac{f(n)}{n^{\log_b a}} = \frac{\frac{n}{\log n}}{n^{log_2 2}} = \frac{n}{n \log n} = \frac{1}{\log n}}$

- $T(n) = 0.5T\left(\frac{n}{2}\right) + n$

  a<1 cannot have less than one sub problem

Examples: Master's method is not applicable

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

  a is not a constant; the number of subproblems should be fixed

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

  non-polynomial difference between f(n) and $n^{\log_b a}$

  $$\frac{f(n)}{n^{\log_b a}} = \frac{\frac{n}{\log n}}{n^{log_2 2}} = \frac{n}{n \log n} = \frac{1}{\log n}$$

- $T(n) = 0.5T\left(\frac{n}{2}\right) + n$

  a<1 cannot have less than one sub problem

- $T(n) = 64T\left(\frac{n}{8}\right) - n^2 \log n$

  f(n) which is the combination time is not positive

Examples: Master's method is not applicable

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

  a is not a constant; the number of subproblems should be fixed

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

  non-polynomial difference between f(n) and $n^{\log_b a}$

  $$\frac{f(n)}{n^{\log_b a}} = \frac{\frac{n}{\log n}}{n^{log_2 2}} = \frac{n}{n \log n} = \frac{1}{\log n}$$

- $T(n) = 0.5T\left(\frac{n}{2}\right) + n$

  a<1 cannot have less than one sub problem

- $T(n) = 64T\left(\frac{n}{8}\right) - n^2 \log n$

  f(n) which is the combination time is not positive

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

  case 3 but regularity violation.

Examples: Master's method is not applicable

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

  a is not a constant; the number of subproblems should be fixed

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

  non-polynomial difference between f(n) and $n^{\log_b a}$

  $$\frac{f(n)}{n^{\log_b a}} = \frac{\frac{n}{\log n}}{n^{log_2 2}} = \frac{n}{n \log n} = \frac{1}{\log n}$$

- $T(n) = 0.5T\left(\frac{n}{2}\right) + n$

  a<1 cannot have less than one sub problem

- $T(n) = 64T\left(\frac{n}{8}\right) - n^2 \log n$

  f(n) which is the combination time is not positive

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

  case 3 but regularity violation.

What about Tower of Hanoi ?

$$T_h = 2T_{h-1} + 1$$

Resources used for these slides

- http://en.wikipedia.org/wiki/Master_theorem
- http://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/05DivideAndConquerI.pdf
- http://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/05DivideAndConquerII.pdf
- http://www.cs.cornell.edu/courses/cs3110/2011sp/lectures/lec19-master/mm-proof.pdf

Experience certainty.

Thank You!