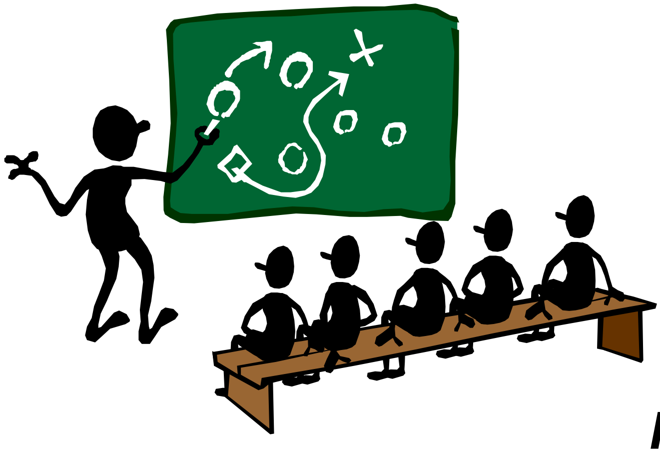
Algorithms – Chapter 4
Divide-and-Conquer



Juinn-Dar Huang

Professor

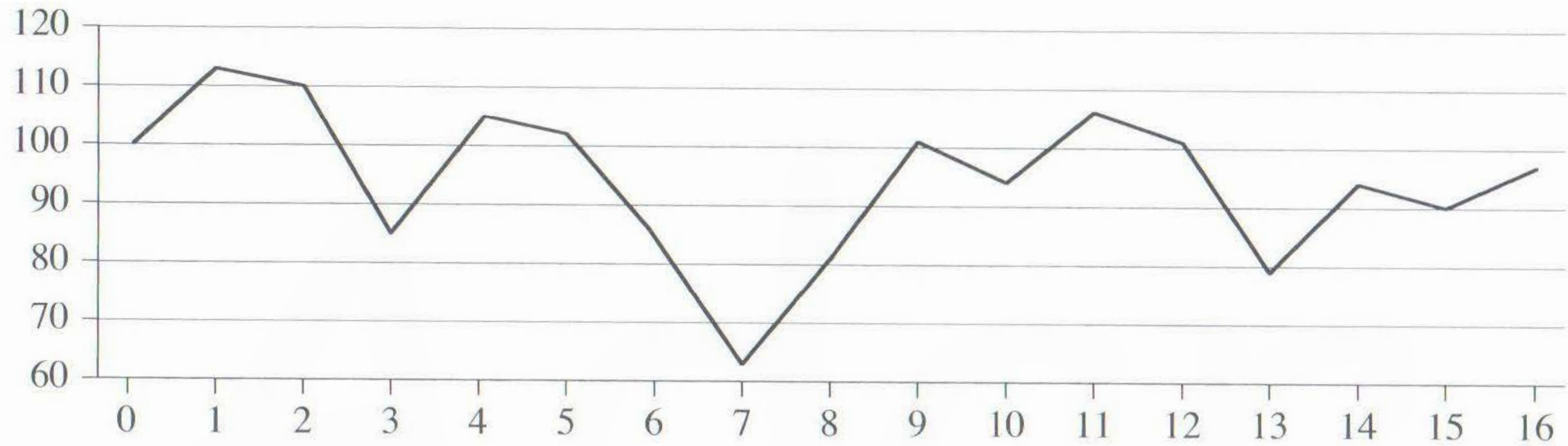
jdhuang@mail.nctu.edu.tw

August 2007

Rev. ’08, ’11, ’12, ’15, ’16, ’18, ’19, ’20, ’21

Juinn-Dar Huang
jdhuang@mail.nctu.edu.tw

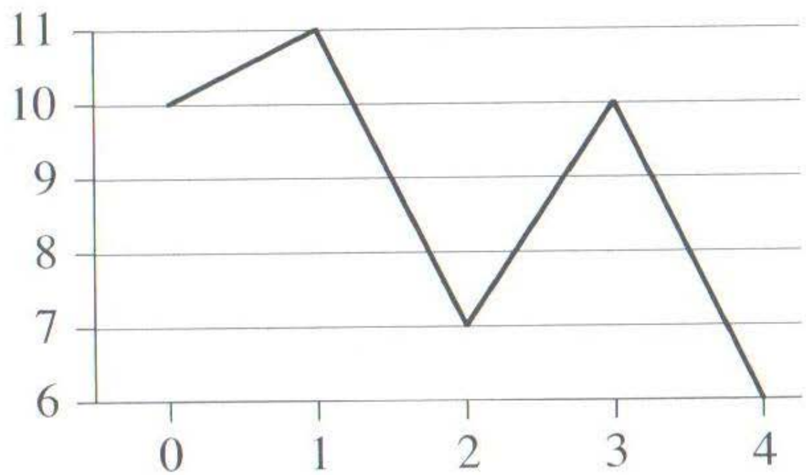
Maximum Subarray Problem (1/2)



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	−3	−25	20	−3	−16	−23	18	20	−7	12	−5	−22	15	−4	7

- Brute-force method ➔ check every day pair
  - time complexity ➔ $\Theta(n^2)$
- Any better idea?

Maximum Subarray Problem (2/2)

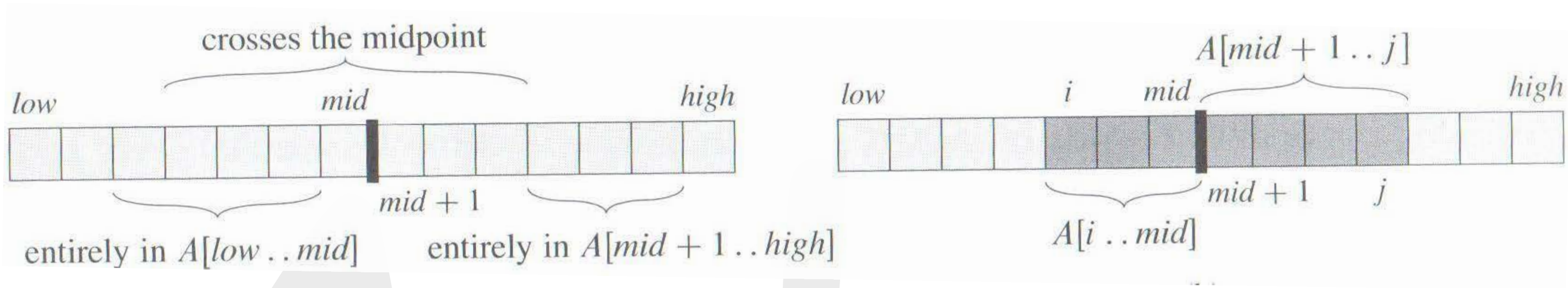


Day	0	1	2	3	4
Price	10	11	7	10	6
Change		1	−4	3	−4

- Simple heuristics don't work!
  - buy lowest, sell highest (impossible sometimes)
  - buy lowest, sell highest on some day later
  - sell highest, buy lowest on some day before

Juinn-Dar Huang
jdhuang@mail.nctu.edu.tw

Divide-and-Conquer



- Find a midpoint day (mid) of A[low..high]
  ➔ only 3 possible exclusive scenarios
  – a subarray is entirely in A[low..mid]
  – a subarray is entirely in A[mid+1..high]
  – a subarray A[i..j], where low ≤ i ≤ mid and mid < j ≤ high
- Find a maximum one for each scenario and then pick a maximum one out of the three

Max Subarray Crossing Midpoint

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
// Find a maximum subarray of the form A[i . . mid].
left-sum = -∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
// Find a maximum subarray of the form A[mid + 1 . . j]
right-sum = -∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
// Return the indices and the sum of the two subarrays.
return (max-left, max-right, left-sum + right-sum)
```

Time complexity: $\Theta(n)$, where $n \propto$ (high – low)

Juinn-Dar Huang jdhuang@mail.nctu.edu.tw

Finding Maximum Subarray

```
FIND-MAXIMUM-SUBARRAY(A, low, high)

if high == low
    return (low, high, A[low])                    // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

Initial call: FIND-MAXIMUM-SUBARRAY$(A, 1, n)$

Juinn-Dar Huang jdhuang@mail.nctu.edu.tw

Time Complexity Analysis

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

$\Rightarrow$ $T(n) = \Theta(n \lg n)$ **by the master theorem**

- Just like Merger Sort

Linear-Time Solution

Self-Study

```
MAX-SUBARRAY-LINEAR(A)
n = A.length
max-sum = -∞
ending-here-sum = -∞
for j = 1 to n
    ending-here-high = j
    if ending-here-sum > 0
        ending-here-sum = ending-here-sum + A[j]
    else ending-here-low = j
        ending-here-sum = A[j]
    if ending-here-sum > max-sum
        max-sum = ending-here-sum
        low = ending-here-low
        high = ending-here-high
return (low, high, max-sum)
```

Time complexity: Θ(n), Brilliant!!

Recurrences

- Recurrence
  - an equation or inequality that describes a function in terms of its value on smaller inputs
- Example

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

- Solving a recurrence [$T(n) = aT(n/b) + f(n)$]
  - substitution method
  - recursion-tree method
  - master method

Juinn-Dar Huang jdhuang@mail.nctu.edu.tw

Technicalities

- In practice, certain details are neglected when stating and solving recurrences

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

- Assumptions
  - integer argument $T(n)=\begin{cases}\Theta(1) & \text{if } n=1\\ T(\lceil n/2\rceil)+T(\lfloor n/2\rfloor)+\Theta(n) & \text{if } n>1\end{cases}$
  - boundary conditions are ignored
  - floors and ceilings are ignored

➡ T(n) = 2T(n/2) + Θ(n)

Juinn-Dar Huang
jdhuang@mail.nctu.edu.tw

Substitution Method

- 2 steps
  - guess the form of the solution
  - use math induction to find the constants and show that the solution works
- Key
  - you have to make a right guess!

Juinn-Dar Huang
jdhuang@mail.nctu.edu.tw

Example

- Determine an upper bound of T(n) = 2T(⌊n/2⌋)+n
- Guess (assumption) : T(n) = O(nlgn)
  - is it right?
  - we have to prove ∃ c > 0 s.t. T(n) ≤ cnlgn

$$T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor$$

$$T(n) \leq 2(c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor) + n \leq cn\lg\frac{n}{2} + n$$

$$= cn\lg n - cn\lg 2 + n \leq cn\lg n \quad (\text{if } c \geq 1)$$

Subtleties

- Recurrence
  - $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$
- Guess: T(n) = O(n)
- Assumption: $T(n) \le cn$

$$T(n) \le c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 = \boxed{cn + 1 \not\le cn}$$

- Another assumption: $T(n) \le cn - b$

$$T(n) \le (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1$$
$$= cn - 2b + 1 \le cn - b \quad (\text{choose } b \ge 1)$$

Juinn-Dar Huang
jdhuang@mail.nctu.edu.tw

Avoiding Pitfalls

$$\begin{cases} T(n) = 2T(\lfloor n/2 \rfloor) + n \\ T(1) = 1 \end{cases}$$

- Guess: T(n) = O(n)
- Assumption: $T(n) \le cn$

$$T(n) \le 2(c\lfloor n/2 \rfloor) + n \le \boxed{cn + n = O(n)}$$ **wrong!!**

$$T(n) \le 2(c\lfloor n/2 \rfloor) + n \le \boxed{cn + n \nleq cn}$$ **correct**

Juinn-Dar Huang
jdhuang@mail.nctu.edu.tw

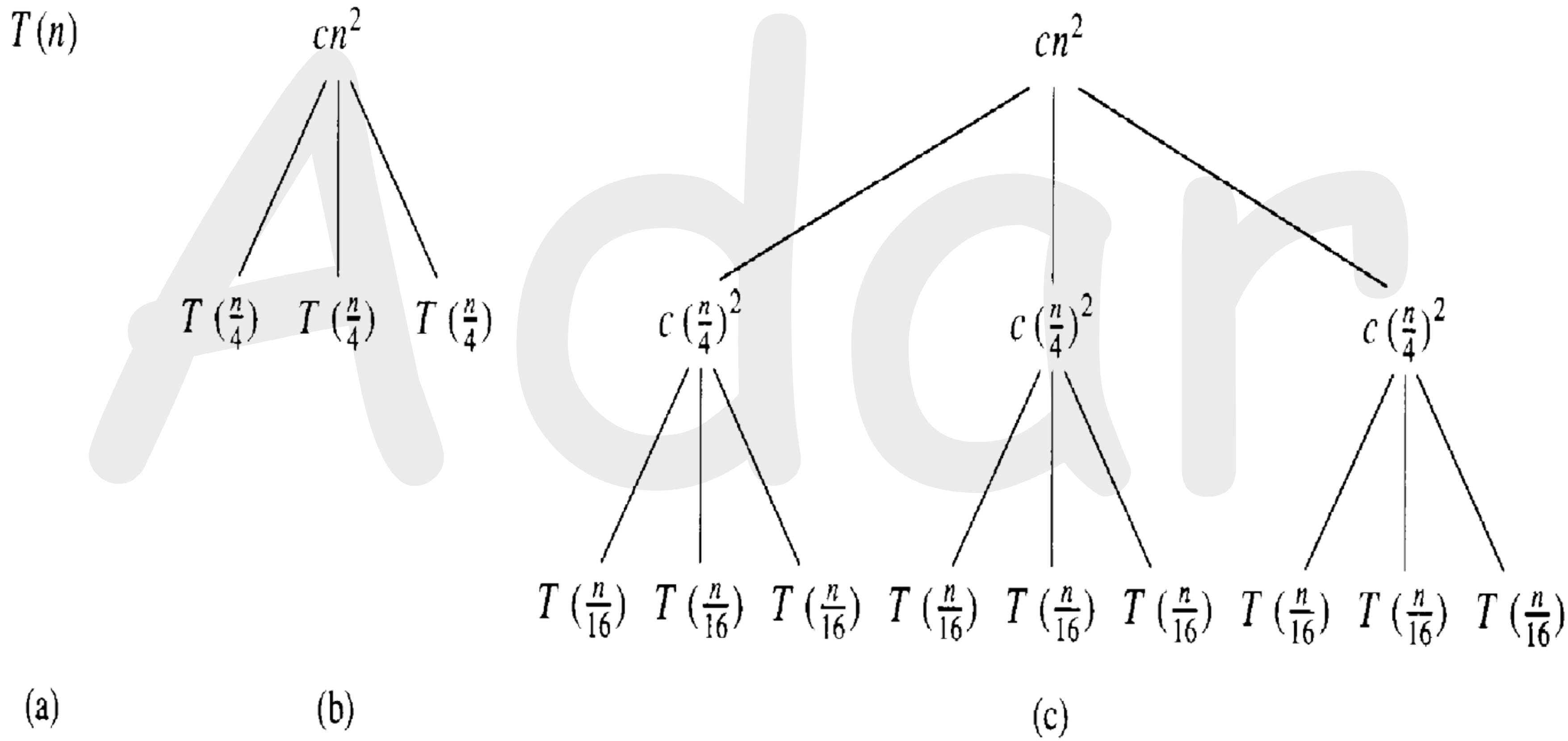
Make a Good Guess

- The substitution is powerful if there is a good guess
- How to make a good guess?
  - make the same guess if the recurrence is similar to one you've seen before
  - try loose upper/lower bounds first, then reduce the range of uncertainty
  - use the recursion-tree method

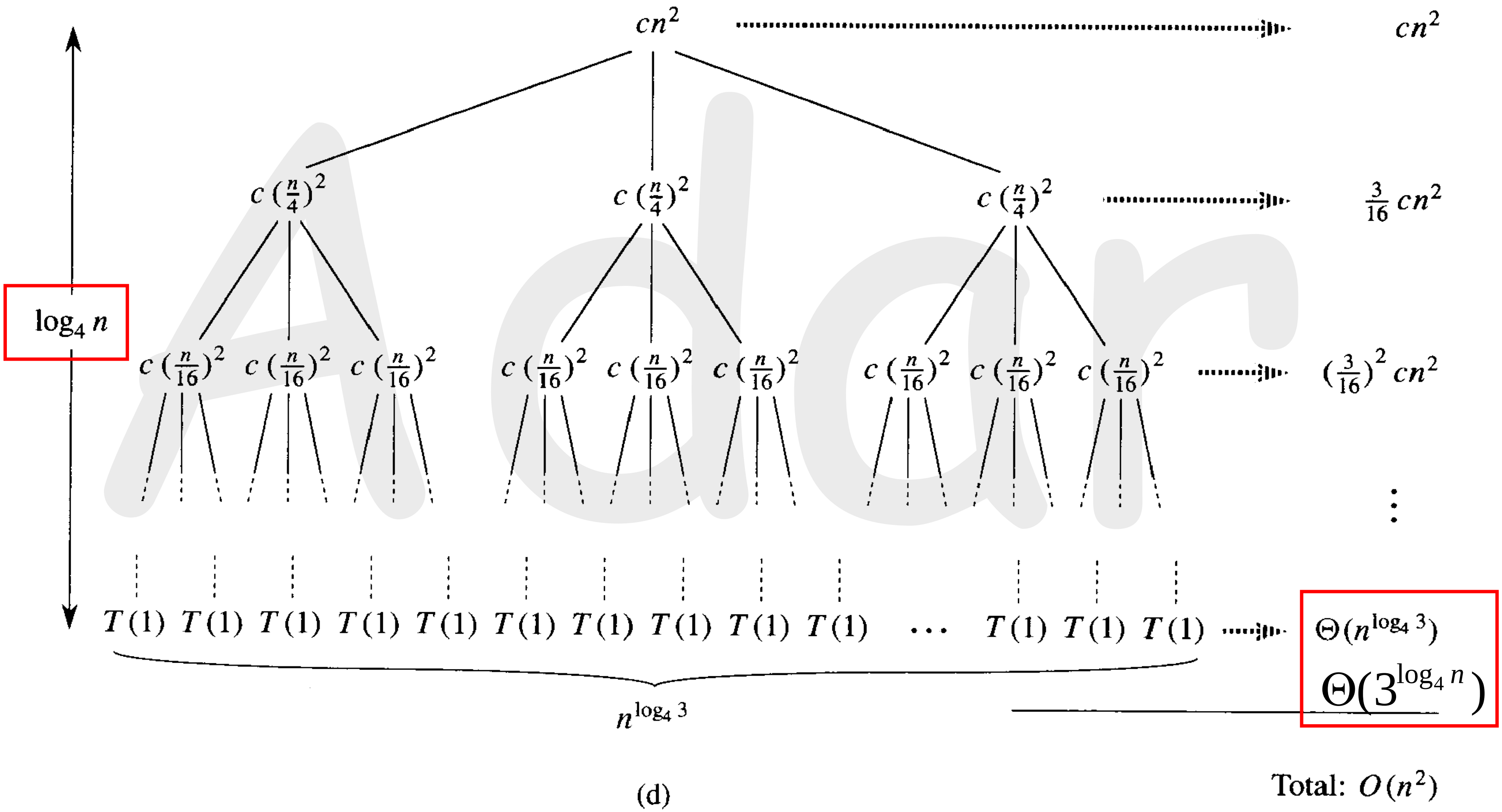
Juinn-Dar Huang jdhuang@mail.nctu.edu.tw

Example: Recursion-Tree Method (1/4)

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$$



Example: Recursion-Tree Method (2/4)



Juinn-Dar Huang jdhuang@mail.nctu.edu.tw

Example: Recursion-Tree Method (3/4)

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \ldots + \left(\frac{3}{16}\right)^{\log_4 n-1} cn^2 + \Theta(n^{\log_4 3})$$

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

$$= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3}).$$

$$T(n) = \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) < \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right)$$

$$= \frac{1}{1 - (3/16)} cn^2 + \Theta\left(n^{\log_4 3}\right) = \frac{16}{13} cn^2 + \Theta(n^{\log_4 3})$$

$$= O(n^2)$$

Juinn-Dar Huang jdhuang@mail.nctu.edu.tw

Example: Recursion-Tree Method (4/4)

Verify by the substitution method

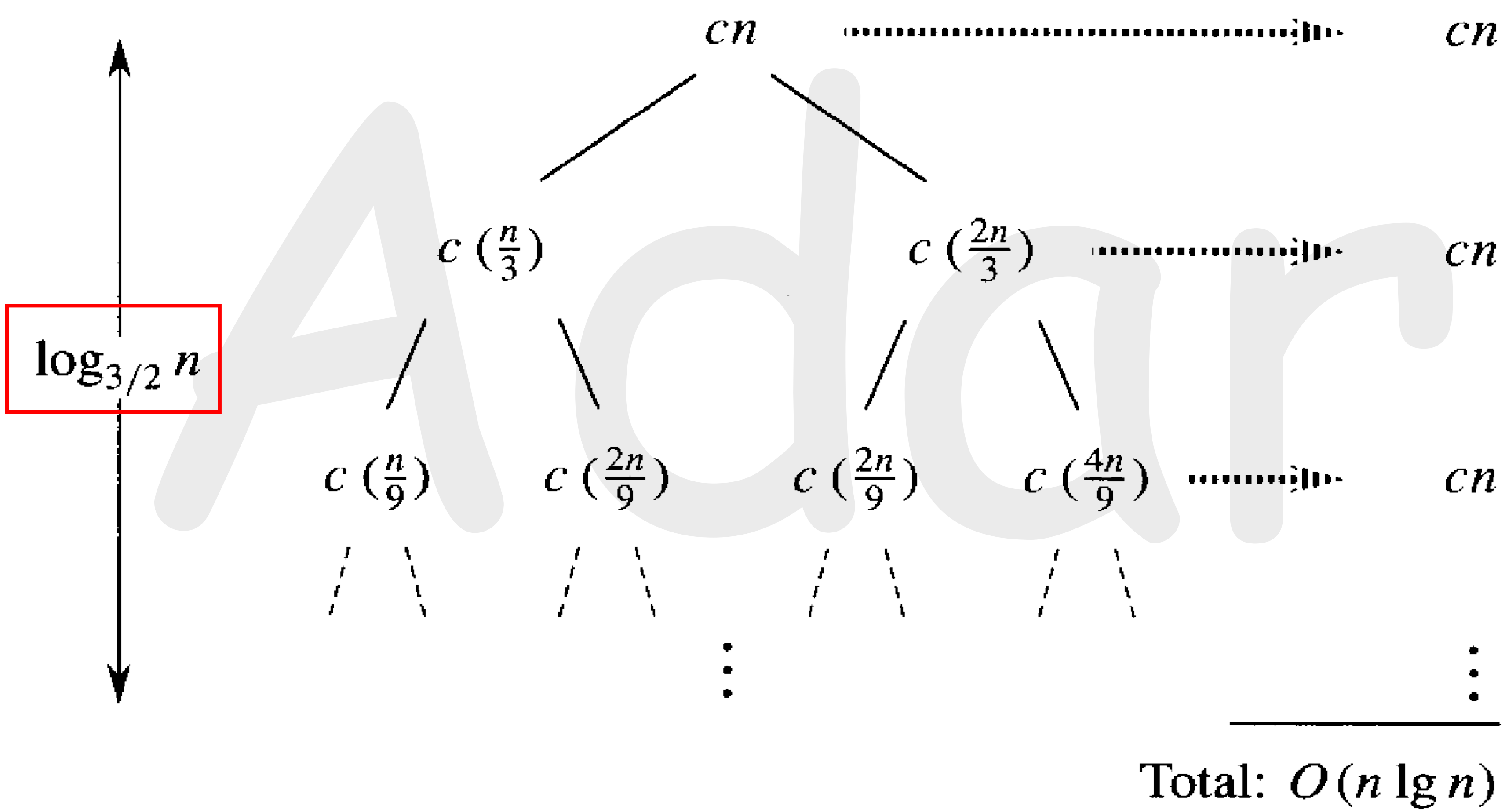
- Guess: T(n) = O(n^2)
- Assumption: T(n) ≤ dn^2 for some constant d > 0

$$
\begin{aligned}
T(n) &\le 3T(\lfloor n/4 \rfloor) + cn^2 \\
&\le 3d\lfloor n/4 \rfloor^2 + cn^2 \\
&\le 3d(n/4)^2 + cn^2 \\
&= \frac{3}{16}dn^2 + cn^2 \\
&\le dn^2, \text{ where } d \ge \frac{16}{13}c
\end{aligned}
$$

Juinn-Dar Huang
jdhuang@mail.nctu.edu.tw

Another Example (1/2)

$$T(n) = T(n/3) + T(2n/3) + cn$$



Another Example (2/2)

Verify by the substitution method

- Guess: T(n) = O(nlgn)
- Assumption: T(n) ≤ dnlgn for some constant d > 0

$$
\begin{aligned}
T(n) &\le T(n/3) + T(2n/3) + cn \\
&\le d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= (d(n/3)\lg n - d(n/3)\lg 3) + (d(2n/3)\lg n - d(2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn\lg n - dn(\lg 3 - 2/3) + cn \\
&\le dn\lg n, \ \text{ where } d \ge c/(\lg 3 - 2/3)
\end{aligned}
$$

Master Method

- Master Theorem
  - let a ≥ 1 and b > 1 be constants,
  - let f(n) be a function,
  - let T(n) be a recurrence $T(n) = aT(n/b) + f(n)$
  - then T(n) can be bounded asymptotically

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

Juinn-Dar Huang jdhuang@mail.nctu.edu.tw

Examples (1/2)

- $T(n) = 9T(n/3) + n$

$a = 9, b = 3, f(n) = n$

$n^{\log_3 9} = n^2, \quad f(n) = O(n^{\log_3 9 - 1})$

$Case \quad 1 \Rightarrow T(n) = \Theta(n^2)$

- $T(n) = T(2n/3) + 1$

$a = 1, b = 3/2, f(n) = 1$

$n^{\log_{3/2} 1} = n^0 = \Theta(1) = f(n),$

$Case 2 \Rightarrow T(n) = \Theta(\lg n)$

Juinn-Dar Huang jdhuang@mail.nctu.edu.tw

Examples (2/2)

- $T(n) = 3T(n/4) + n \lg n$

$a = 3, b = 4, f(n) = n \lg n$

$n^{\log_4 3} = n^{0.793}, f(n) = \Omega(n^{\log_4 3 + \varepsilon})$

$Case 3 \Rightarrow$

Check

$af(n/b) = 3(\frac{n}{4})\lg(\frac{n}{4}) \le \frac{3}{4} n \lg n = cf(n)$

for $c = \frac{3}{4}$, and sufficiently large n

$\Rightarrow T(n) = \Theta(n \lg n)$

