طراحی و تحلیل الگوریتم ها

دکتر امیر لکی زاده

استادیار گروه مهندسی کامپیوتر دانشگاه قم

روش های طراحی الگوریتم ها

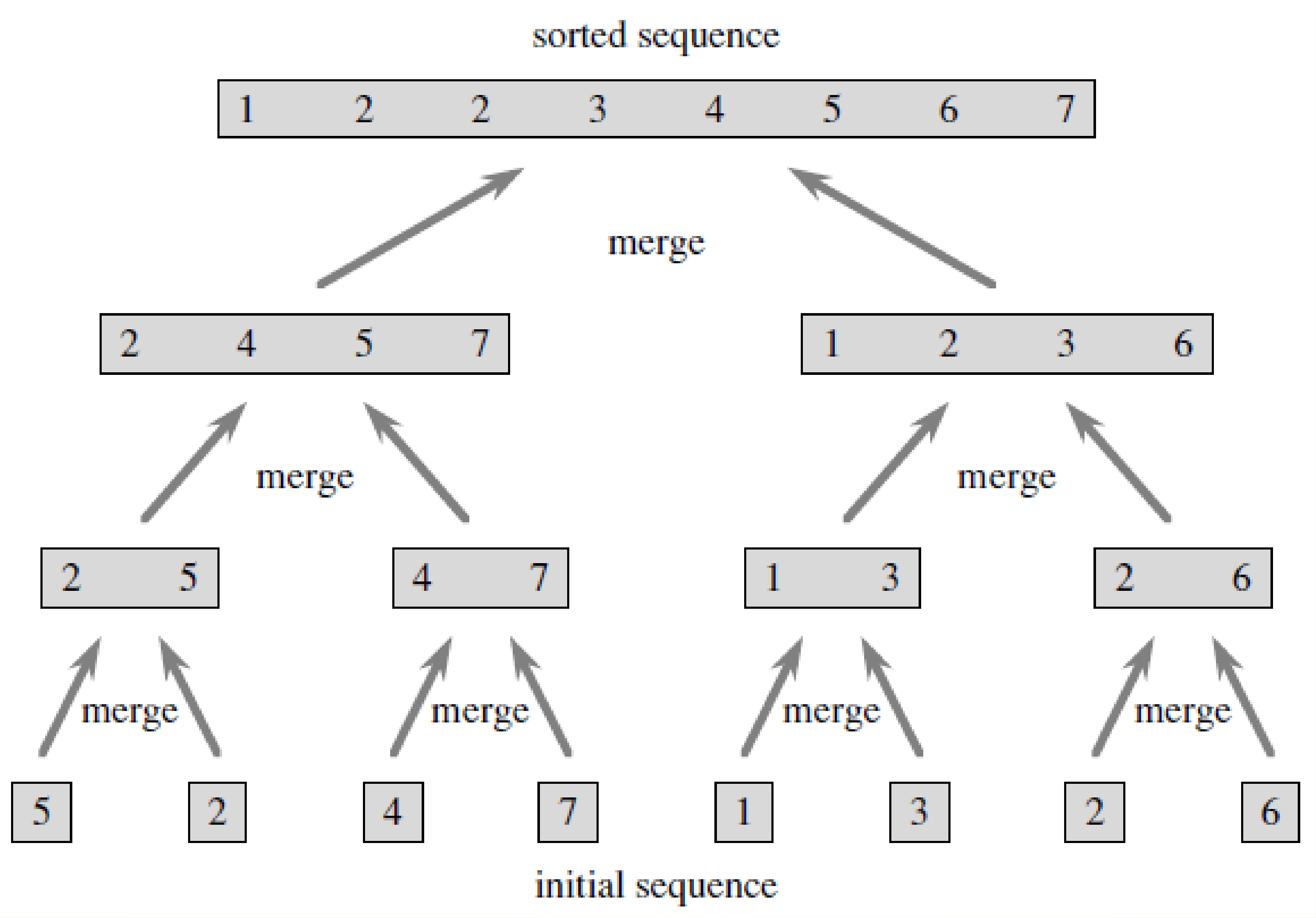
- ـ روش تقسیم و حل (Divide & conquer)
- ـ برنامه نویسی پویا (Dynamic Programming)
- ـ الگوریتم های حریصانه (Greedy Algorithms)
- ـ بازگشت به عقب (Back Tracking)
- ـ شاخه و هرس (Branch & bound)

روش تقسیم و حل (D & G)

- یک روش بازگشتی است و از سه بخش تشکیل شده است:
- ۱ـ تقسیم (Divide): تقسیم مسأله به تعدادی زیر مسأله (عمل تقسیم تا جایی ادامه می یابد که اندازه هر مسأله به مقدار کافی کوچک شود).
- ۲ـ حل زیر مسأله ها (conquer): حل تک تک زیرمسئله ای کوچک
- ۳ـ ترکیب (combine): ترکیب حل زیر مسأله های کوچک و ساختن یک حل برای مسأله اصلی

روش تقسیم و حل (D & G)

مرتب سازی ادغام :



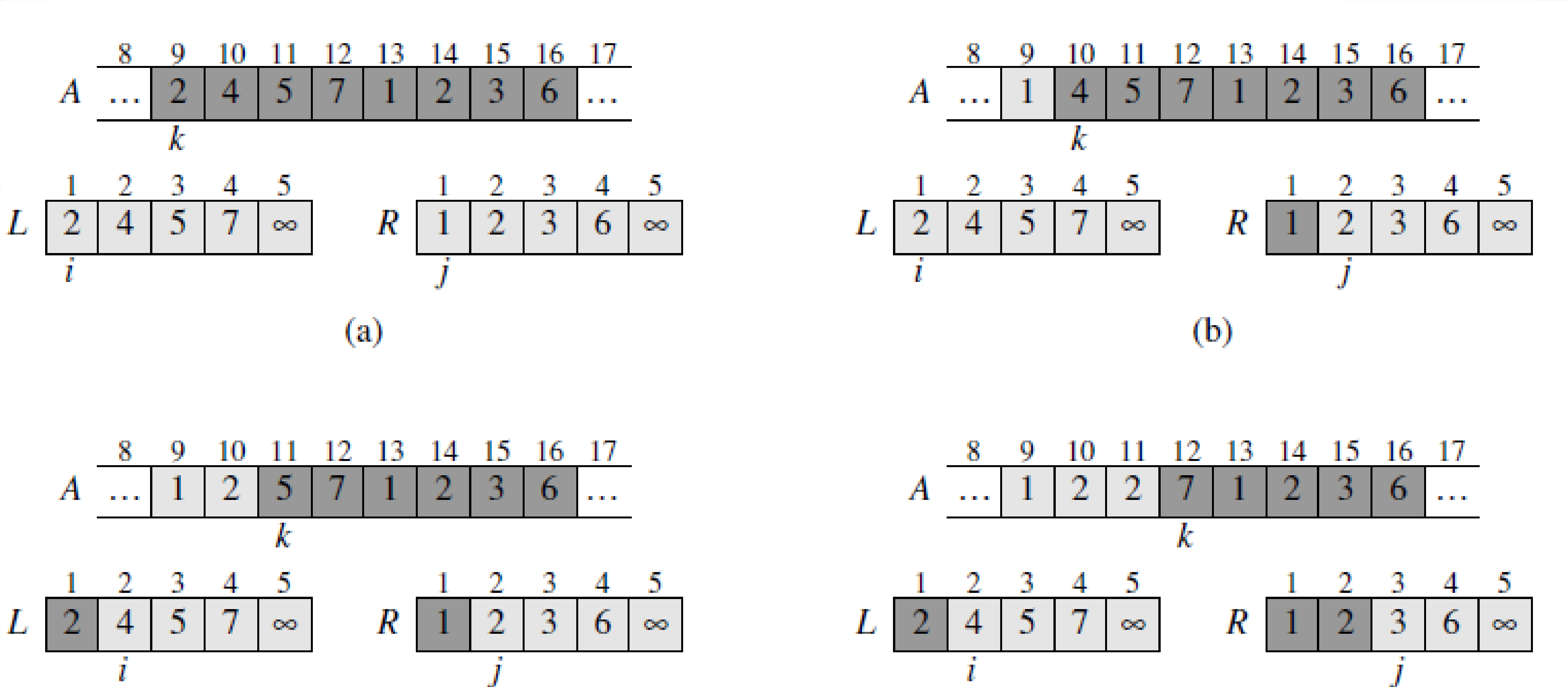
روش تقسیم و حل (D & G)

مرتب سازی ادغام

```
MergeSort(A, left, right) {
   if (left < right) {
    mid = floor((left + right) / 2);
    MergeSort(A, left, mid);
    MergeSort(A, mid+1, right);
    Merge(A, left, mid, right);
   }
}
// Merge() takes two sorted subarrays of A and
// merges them into a single sorted subarray of A.
// Code for this is in the book.  It requires O(n)
// time, and *does* require allocating O(n) space
```

روش تقسیم و حل (D & G)

ادغام دو لیست مرتب



(a) (b)

روش تقسیم و حل (D & G)

- الگوریتم ادغام

```
MERGE(A, p, q, r)
  n1 ← q − p + 1
  n2 ← r − q
  create arrays L[1 .. n1 + 1] and R[1 .. n2 + 1]
  for i ← 1 to n1
       do L[i] ← A[p + i − 1]
  for j ← 1 to n2
       do R[j] ← A[q + j]
  L[n1 + 1] ← ∞
  R[n2 + 1] ← ∞
  i ← 1
  j ← 1
  for k ← p to r
       do if L[i] ≤ R[j]
             then A[k] ← L[i]
                  i ← i + 1
             else A[k] ← R[j]
                  j ← j + 1
```

روش تقسیم و حل (D & G)

- بطور کلی پیچیدگی زمانی الگوریتم های D & G:

$$T(n) = \begin{cases} \theta(1) & n \le c \\ a\,T\left(\frac{n}{b}\right) + D(n) + C(n) & n > c \end{cases}$$

$$\text{Merge} - \text{Sort}$$

$$T(n) = \begin{cases} \theta(1) & n > 1 \\ 2\,T\left(\frac{n}{2}\right) + \theta(n) & n > 1 \end{cases} \qquad T(n) = \begin{cases} C & n = 1 \\ 2\,T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

D(n): مرحله تقسیم، فقط میانه زیر آرایه را تعیین می کند که زمان ثابتی را مصرف می کند بنابراین $D(n) = \theta(1)$.

$$C(n) + D(n) = \theta(n) + \theta(1) = \theta(n)$$

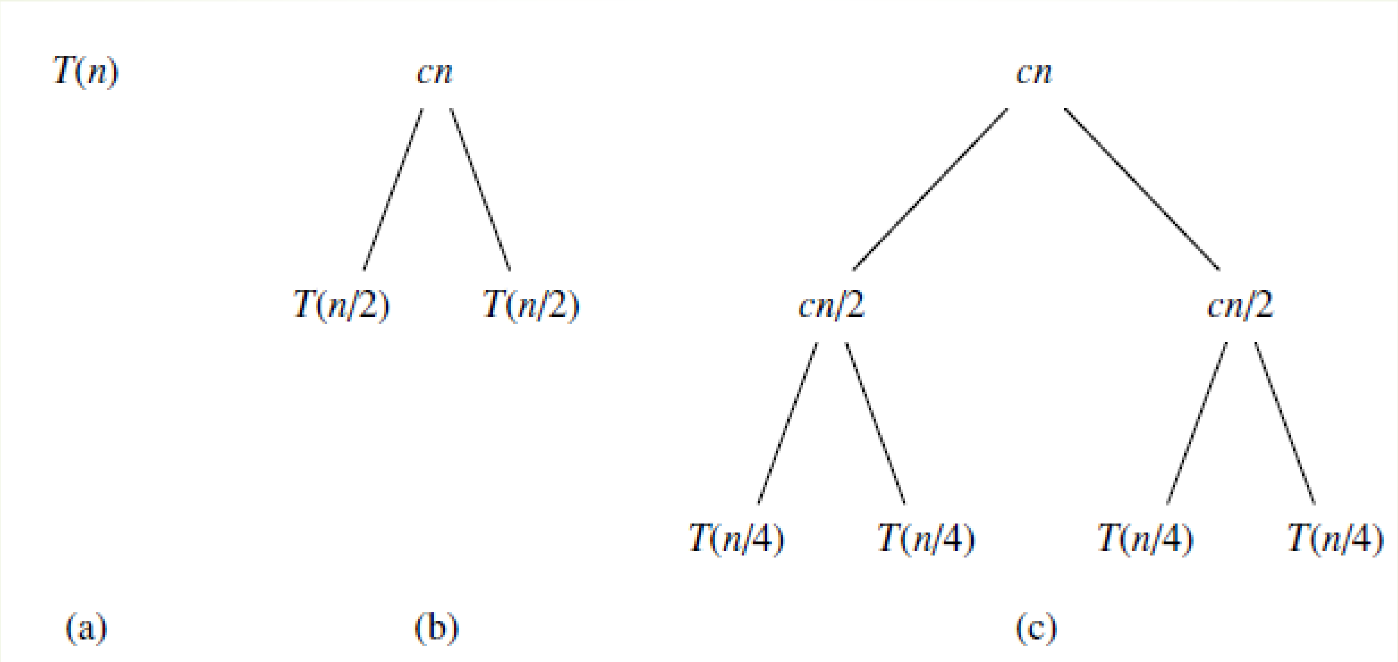
C (n): با توجه به اینکه رویه Merge روی دو زیر آرایه مرتب صورت می گیرد و حداکثر n مقایسه صورت می گیرد بنابراین داریم $C(n) = \theta(n)$

حل روابط بازگشتی

- منظور از حل یک رابطه بازگشتی مثل T(n):

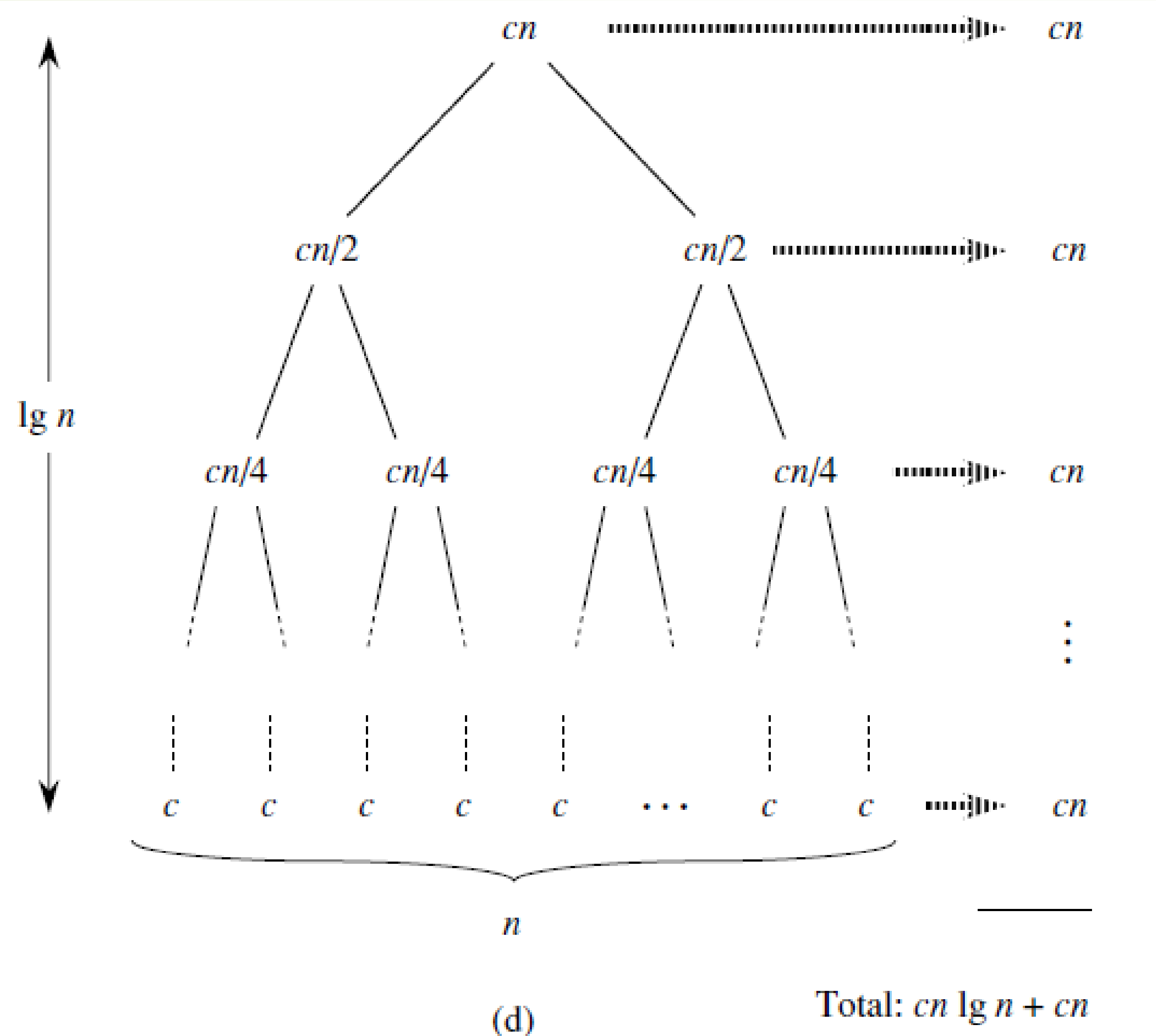
$T(n) = O\ (?)$
$T(n) = \theta\ (?)$

معرفی recursion tree :



حل روابط بازگشتی

معرفی recursion tree :



حل روابط بازگشتی

2.3-7 ★

Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

2-1 ***Insertion sort on small arrays in merge sort***

Although merge sort runs in $\Theta(n \lg n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small n. Thus, it makes sense to use insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

a. Show that the n/k sublists, each of length k, can be sorted by insertion sort in $\Theta(nk)$ worst-case time.

حل روابط بازگشتی

2-3 *Correctness of Horner's rule*

The following code fragment implements Horner's rule for evaluating a polynomial

$$
\begin{aligned}
P(x) &= \sum_{k=0}^{n} a_k x^k \\
&= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x a_n) \cdots)) ,
\end{aligned}
$$

given the coefficients $a_0, a_1, \ldots, a_n$ and a value for x:

```
y ← 0
i ← n
while i ≥ 0
        do y ← a_i + x · y
           i ← i − 1
```

a. What is the asymptotic running time of this code fragment for Horner's rule?

حل روابط بازگشتی

2-4 *Inversions*

Let $A[1 \,..\, n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an ***inversion*** of A.

a. List the five inversions of the array $\langle 2, 3, 8, 6, 1 \rangle$.

b. What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many does it have?

c. What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

d. Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \lg n)$ worst-case time. (*Hint:* Modify merge sort.)

حل روابط بازگشتی

۱. جایگذاری با تکرار

$$T(n) = 2\,T(n/2) + cn$$

$$T\left(n = 2^i\right) = 2\,T\left(2^{i-1}\right) + C2^i$$

$$= 2\left(2\,T(2^{i-2}\right) + C2^{i-1}) + C2^i$$

$$= 2^2 T\left(2^{i-2}\right) + C2 \times 2^i$$

.

.

.

$$= 2^i\,T(1) + C \times i \times 2^i = Cn + C \times n\log n \rightarrow T(n) \;\; = O(n\log n)$$

حل روابط بازگشتی

۲. معادله مشخصه characteristic function

$$a_n = 5a_{n-1} - 6\,a_{n-2} = 0$$

$$r^2 - 5r + 6 = 0 \begin{cases} r_1 = 2 \\ r_2 = 3 \end{cases} \qquad a_n = C_1 2^n + C_2 3^n$$

۳. توابع مولد Generating function

$$a_0, a_1, \ldots a_n, \ldots$$

$$A\,(x) = \sum_{i=0}^{\infty} a_i x^i \qquad\qquad f(x) = (\,x^3 + x^4 + \cdots + x^{10})^4$$

۲۰ شی را به چند طریق می توان میان ۴ نفر تقسیم کرد بطوریکه هر نفر حداقل ۳ شی و حداکثر ۱۰ شی ء در اختیار داشته باشد؟ ضریب $x^{20} = ?$

$$A(x) = x^{12}(1 + x + x^2 + \cdots)^4 \longrightarrow \left(\frac{1}{1-x}\right)^4$$

حل روابط بازگشتی

- ۴ـ جانشینی substitution
- ۵ـ درخت بازگشت recursion tree
- ۶ـ قضیه اصلی master theorem

حل روابط بازگشتی

- روش جایگزینی:

۱ـ حدس جواب

۲ـ استفاده از استقرای ریاضی برای اثبات درستی حدس.

این روش موقعی کاربرد دارد که بتوان جواب را به آسانی حدس زد.

حل روابط بازگشتی

روش جایگزینی:

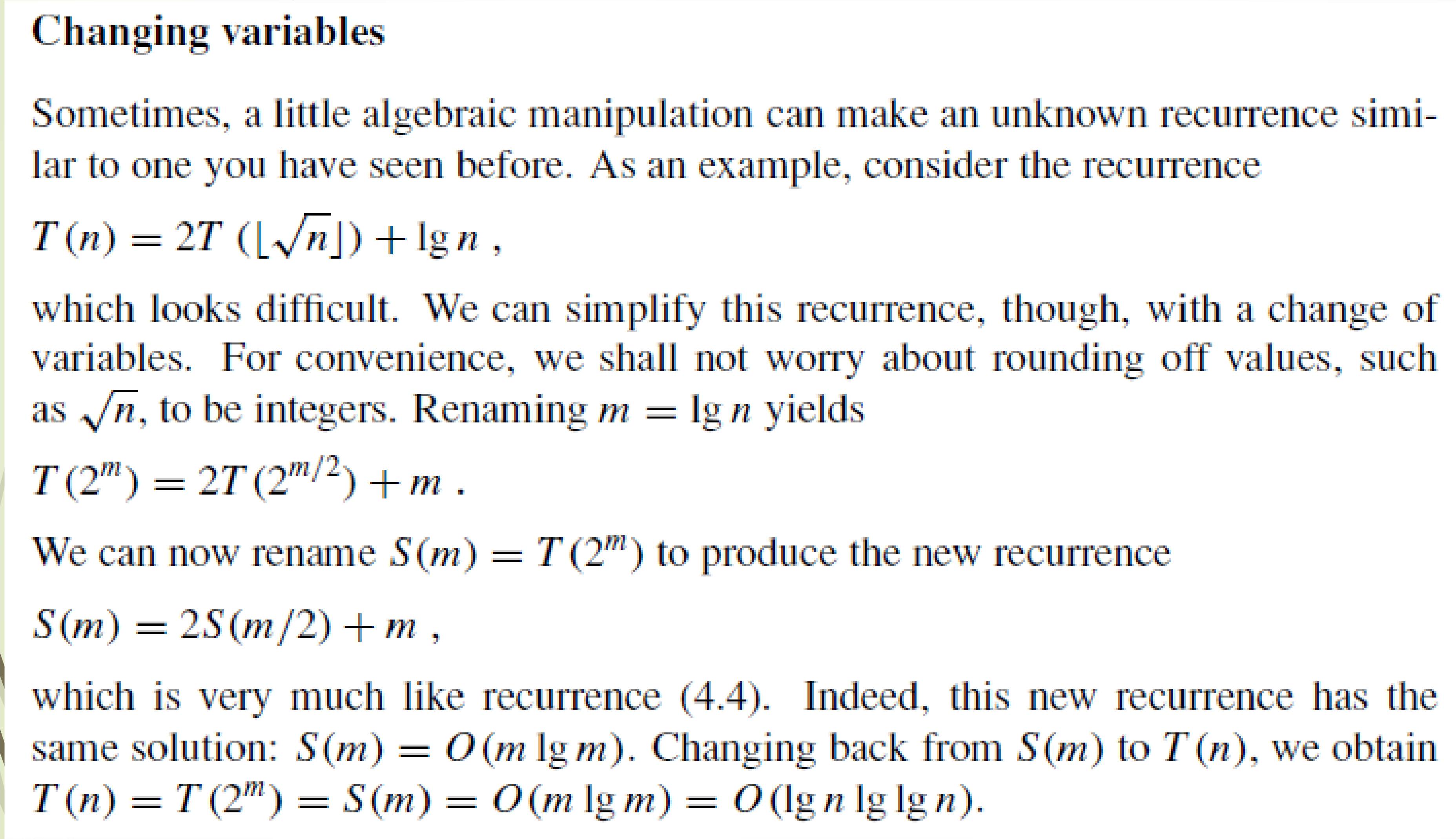
$$T(n) = 2T(\lfloor n/2 \rfloor) + n ,$$

make the guess that $T(n) = O(n \lg n)$, $T(n) \le cn \lg n$

$$\begin{aligned} T(n) &\le 2(c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\ &\le cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\le cn \lg n , \end{aligned}$$

حل روابط بازگشتی

روش جایگزینی:



Changing variables

Sometimes, a little algebraic manipulation can make an unknown recurrence similar to one you have seen before. As an example, consider the recurrence

$$T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n ,$$

which looks difficult. We can simplify this recurrence, though, with a change of variables. For convenience, we shall not worry about rounding off values, such as $\sqrt{n}$, to be integers. Renaming $m = \lg n$ yields

$$T(2^m) = 2T(2^{m/2}) + m .$$

We can now rename $S(m) = T(2^m)$ to produce the new recurrence

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

which is very much like recurrence (4.4). Indeed, this new recurrence has the same solution: $S(m) = O(m \lg m)$. Changing back from $S(m)$ to $T(n)$, we obtain $T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$.

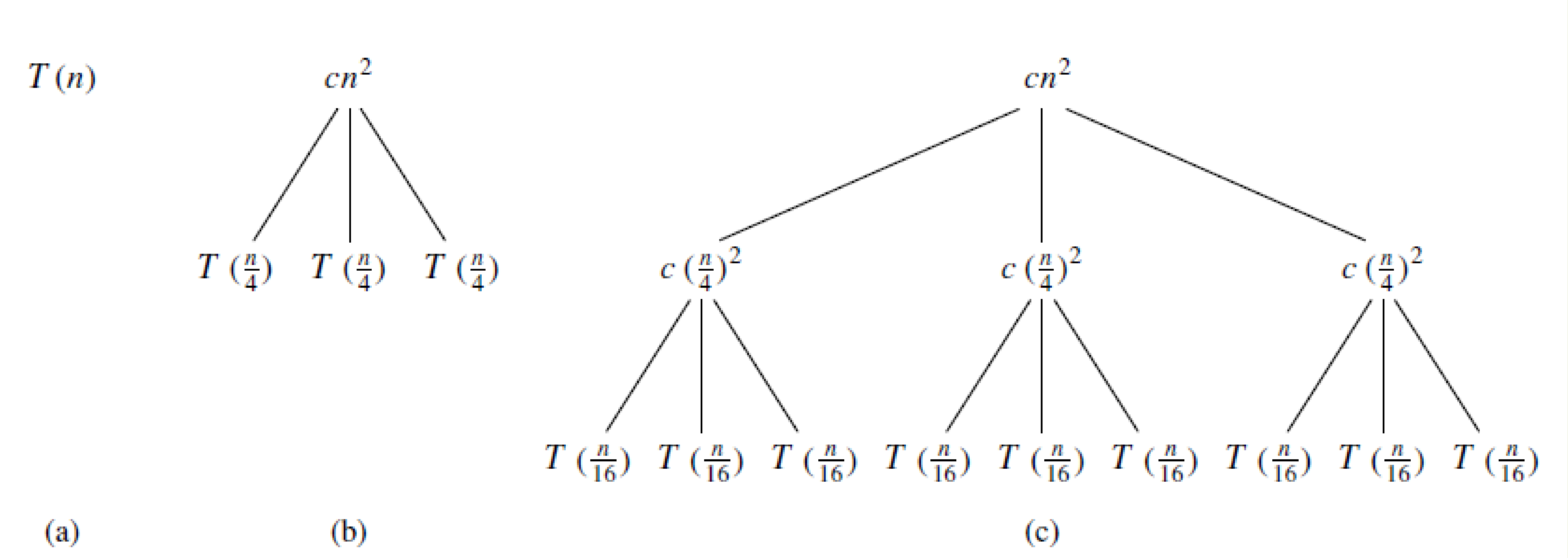
حل روابط بازگشتی

روش جایگزینی:

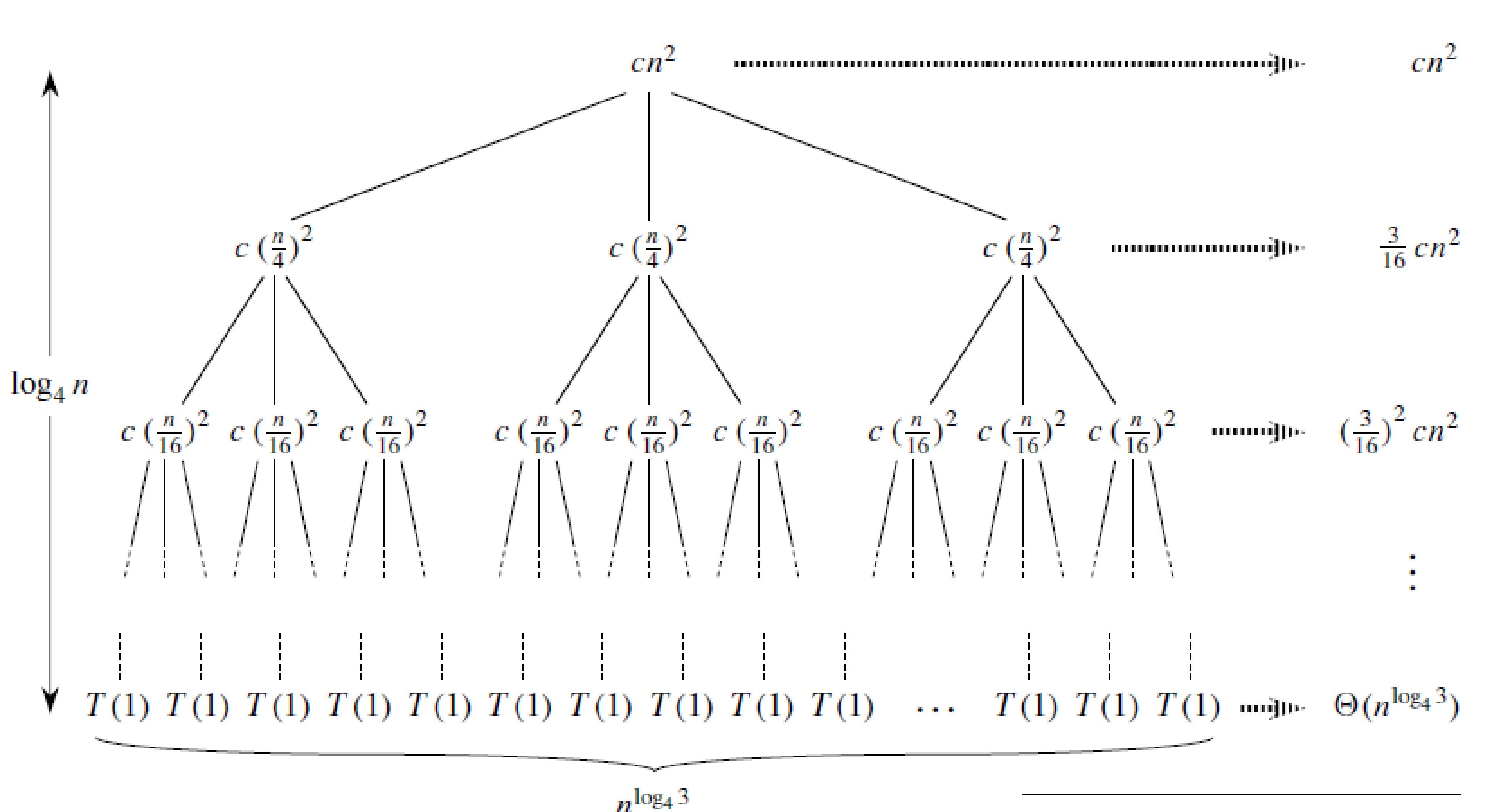
حل روابط بازگشتی

- درخت بازگشت:
- $T(n) = 3\,T\,(n/4) \ + \theta(n^2)$
- $T\,(n) = 3\,T\,(n/4) + Cn^2$

حل روابط بازگشتی



حل روابط بازگشتی

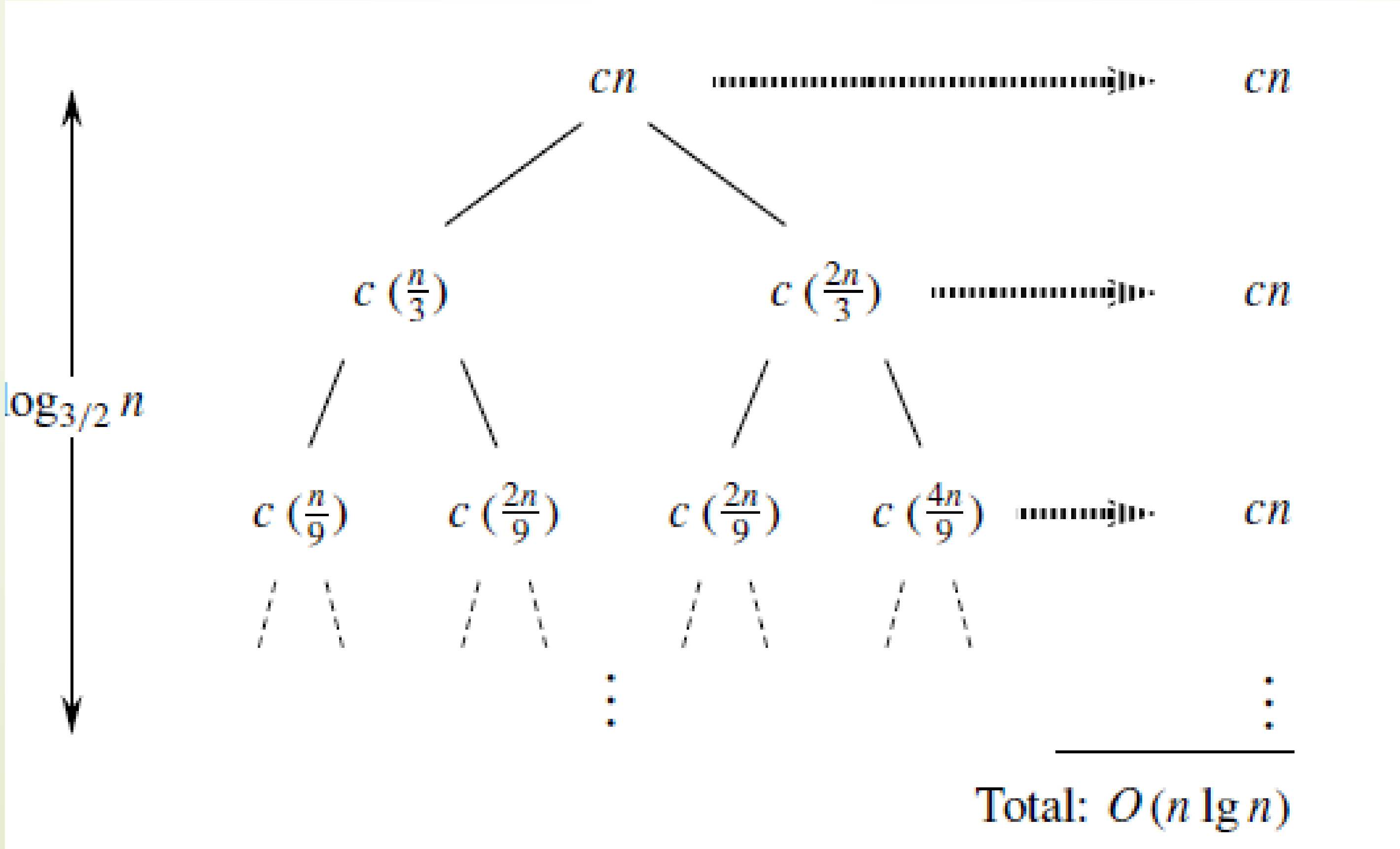


حل روابط بازگشتی

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2) .
\end{aligned}
$$

حل روابط بازگشتی

$$T(n) = T(n/3) + T(2n/3) + O(n) .$$



حل روابط بازگشتی

4.2-5

Use a recursion tree to give an asymptotically tight solution to the recurrence $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$, where α is a constant in the range $0 < \alpha < 1$ and $c > 0$ is also a constant.