Divide and Conquer

Recursively:

Divide the problem into a number of sub-problems

Conquer the sub-problems by solving them recursively

Combine the solutions to the sub-problems into the solution for the original problem

General recurrence: $T(N) = aT(N/b) + f(N)$

Cases solved by divide and conquer

❖ The maximum subsequence sum – the O(N log N) solution

❖ Tree traversals – O(N)

❖ Mergesort and quicksort – O(N log N)

Closest Points Problem

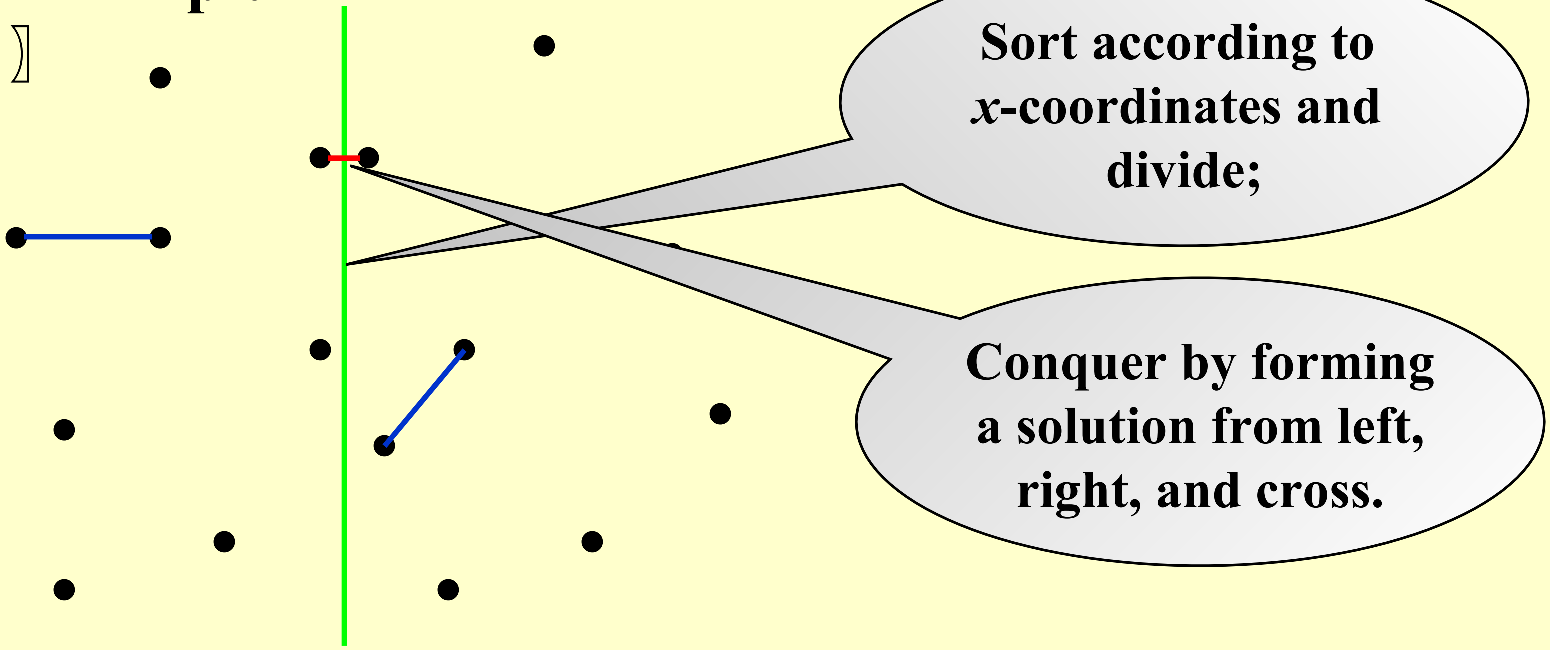
Given N points in a plane. Find the closest pair of points. (If two points have the same position, then that pair is the closest with distance 0.)

- **Simple Exhaustive Search**

Check $N(N-1)/2$ pairs of points. $T = O(N^2)$.

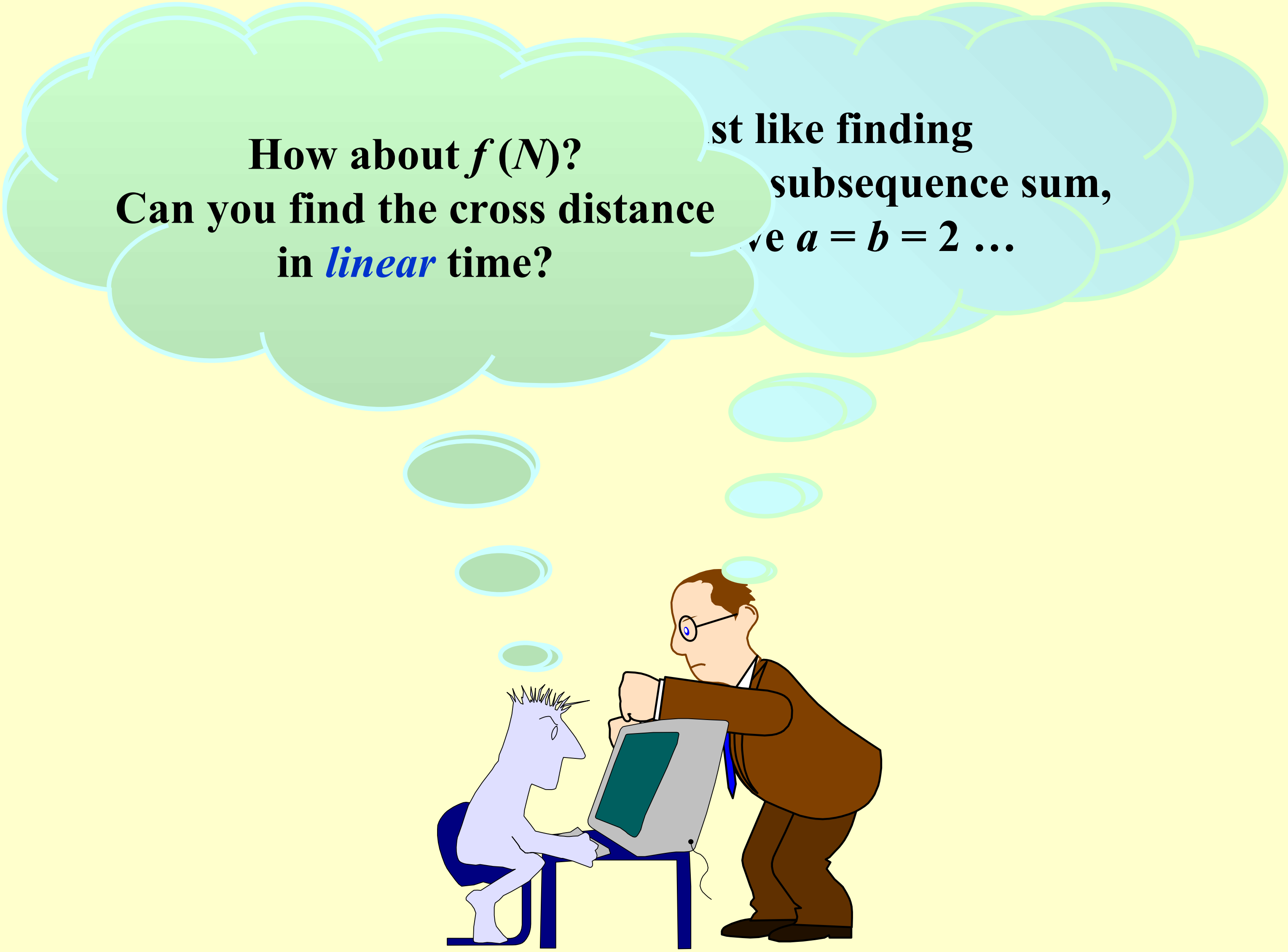
- **Divide and Conquer** – similar to the maximum subsequence sum problem

〖 Example 〗



Sort according to *x*-coordinates and divide;

Conquer by forming a solution from left, right, and cross.



st like finding
subsequence sum,
ve $a = b = 2$ …

How about $f(N)$?
Can you find the cross distance
in *linear* time?

Recall: $T(N) = 2T(N/2) + cN$

$= 2[\ 2T(N/2^2) + cN/2\ ] + cN$

$= 2^2\ T(N/2^2) + 2cN$

$= \ldots\ldots$

$= 2^k\ T(N/2^k) + kcN$

$= N + c\ N\log N\ = O(\ N\log N\ )$

if $T(N) = 2T(N/2) + cN^2$

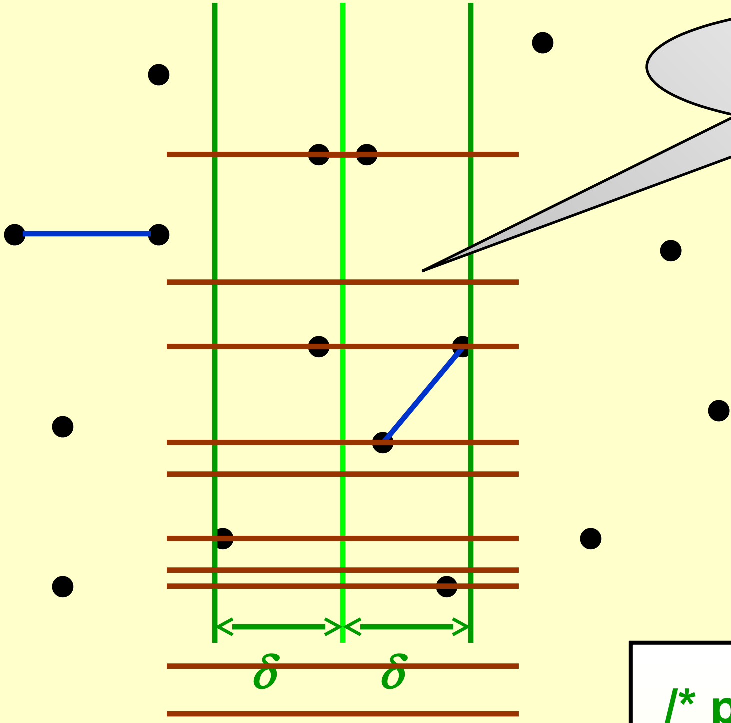
$= 2[\ 2T(N/2^2) + cN^2/2^2\ ] + cN^2$

$= 2^2\ T(N/2^2) + cN^2(1+1/2)$

$= \ldots\ldots$

$= 2^k\ T(N/2^k) + cN^2(1+1/2+\ldots+1/2^{k-1}\ )$

$= O(\ N^2\ )$

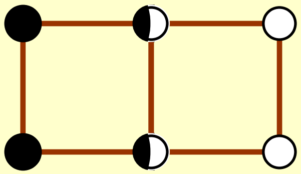


If NumPointInStrip = $O(\sqrt{N})$, we have

```
/* points are all in the strip */
for ( i=0; i<NumPointsInStrip; i++ )
    for ( j=i+1; j<NumPointsInStrip; j++ )
        if ( Dist( $P_i$ , $P_j$ ) < $\delta$ )
              $\delta$ = Dist( $P_i$ , $P_j$ );
```

The worst case: NumPointInStrip = N

The worst case:



For any p_i , at most 7 points are considered.

$f(N) = O(N)$

```
/* points are all in the strip */
/* and sorted by y coordinates */
for ( i = 0; i < NumPointsInStrip; i++ )
    for ( j = i + 1; j < NumPointsInStrip; j++ )
        if ( Dist_y( $P_i$ , $P_j$ ) > $\delta$ )
              break;
        else  if ( Dist( $P_i$ , $P_j$ ) < $\delta$ )
              $\delta$ = Dist( $P_i$ , $P_j$ );
```

Three methods for solving recurrences:

$$T(N) = a\,T(N/b) + f(N)$$

☞ **Substitution method**

☞ **Recursion-tree method**

☞ **Master method**

✂ **Details to be ignored:**

👌 **if (N / b) is an integer or not**

👌 **always assume $T(n) = \Theta(1)$ for small n**

☞ **Substitution method — guess, then prove by induction**

〖 Example 〗 $T(N) = 2\,T(\lfloor N/2 \rfloor) + N$

Guess: $T(N) = O(N \log N)$

Proof: Assume it is true for all $m < N$, in particular for $m = \lfloor N/2 \rfloor$.

Then 0 so that

$T($

Su

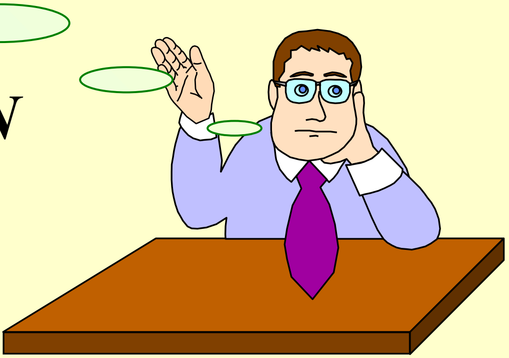
$T($

Relax!
As long as we can choose
sufficiently large c so that
it is true for $T(2)$ and $T(3)$.

$\le 2\,c \lfloor N/2 \rfloor \log \lfloor N/2 \rfloor + N$

$\le c\,N (\log N - \log 2) + N$

$\le c\,N \log N \quad \text{for } c \ge 1$



〖 **Example** 〗 $T(N) = 2\,T(\lfloor N/2 \rfloor) + N$

Wrong guess: $T(N) = O(N)$

Proof: **Assume it is true for all** $m < N$**, in particular for** $m = \lfloor N/2 \rfloor$.

$$T(\lfloor N/2 \rfloor) \le c \lfloor N/2 \rfloor$$

Substituting into the recurrence:

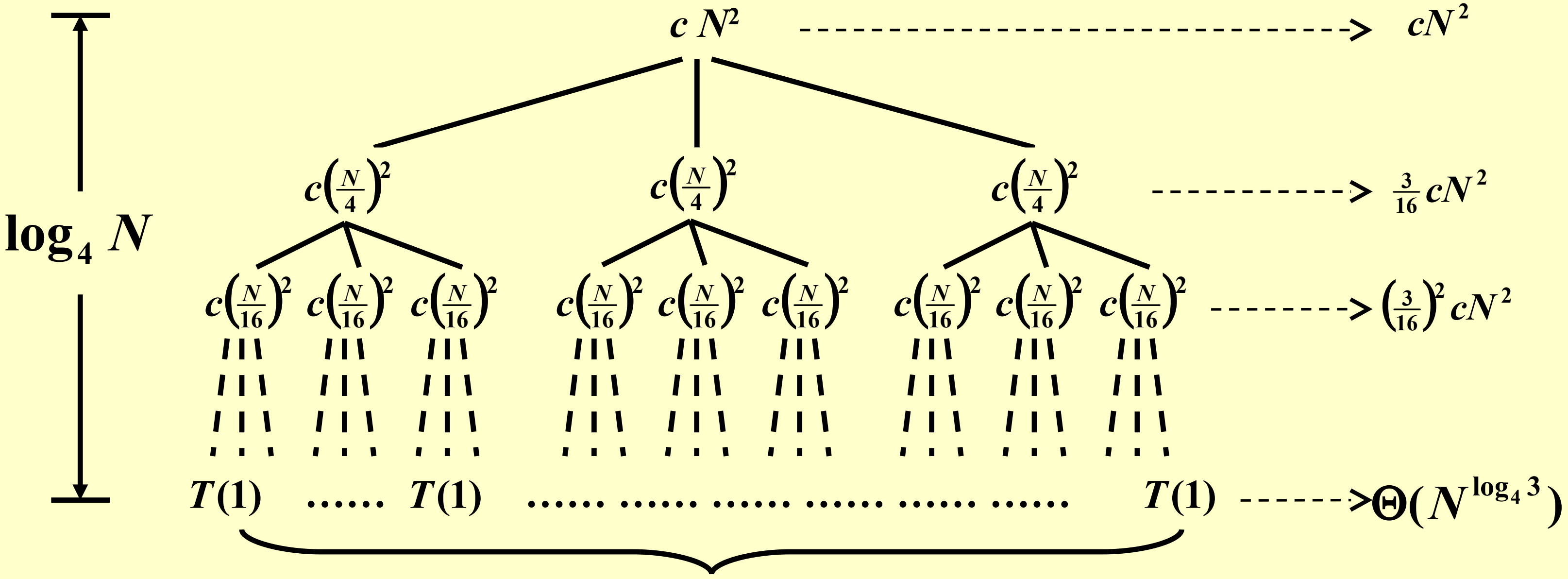
$$\begin{aligned} T(N) &= 2\,T(\lfloor N/2 \rfloor) + N \\ &\le 2\,c \lfloor N/2 \rfloor + N \\ &\le cN + N = O(N) \end{aligned}$$ ✗

Must prove the *exact form*

How to make a good guess?

☞ Recursion-tree method

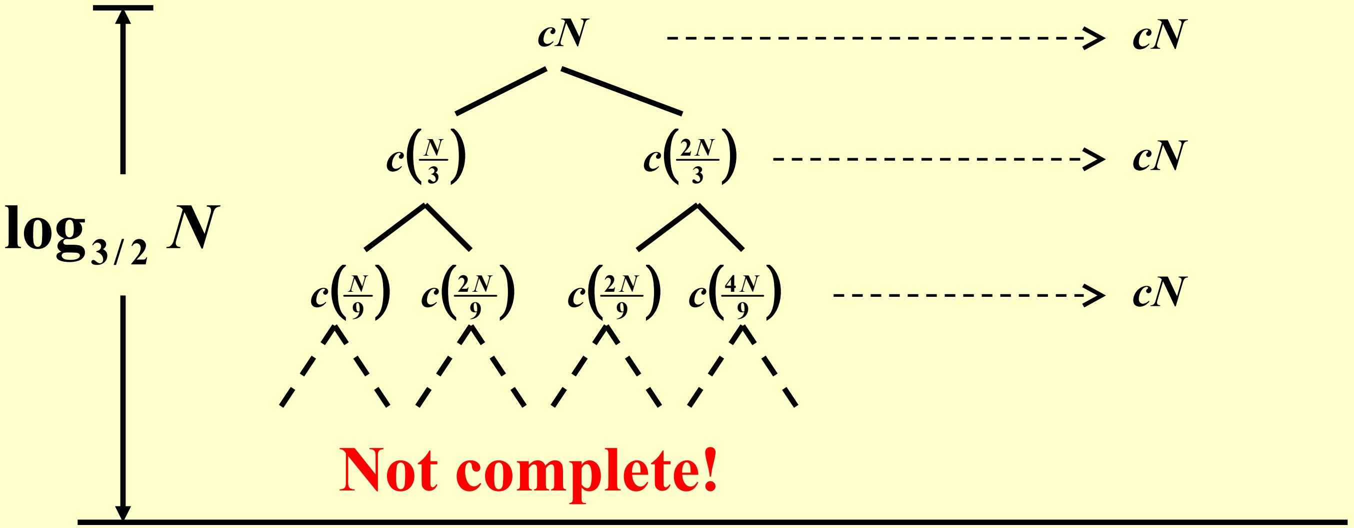
〖 Example 〗 $T(N) = 3\,T(N/4) + \Theta(N^2)$


$\log_4 N$
cN^2
$c\left(\frac{N}{4}\right)^2$
$c\left(\frac{N}{16}\right)^2$
$T(1)$ …… $T(1)$ …… …… …… …… …… …… $T(1)$
cN^2
$\frac{3}{16}cN^2$
$\left(\frac{3}{16}\right)^2 cN^2$
$\Theta(N^{\log_4 3})$

$$3^{\log_4 N} = N^{\log_4 3}$$

$$T(N) = \sum_{i=0}^{\log_4 N - 1} \left(\frac{3}{16}\right)^i cN^2 + \Theta(N^{\log_4 3})$$

〖 **Example** 〗 $T(N) = T(N/3) + T(2N/3) + cN$



$\log_{3/2} N$

cN ----------> cN

$c\left(\frac{N}{3}\right)$ $c\left(\frac{2N}{3}\right)$ ----------> cN

$c\left(\frac{N}{9}\right)$ $c\left(\frac{2N}{9}\right)$ $c\left(\frac{2N}{9}\right)$ $c\left(\frac{4N}{9}\right)$ ----------> cN

Not complete!

Guess: $O(N \log N)$

Proof by substitution:

$$T(N) = T(N/3) + T(2N/3) + cN \le d(N/3)\log(N/3) + d(2N/3)\log(2N/3) + cN$$

$$= dN\log N - dN(\log_2 3 - \tfrac{2}{3}) + cN \le dN\log N$$

$$\text{for } d \ge c/(\log_2 3 - \tfrac{2}{3})$$

☞ Master method

【 Master Theorem 】 Let $a \geq 1$ and $b > 1$ be constants, let $f(N)$ be a function, and let $T(N)$ be defined on the nonnegative integers by the recurrence $T(N) = aT(N/b) + f(N)$. Then:

1. If $f(N) = O(\underline{N}^{\log_b a-\varepsilon})$ for some constant $\varepsilon > 0$, then $T(N) = \Theta(\underline{N}^{\log_b a})$
2. If $f(N) = \Theta(N^{\log_b a})$, then $T(N) = \Theta(N^{\log_b a} \log N)$
3. If $\underline{f(N)} = \Omega(N^{\log_b a+\varepsilon})$ for some constant $\varepsilon > 0$, and if $af(N/b) < cf(N)$ for some constant $c < 1$ and all sufficiently large N, then $T(N) = \Theta(\underline{f(N)})$

regularity condition

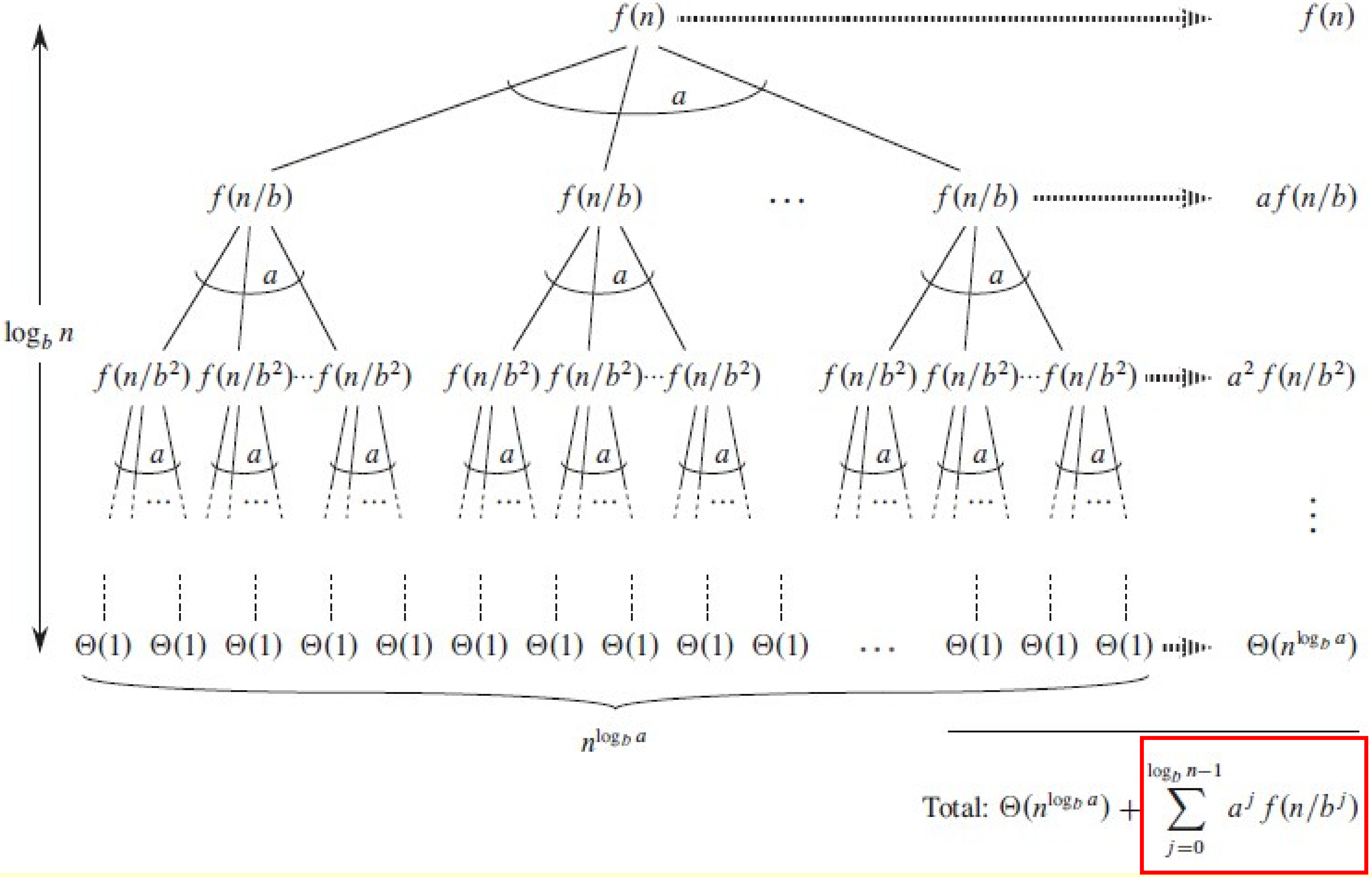
〖 Example 〗 Mergesort has $a = b = 2$, and case 2

→ $T = O(N \log N)$

〖 Example 〗 $a = b = 2,\ f(N) = N\log N$?

→ $T = O(N \cancel{\log} N)$

Proof by recursion tree: for $n = b^k$ for some integer k



$$\text{Total: } \Theta(n^{\log_b a}) + \boxed{\sum_{j=0}^{\log_b n-1} a^j f(n/b^j)}$$

For case 1 where $f(N) = O(N^{\log_b a-\varepsilon})$

$$\sum_{j=0}^{\log_b N-1} a^j f(N/b^j) =$$

$$= O(N^{\log_b a-\varepsilon}\sum_{j=0}^{\log_b N-1}(b^{\varepsilon})^j) = O(N^{\log_b a-\varepsilon}\frac{b^{\varepsilon\log_b N}-1}{b^{\varepsilon}-1})$$

$$= O(N^{\log_b a-\varepsilon}N^{\varepsilon}) = O(N^{\log_b a})$$

$$T(N) = \Theta(N^{\log_b a}) + O(N^{\log_b a}) = \Theta(N^{\log_b a})$$

Discussion 9:
Please prove case 2.

Read Ch.4 of “Introduction to Algorithms” for the rest of the proof.

☞ Master method – *another form*

【 **Master Theorem** 】 The recurrence $T(N) = aT(N/b) + f(N)$ can be solved as follows:

1. If $af(N/b) = \kappa f(N)$ for some constant $\kappa < 1$, then $T(N) = \Theta(f(N))$
2. If $af(N/b) = K f(N)$ for some constant $K > 1$, then $T(N) = \Theta(N^{\log_b a})$
3. If $af(N/b) = f(N)$, then $T(N) = \Theta(f(N) \log_b N)$

〖 **Example** 〗 $a = 4$, $b = 2$, $f(N) = N\log N$

$$af(N/b) = 4(N/2)\log(N/2) = 2N\log N - 2N \text{ ?}$$

$$f(N) = N\log N \qquad O(N^{\log_b a - \varepsilon}) = O(N^{2-\varepsilon})$$

➡ $T = O(N^2)$

【 Theorem 】 The solution to the equation

$$T(N) = a\,T(N/b) + \Theta(N^k \log^p N),$$

where $a \geq 1$, $b > 1$, and $p \geq 0$ is

$$T(N) = \begin{cases} O(N^{\log_b a}) & \text{if } a > b^k \\ O(N^k \log^{p+1} N) & \text{if } a = b^k \\ O(N^k \log^p N) & \text{if } a < b^k \end{cases}$$

〖 Example 〗 Mergesort has $a = b = 2$, $p = 0$ and $k = 1$.

→ $T = O(N \log N)$

〖 Example 〗 Divide with $a = 3$, and $b = 2$ for each recursion;

Conquer with O(N) – that is, $k = 1$ and $p = 0$.

→ $T = O(N^{1.59})$

If conquer takes O(N^2) then $T = O(N^2)$.

〖 Example 〗 $a = b = 2$, $f(N) = N \log N$ → $T = O(N \log^2 N)$

Reference:

Introduction to Algorithms, 3rd Edition: Ch.4, p. 65-113 ； ***Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. The MIT Press. 2009***

Data Structure and Algorithm Analysis in C (2nd Edition)： Ch.10 ， p.370-375 ； ***M.A.Weiss 著、陈越改编，人民邮件出版社， 2005***

Lecture Notes of CS 373: Combinatorial Algorithms: Notes on Solving Recurrence Relations, p.10-13; ***Jeff Erickson, University of Illinois, Urbana-Champaign, 2003***