

طراحی الگوریتم‌ها – تقسیم و حل

Introduction to Algorithm

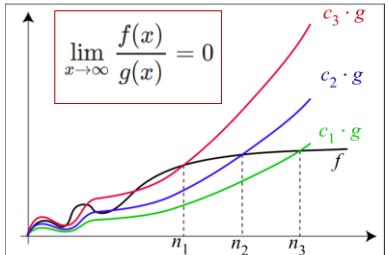
مهندی جوانمردی

۱۴۰۱

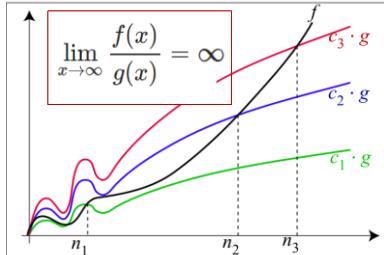
مرور مباحث قبل

نمادهای o و ω کوچک

The o -Notation



The ω -Notation



$$T_A(n) = o(T_B(n)) \quad \leftarrow \text{الgoritم A بهتر است algoritm B است}$$

خواص توابع و نمادهای استاندارد

تکرار تابع

Functional Iteration

$$f^{(i)}(n) = \begin{cases} n & \text{if } i = 0, \\ f(f^{(i-1)}(n)) & \text{if } i > 0. \end{cases}$$

لگاریتم استار

Iterated logarithmic function

$$\lg^* n = \min \{i \geq 0 : \lg^{(i)} n \leq 1\}$$

تراگذری

Transitivity

یکنواهی تابع

Monotonicity

بازتابی

Reflexivity

تقارنی

Symmetry

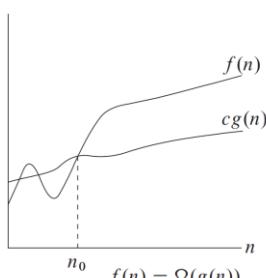
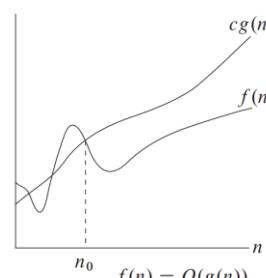
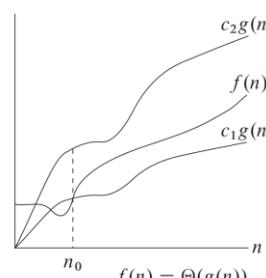
ضد تقارنی

Transpose Symmetry

نمادهای تقریبی رشد توابع

Name	Notation	$f(n) = O(g(n)) \approx a \leq b,$
Big O	O or O	$f(n) = \Omega(g(n)) \approx a \geq b,$
Big Omega	Ω	$f(n) = \Theta(g(n)) \approx a = b,$
Big Theta	Θ	
Small O	o	$f(n) = o(g(n)) \approx a < b,$
Small Omega	ω	$f(n) = \omega(g(n)) \approx a > b.$

رشد توابع و حد بینهایت



$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = C$$

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = C \text{ or } 0$$

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \infty \text{ or } C$$

فصل چهارم: تقسیم و حل | Divide and Conquer

- مسئله زیرآرایه بیشینه یا Maximum Subarray
- انواع روش‌های حل رابطه بازگشت
- جایگذاری و استقرای ریاضی
- درخت بازگشت
- قضیه اصلی
- الگوریتم استراسن برای ضرب ماتریسی

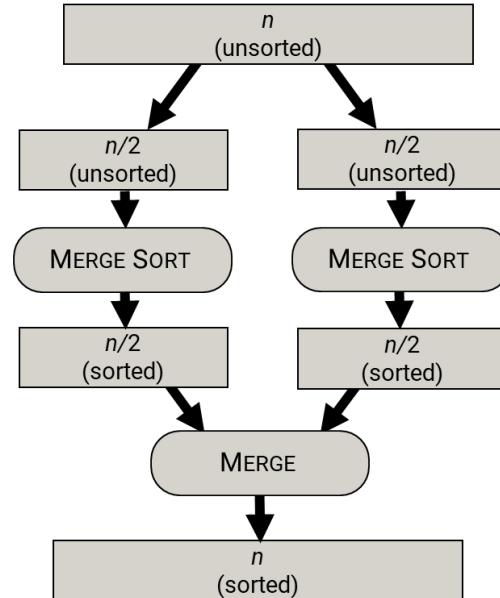
4 Divide-and-Conquer 65

- 4.1 The maximum-subarray problem 68
- 4.2 Strassen's algorithm for matrix multiplication 75
- 4.3 The substitution method for solving recurrences 83
- 4.4 The recursion-tree method for solving recurrences 88
- 4.5 The master method for solving recurrences 93
- ★ 4.6 Proof of the master theorem 97

الگوریتم‌های تقسیم و حل

- **The divide-and-conquer approach:**
 - **Divide** the problem into a number of subproblems (that are smaller instances of the same problem)
 - **Conquer** the subproblems by solving them recursively
 - **Base case:** If the subproblem sizes are small enough, solve it in a straightforward manner
 - **Combine** the solutions to solve the original problem
- **Recursive structure**
 - To solve a given problem, they call themselves recursively one or more times to deal with related subproblems

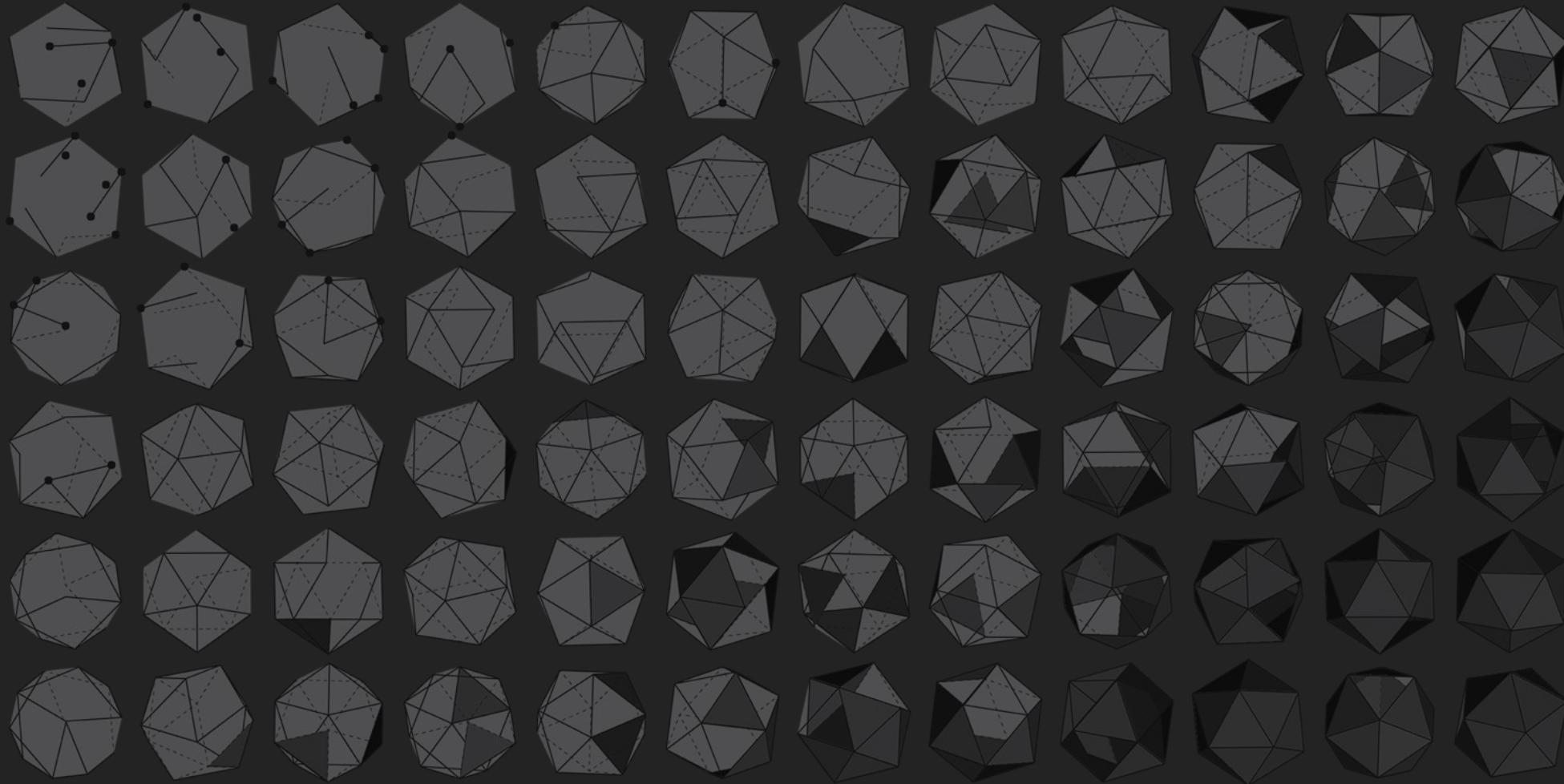
مثال مرتب‌سازی ادغامی



بخشی از مرحله ترکیب

بعضی در الگوریتم‌های حل و تقسیم، در کنار حل زیرمسئله‌های مشابه کوچکتر، لازم است زیرمسئله‌هایی که خیلی تشابه‌ی با مسئله اصلی ندارند را نیز حل کنیم!

نکته:

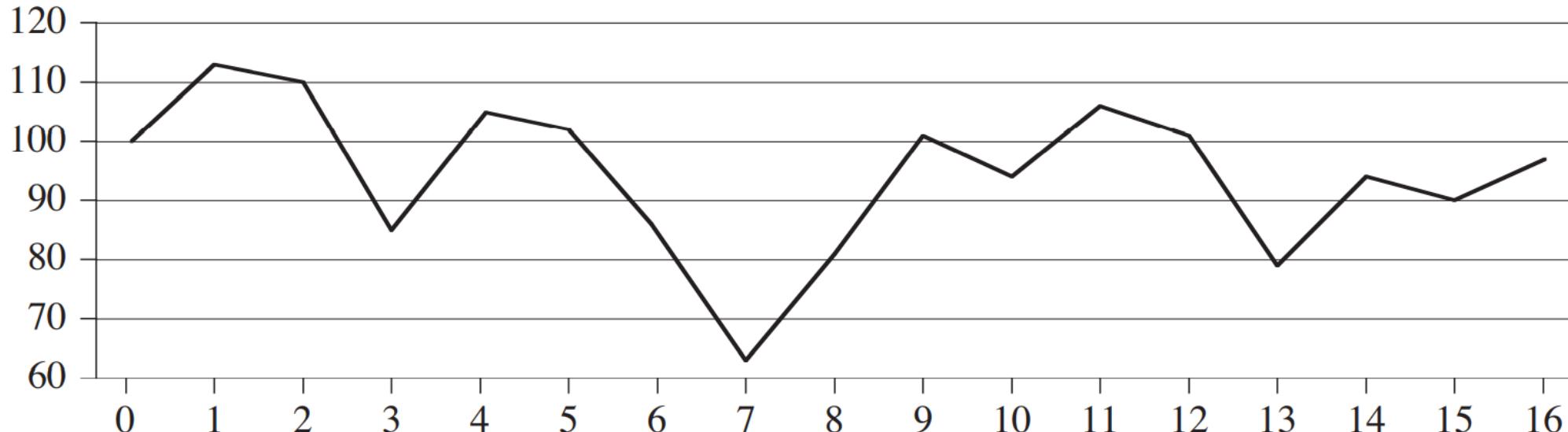


مسئله زیر آرایه پیشینه یا

فصل ۴.۱ کتاب

کسب حد اکثر سود در خرید و فروش سهام

تغییرات سهام شرکت الف در دوره زمانی ۱۷ روزه



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

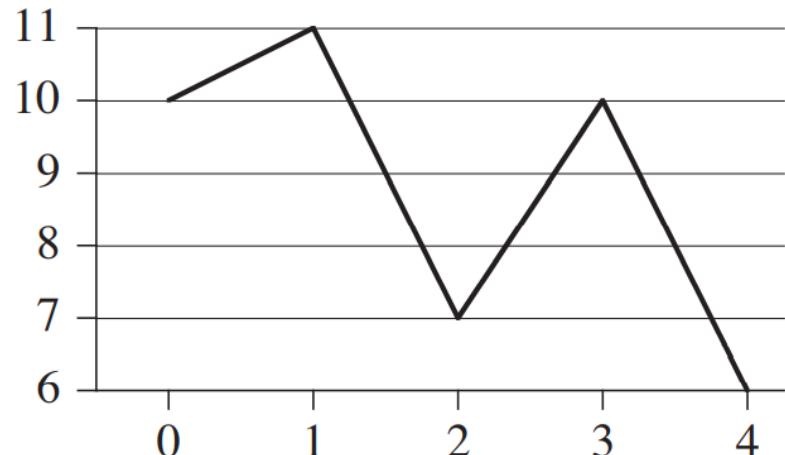
Goal: Maximize your profit



“buy low, sell high” ??

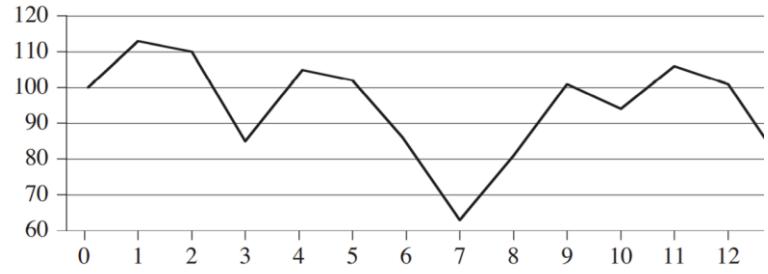
کسب حد اکثر سود در خرید و فروش سهام

- Lowest price might occur *after* the highest price.
- But wouldn't the optimal strategy involve buying at the lowest price *or* selling at the highest price?
- Not necessarily:



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

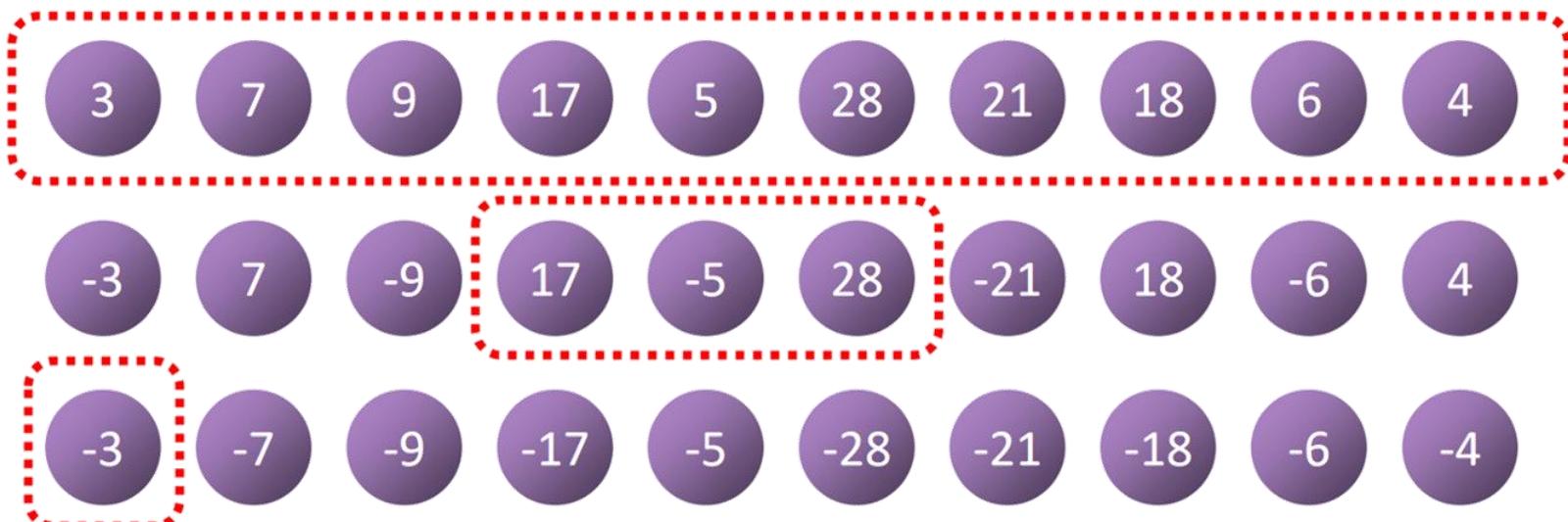
مسئله زیرآرایه بیشینه یا Maximum Subarray



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

- Input: A sequence $A[1], A[2], \dots, A[n]$ of integers.
- Output: Two indices i and j with $1 \leq i \leq j \leq n$ that maximize

$$A[i] + A[i+1] + \dots + A[j].$$



حل زیرآرایه پیشینه با روش تهاجمی

Brute Force Algorithm

تاریخ خرید

ردیف	نام	پیشنهاد	کیا و متوجه																	
۱۸۱۴	۱۵۸۵	۸۸۱	۱۱۰۸	۷۰۵	۷۹۸	۷۷۷	۱۶۰۳	۷۰۰	۷۷۰	۷۷۰	۱۱۸۶	۷۷۲	۷۷۷	۷۷۷	۷۷۷	۷۷۷	۷۷۷	۷۷۷	۷۷۷	ستندج
۹۲۴	۱۲۸۵	۲۸۰	۱۲۷۶	۷۰۰	۱۵۹۱	۱۲۹۱	۴۹۸	۴۰۰	۲۸۰	۱۱۲۲	۱۵۷۸	۱۲۶۵	۱۱۴۴	۷۹۶	۱۳۱۶	۹۷۷	۷۰۵	۹۰۰	۱۱۷۲	شاہرود
۱۲۷۷	۱۰۶۱	۹۲۳	۵۶۴	۹۰۸	۱۰۵۶	۱۰۲۲	۱۰۹۵	۷۹۳	۷۷۷	۷۱۹	۱۳۱۷	۸۰۹	۱۰۲۲	۱۰۴	۱۱۷۸	۱۱۳۴	۷۹۲	۱۰۵۷	۹۷۲	شهرکرد
۱۱۲۵	۶۱۹	۱۳۰۴	۳۰۴	۱۲۸۹	۵۱۵	۱۸۰۳	۱۰۷۶	۱۱۷۷	۱۱۰۲	۱۱۰۰	۱۱۱۲	۹۰۹	۱۰۱۳	۴۸۰	۱۰۰۹	۱۰۱۵	۷۷۲	۱۰۲۸	۵۹۴	شیرواز
۱۵۹۲	۱۵۰۵	۵۳۰	۱۰۵۰	۲۲۵	۱۰۰۷	۷۰۰	۱۰۰۲	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۱۰۰۵	قزوین
۱۹۹۵	۱۱۷۲	۵۱۲	۸۰۵	۴۹۷	۸۹۰	۱۰۱۱	۱۰۰۷	۷۰۱	۷۰۱	۷۰۱	۱۰۲۸	۷۱۵	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	قم
۱۲۹۶	۱۰۴۸	۶۲۶	۷۹۲	۹۱۱	۹۷۲	۱۱۰۰	۱۱۸۸	۹۰۰	۹۰۰	۹۰۰	۱۰۲۸	۹۳۵	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	کاشان
۹۹۹	۴۸۰	۱۵۱۴	۸۰۵	۱۰۰۳	۱۰۰۵	۱۹۱۷	۱۷۹۰	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۰۰۲	۱۰۰۲	۱۰۰۲	۸۸۱	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۱۶۵	کرمان
۱۸۰۰	۱۱۷۹	۹۰۸	۹۷۲	۸۲۰	۸۹۲	۸۸۰	۱۰۷۸	۷۰۰	۷۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۶۱۰	کرمانشاه
۱۰۰۰	۱۰۷۱	۶۷	۱۵۰۵	۸۰۷	۱۰۰۵	۱۰۰۵	۱۰۰۵	۰۰۰	۱۰۰۵	۱۰۰۵	۱۰۰۵	۱۰۰۵	۱۰۰۵	۱۰۰۵	۷۰۰	۷۰۰	۷۰۰	۷۰۰	۷۰۰	گرگان
۱۰۰۸	۳۷۰	۱۵۹۸	۵۹۸	۱۵۰۳	۸۰۹	۲۱۵۷	۲۱۹۰	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۹۰۸	لار
۷۷۰	۱۵۹۳	۸۳۰	۱۵۹۷	۱۱۰۵	۱۰۰۲	۱۰۰۷	۱۰۰۷	۱۰۰۷	۷۰۰	۷۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	لطوف آباد
۷۸۱	۱۰۷۶	۶۳۶	۱۵۹۸	۱۱۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۲۲۲	۷۶۱	۷۶۱	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	مشهد
۱۹۹۱	۱۰۰۵	۱۰۰۷	۱۰۰۸	۷۰۵	۱۰۰۹	۴۷۷	۱۰۰۹	۹۳۶	۹۱۶	۹۲۰	۲۱۶۱	۹۲۲	۰۰۰	۹۲۷	۴۷۵	۹۰۰	۵۱۲	۱۰۴۹	۵۱۲	مهاباد
۵۰۰۴	۸۲۷	۱۰۰۱	۱۰۰۸	۲۰۱۵	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	۱۰۰۹	سپاهاباد
۲۱۱۰	۲۲۳۱	۱۱۷۷	۱۰۰۷	۰۰۹	۱۰۰۸	۳۹۷	۱۰۰۷	۹۲۲	۹۲۲	۹۷۰	۲۲۸۷	۱۲۷۷	۱۱۷	۱۱۷	۱۱۷	۱۱۷	۱۱۷	۱۱۷	۱۱۷	نور دوز
۱۸۲۷	۱۰۲۱	۷۱۷	۱۰۰۹	۴۹۱	۸۱۲	۸۰۹	۱۰۰۹	۵۰۶	۵۰۶	۳۷۷	۱۰۰۷	۹۷۸	۸۰۹	۴۹۶	۵۰۶	۵۰۶	۷۸۱	۵۰۶	۷۸۱	همدان
۱۷۰۵	۷۸۱	۱۱۱۸	۷۰۱	۱۱۰۷	۹۹۲	۱۰۱۷	۱۰۰۷	۹۱۷	۹۱۷	۹۷۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	پاسوج
۸۰۰۷	۸۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۹۷۷	۱۰۰۹	۱۰۰۹	۹۲۶	۹۲۶	۹۰۶	۹۷۸	۹۱۳	۱۰۰۱	۱۰۰۰	۷۰۰	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	پزد

تاریخ نهادن

سود حاصله

حل زیرآرایه بیشینه با روش تهاجمی $O(n^3)$

$O(n^3)$ Brute Force Algorithm

```
MaxSubarray-1(i, j)
```

```
    for i = 1, ..., n
        for j = 1, ..., n
            S[i][j] = -∞
```

 $O(n^2)$

```
    for i = 1, ..., n
        for j = i, i+1, ..., n
            S[i][j] = A[i] + A[i+1] + ... + A[j]
```

 $\left. \right\} O(n^3)$

```
return Champion(S)
```

 $O(n^2)$

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

حل زیرآرایه بیشینه با روش تهاجمی $O(n^2)$

$O(n^2)$ Brute Force Algorithm

```

MaxSubarray-2(i, j)
    for i = 1,...,n
        for j = 1,...,n
            S[i][j] = -∞
            R[0] = 0
            for i = 1,...,n
                R[i] = R[i-1] + A[i]
            for i = 1,...,n
                for j = i+1,i+2,...,n
                    S[i][j] = R[j] - R[i-1]
    return Champion(S)

```

$O(n^2)$

$O(n^2)$

$O(n^2)$

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

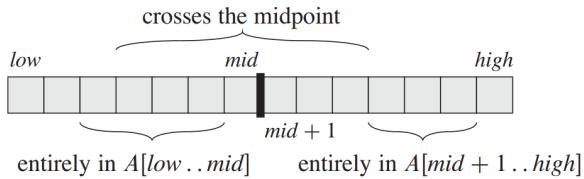
حل زیرآرایه بیشینه با روش تقسیم و حل

آیا روشی با مرتبه زمانی بهتر برای این مسئله می‌توان پیدا کرد؟

نکته: روش تهاجمی قادر به کشف همه MSA بود در حالیکه ما معمولاً نیاز به صرفاً یکی از MSA‌ها داریم و نه همه آن‌ها

روش تقسیم حل برای MSA

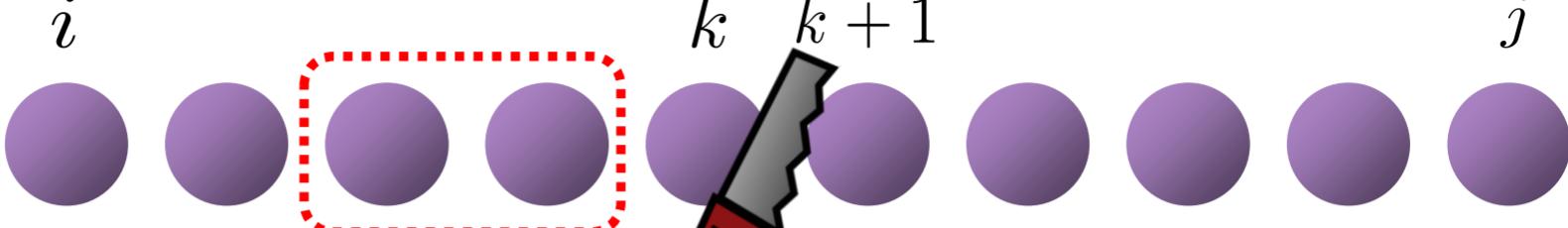
- Base case ($n = 1$)
 - Return itself (maximum subarray)
- Recursive case ($n > 1$)
 - Divide the array into two sub-arrays
 - Find the maximum sub-array recursively
 - Merge the results **How?**



جواب مسئله کجاست؟

- The maximum subarray for any input must be in one of following cases:

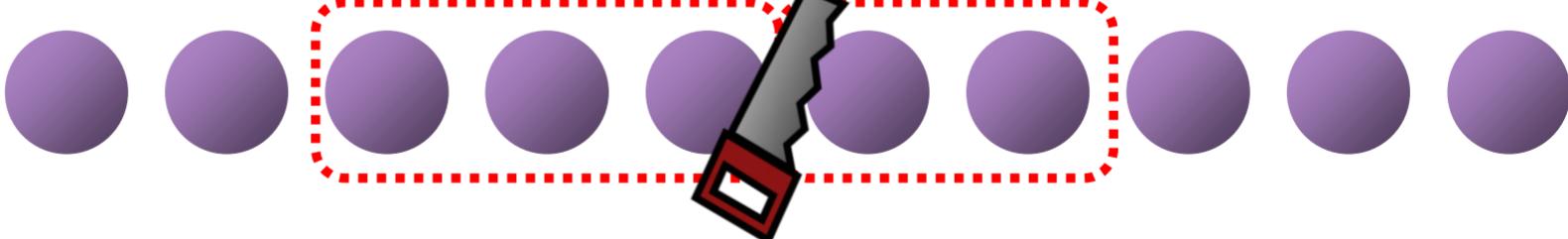
Case 1: left



Case 2: right



Case 3: cross the middle



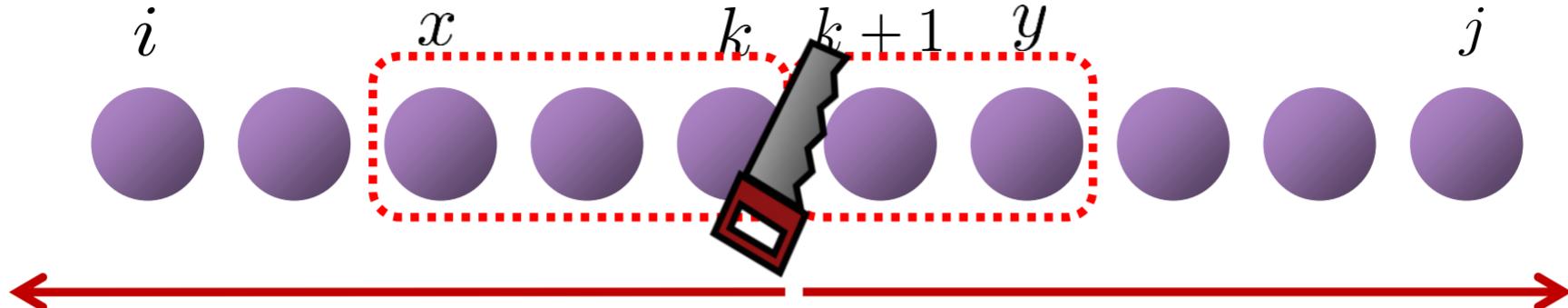
Case 1: $\text{MaxSub}(A, i, j) = \text{MaxSub}(A, i, k)$

Case 2: $\text{MaxSub}(A, i, j) = \text{MaxSub}(A, k+1, j)$

Case 3: $\text{MaxSub}(A, i, j)$ cannot be expressed using MaxSub!

حالت سوم: آرایه‌ای که از وسط عبور می‌کند!

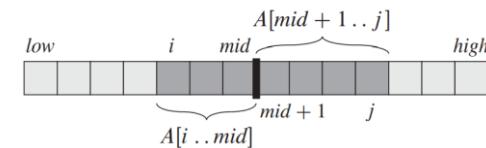
- Goal: find the maximum subarray that crosses the middle



(1) Start from the middle to find the left maximum subarray

(2) Start from the middle to find the right maximum subarray

The solution of Case 3 is the combination of (1) and (2)



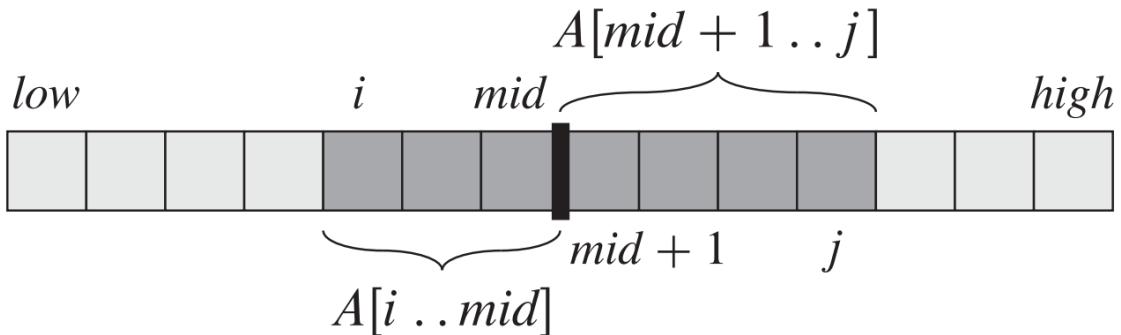
- Observation
 - The sum of $A[x \dots k]$ must be the maximum among $A[i \dots k]$ (left: $i \leq k$)
 - The sum of $A[k + 1 \dots y]$ must be the maximum among $A[k + 1 \dots j]$ (right: $j > k$)
 - Solvable in linear time $\rightarrow \Theta(n)$

حل حالت سوم در زمان خطی

FIND-MAX-CROSSING-SUBARRAY($A, low, mid, high$)

```

1   left-sum = -∞
2   sum = 0
3   for  $i = mid$  downto  $low$ 
4       sum = sum +  $A[i]$ 
5       if sum > left-sum
6           left-sum = sum
7           max-left = i
8   right-sum = -∞
9   sum = 0
10  for  $j = mid + 1$  to  $high$ 
11      sum = sum +  $A[j]$ 
12      if sum > right-sum
13          right-sum = sum
14          max-right = j
15  return (max-left, max-right, left-sum + right-sum)
    
```



$\Theta(n)$

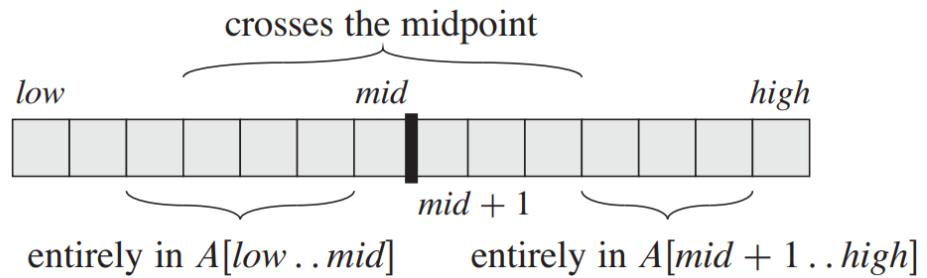
حل زیرآرایه بیشینه با روش تقسیم و حل

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

```

1  if  $high == low$ 
2    return ( $low, high, A[low]$ )           // base case: only one element
3  else  $mid = \lfloor (low + high)/2 \rfloor$ 
4    ( $left-low, left-high, left-sum$ ) =
      FIND-MAXIMUM-SUBARRAY( $A, low, mid$ )
5    ( $right-low, right-high, right-sum$ ) =
      FIND-MAXIMUM-SUBARRAY( $A, mid + 1, high$ )
6    ( $cross-low, cross-high, cross-sum$ ) =
      FIND-MAX-CROSSING-SUBARRAY( $A, low, mid, high$ )
7    if  $left-sum \geq right-sum$  and  $left-sum \geq cross-sum$ 
8      return ( $left-low, left-high, left-sum$ )
9    elseif  $right-sum \geq left-sum$  and  $right-sum \geq cross-sum$ 
10   return ( $right-low, right-high, right-sum$ )
11   else return ( $cross-low, cross-high, cross-sum$ )

```



FIND-MAXIMUM-SUBARRAY($A, 1, A.length$)

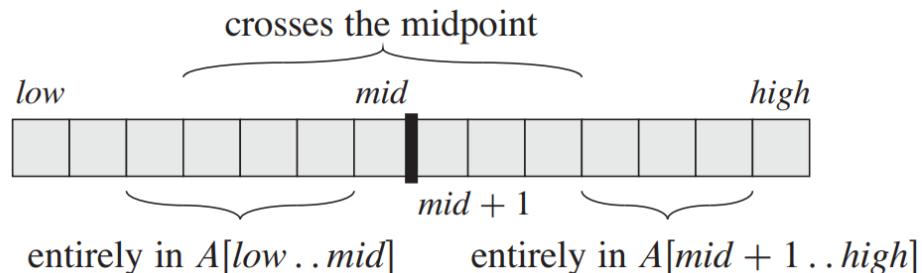
حل زیرآرایه بیشینه با روش تقسیم و حل

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

```

1  if  $high == low$ 
2      return ( $low, high, A[low]$ )           // base case: only one element
3  else  $mid = \lfloor (low + high)/2 \rfloor$ 
Divide   ( $left-low, left-high, left-sum$ ) =
        FIND-MAXIMUM-SUBARRAY( $A, low, mid$ ) Conquer
5    ( $right-low, right-high, right-sum$ ) =
        FIND-MAXIMUM-SUBARRAY( $A, mid + 1, high$ )
6    ( $cross-low, cross-high, cross-sum$ ) =
        FIND-MAX-CROSSING-SUBARRAY( $A, low, mid, high$ )
7  if  $left-sum \geq right-sum$  and  $left-sum \geq cross-sum$ 
8      return ( $left-low, left-high, left-sum$ )
9  elseif  $right-sum \geq left-sum$  and  $right-sum \geq cross-sum$ 
10     return ( $right-low, right-high, right-sum$ )
11  else return ( $cross-low, cross-high, cross-sum$ )

```



Combine

FIND-MAXIMUM-SUBARRAY($A, 1, A.length$)

تحلیل زمانی روش تقسیم و حل MSA

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

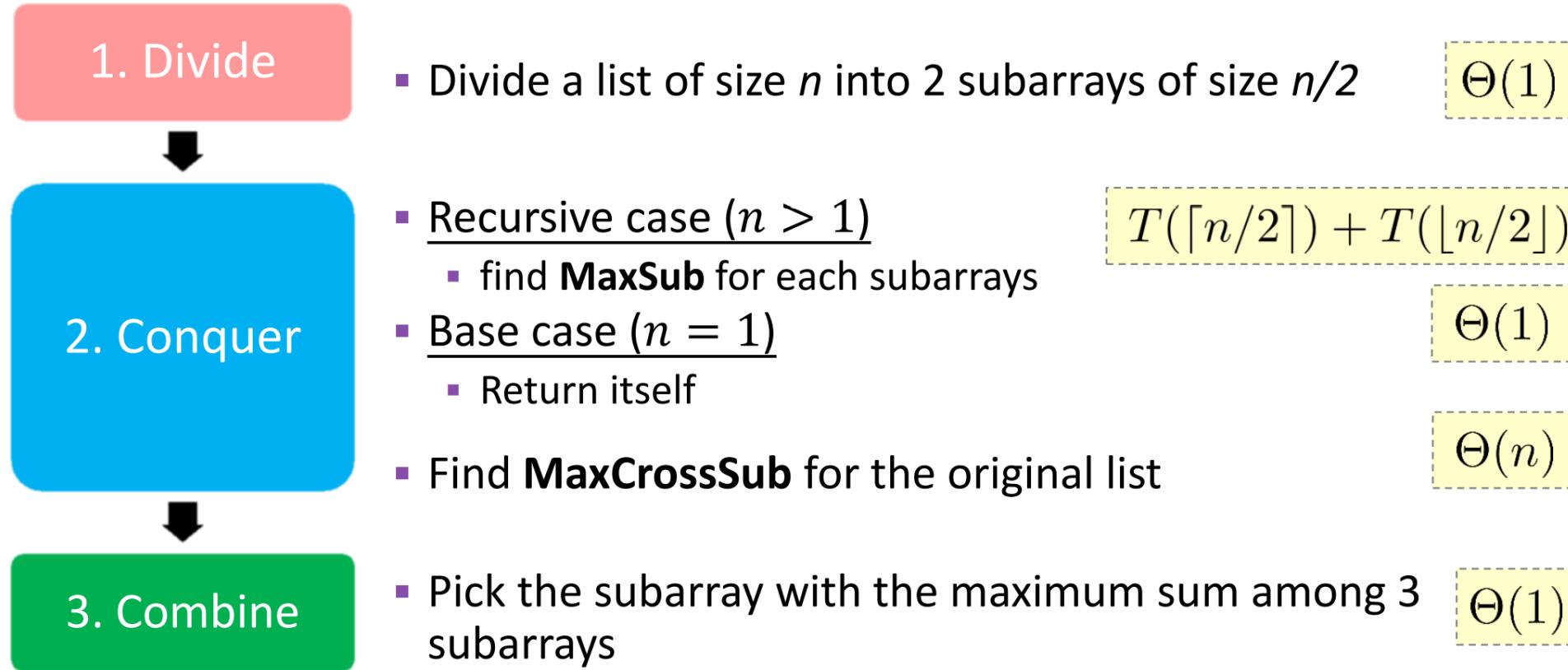
```

1  if  $high == low$                                       $O(1)$ 
2    return ( $low, high, A[low]$ )           // base case: only one element
3  else  $mid = \lfloor (low + high)/2 \rfloor$ 
4    ( $left-low, left-high, left-sum$ ) =  $T(mid - low)$ 
5      FIND-MAXIMUM-SUBARRAY( $A, low, mid$ )
6    ( $right-low, right-high, right-sum$ ) =  $T(high - (mid + 1))$ 
7      FIND-MAXIMUM-SUBARRAY( $A, mid + 1, high$ )
8    ( $cross-low, cross-high, cross-sum$ ) =  $O(high - mid)$ 
9      FIND-MAX-CROSSING-SUBARRAY( $A, low, mid, high$ )
10     if  $left-sum \geq right-sum$  and  $left-sum \geq cross-sum$   $O(1)$ 
11       return ( $left-low, left-high, left-sum$ )
12     elseif  $right-sum \geq left-sum$  and  $right-sum \geq cross-sum$   $O(1)$ 
13       return ( $right-low, right-high, right-sum$ )
14     else return ( $cross-low, cross-high, cross-sum$ )  $O(1)$ 

```

FIND-MAXIMUM-SUBARRAY($A, 1, A.length$)

تحلیل زمانی روش تقسیم و حل MSA



- $T(n) = \text{time for running } \text{MaxSubarray}(\mathbb{A}, i, j) \text{ with } j - i + 1 = n$

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + O(n) & \text{if } n \geq 2 \end{cases}$$

تحلیل زمانی روش تقسیم و حل

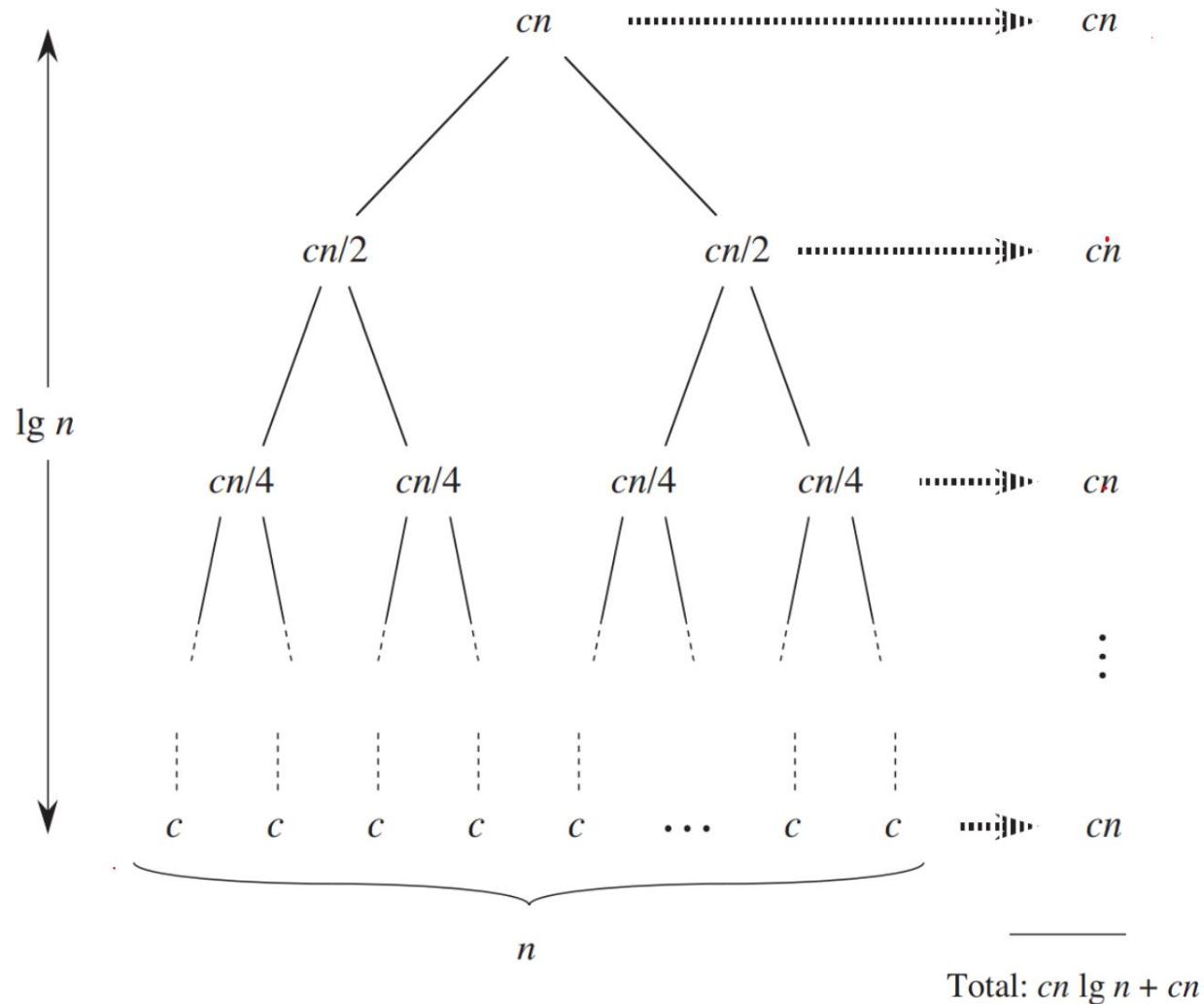
$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + O(n) & \text{if } n \geq 2 \end{cases}$$

ساده‌سازی: تعداد ورودی‌ها توانی از ۲

تعداد ورودی زیرمسئله‌ها
همیشه عدد صحیح

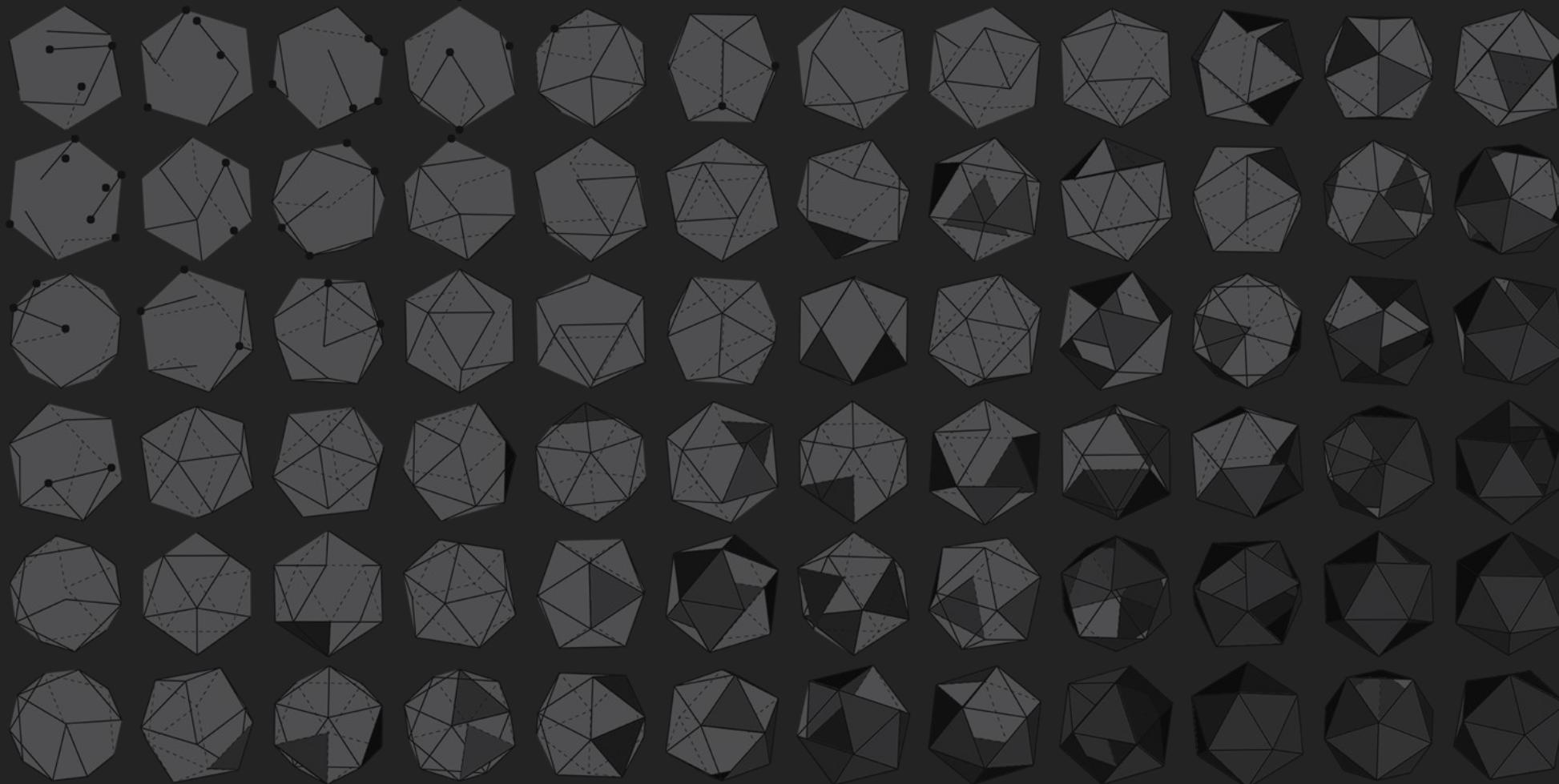
Merge Sort

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



Total: $cn \lg n + cn$

آیا این سریع‌ترین راه حل است؟؟ جواب: نه، به تمرین ۵-۱۴ مراجعه شود



روش‌های حل روابط بازگشتی

فصل ۴.۳ الی ۴.۶ کتاب

مرواری بر رابطه بازگشت

Recurrences go hand in hand with the divide-and-conquer paradigm, because they give us a natural way to characterize the running times of divide-and-conquer algorithms.

- A **recurrence** is an equation or inequality that describes a function in terms of its value on smaller inputs.
- Example from MERGE-SORT or MAXIMUM-SUBARRAY

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

$$T(n) = T(2n/3) + T(n/3) + \Theta(n)$$

$$T(n) = T(n - 1) + \Theta(1)$$

روش‌های حل رابطه بازگشت

- جایگذاری و استقرار ریاضی
- درخت بازگشت
- قضیه اصلی Master Theorem

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

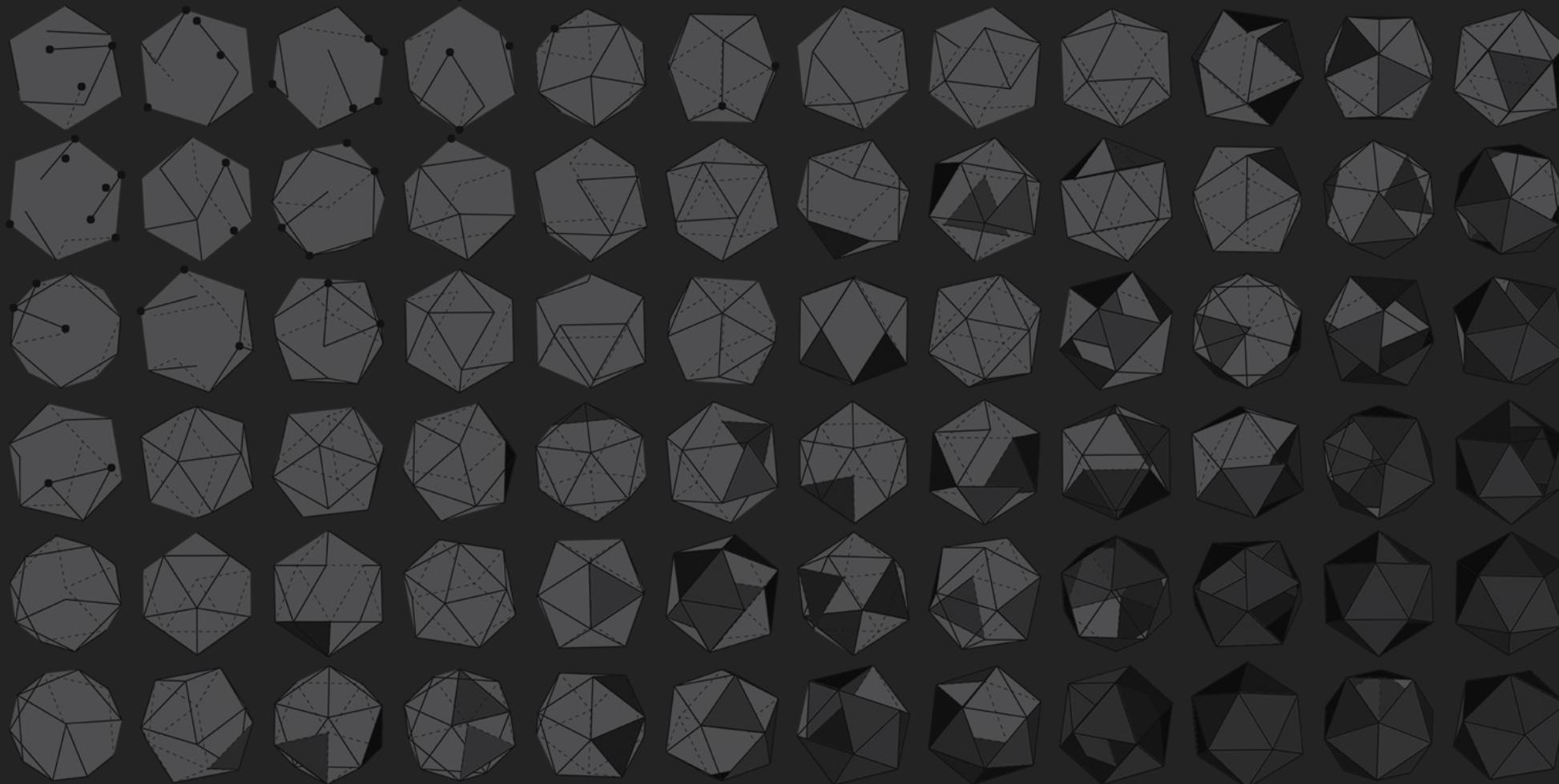
جزیيات فنی حل رابطه بازگشت

- کف یا جزء صحیح
- سقف
- حالت‌های مرزی

در حل رابطه بازگشتی معمولاً از جزیياتی مانند کف، سقف و حالت‌های مرزی چشم‌پوشی می‌کنیم زیرا معمولاً تاثیرگذار در نتیجه نیستند!



ولی باید بدانیم در چه شرایطی آن‌های موثر هستند و باید در نظر گرفته شوند



روش‌های حل روابط بازگشتی: جایگذاری

فصل ۴.۳ کتاب

روش جایگذاری

- Involves two steps:
 - Guess the form of the solution
 - Use mathematical induction to find the constants and show the solution works
- Drawback: applied only in cases where it is easy to guess at solution



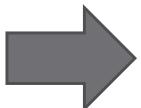
حدس از کجا؟ میتوانیم از روش درخت بازگشت بهره بگیریم

- Example: MERGE-SORT or MAXIMUM-SUBARRAY

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

- Guess:

$$T(n) = O(n \lg n)$$



Prove by induction:

$$T(n) \leq dn \lg n$$

for suitable $d > 0$.

اثبات از طریق استقراء در روش جایگذاری

$$T(1) = \Theta(1) \leq dn \lg n$$

- در این مثال پایه استقراء با فرض $n = 1$ برقرار است؟

$$T(n) \leq cn \lg n \quad \rightarrow \quad T(1) \leq c1 \lg 1 = 0 \quad \times$$

- طبق تعریف ما نیاز به n_0 دلخواه داریم که برای $n > n_0$ ثابت کنیم که $T(n) \leq dn \lg n$
- از طرفی برای $n > 3$ ما مستقیماً وابسته به $T(1)$ نیستم. پس بجای $T(1)$ می‌توانیم $T(2)$ و $T(3)$ را جایگزین کنیم به عنوان پایه استقراء و در نظر بگیریم که $n_0 = 0$

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

$$(2T(1)+2c)/2 \leq d$$

$$(2T(1)+3c)/3\lg 3 \leq d$$

- | | | |
|---------------------|---------------------|----------------|
| $T(2) \leq 2 \lg 2$ | $T(2) = 2T(1) + 2c$ | • پایه استقراء |
| $T(3) \leq 3 \lg 3$ | $T(3) = 2T(1) + 3c$ | • پایه استقراء |

اثبات از طریق استقراء در روش جایگذاری

- پایه استقراء به شرط $T(1) + c \leq d$ برقرار است

$$T(n) \leq dn \lg n$$

- فرض استقراء: برای $n < k$ خواهیم داشت:

- حکم استقراء: برای $n = k$ نیز نابرابری فوق برقرار خواهد بود

- فرض استقراء: $T(k/2) \leq d \cdot k/2 \cdot \lg(k/2)$
- حکم استقراء: $T(k) \leq d \cdot k \cdot \lg(k)$

- مثال رابطه بازگشتی

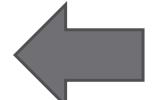
$$T(n) = 2T(n/2) + cn$$

- حدس

$$T(n) = O(n \lg n)$$

- در مثال اخیر

$$\begin{aligned} T(k) &= 2T(k/2) + ck \leq 2d \cdot k/2 \cdot \lg(k/2) + ck \\ &\leq d \cdot k \cdot \lg(k/2) + ck \\ &\leq d \cdot k \cdot [\lg(k) - \lg(2)] + ck \\ &\leq d \cdot k \cdot \lg(k) + ck - dk \end{aligned}$$



دسته‌ای از d که رابطه را صحیح می‌کند

$$d \geq c$$

مثال روش جایگذاری

- مثال دیگر...

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

$c > 0$ برای یک $T(n) \leq cn \lg n$ باید اثبات شود که $T(n) = O(n \lg n)$ حدس:

فرض: حکم قضیه برای $\underline{T(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)}$ صادق است. در نتیجه خواهیم داشت: $m = \lfloor n/2 \rfloor$

جایگذاری در رابطه بازگشت

$$\begin{aligned}
 T(n) &\leq 2(c \lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\
 &\leq cn \lg(n/2) + n \\
 &= cn \lg n - cn \lg 2 + n \\
 &= cn \lg n - cn + n \xrightarrow{-cn + n \leq 0} c \geq 1 \\
 &\leq cn \lg n ,
 \end{aligned}$$

چگونه حدس خوبی بزنیم؟

- روش عمومی برای یک حدس خوب وجود ندارد!
- تجربه! خلاقیت! شهود! و شاید هم درخت بازگشتی!
- اگر مشابه رابطه بازگشت را قبل دیده اید، راه حل مشابه میتواند منطقی باشد!

$$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$$

← تفاوت چندانی نخواهد داشت!

$$T(n) = O(n \lg n) \longrightarrow \text{Exercise 4.3-6}$$

- در برخی موارد مرتبه را صحیح حدس میزنیم ولی استقراء بنوعی درست کار نمیکند!!

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

$$T(n) \leq c \lfloor n/2 \rfloor + c \lceil n/2 \rceil + 1$$

$$T(n) = O(n) \longrightarrow T(n) \leq cn$$

$$= cn + 1, \times$$

تصحیح حدس زده شده

- در برخی موارد مرتبه را صحیح حدس میزنیم ولی استقراره بنوعی درست کار نمیکند!!

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

حدس $T(n) = O(n) \rightarrow T(n) \leq cn \times \rightarrow T(n) \leq cn - d$, where $d \geq 0$ is a constant.

$$\begin{aligned} T(n) &\leq (c \lfloor n/2 \rfloor - d) + (c \lceil n/2 \rceil - d) + 1 \\ &= cn - 2d + 1 \\ &\leq cn - d, \end{aligned}$$

$$d \geq 1$$

- درجه جمله اضافه کمتر از حکم باشد: از حکم یک جمله با درجه کمتر کم میکنیم
- درجه جمله اضافه با حکم برابر باشد: یک فاکتور لگاریتم در حکم کمتر حدس زده ایم
- درجه جمله اضافه بیشتر از حکم باشد: باید حکم از درجه بالاتری باشد

تصحیح حدس زده شده

۱. درجه جمله اضافه کمتر از حکم باشد: به حکم یک جمله از درجه کمتر اضافه میکنیم
۲. درجه جمله اضافه با حکم برابر باشد: یک فاکتور لگاریتم در حکم کمتر حدس زدیم
۳. درجه جمله اضافه بیشتر از حکم باشد: باید حکم از درجه بالاتری باشد

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

حدس

$$T(n) = O(n) \longrightarrow T(n) \leq cn$$

$$\begin{aligned} T(n) &\leq c \lfloor n/2 \rfloor + c \lceil n/2 \rceil + n \\ &= cn + n, \end{aligned}$$

حالت دوم، درجه جمله اضافی برابر حکم است. پس یک لگاریتم کم حدس زده ایم!

خطاهای احتمالی در استقراء

- خطأ کردن با عدم دقت در استقراء خیلی هم سخت نیست!

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

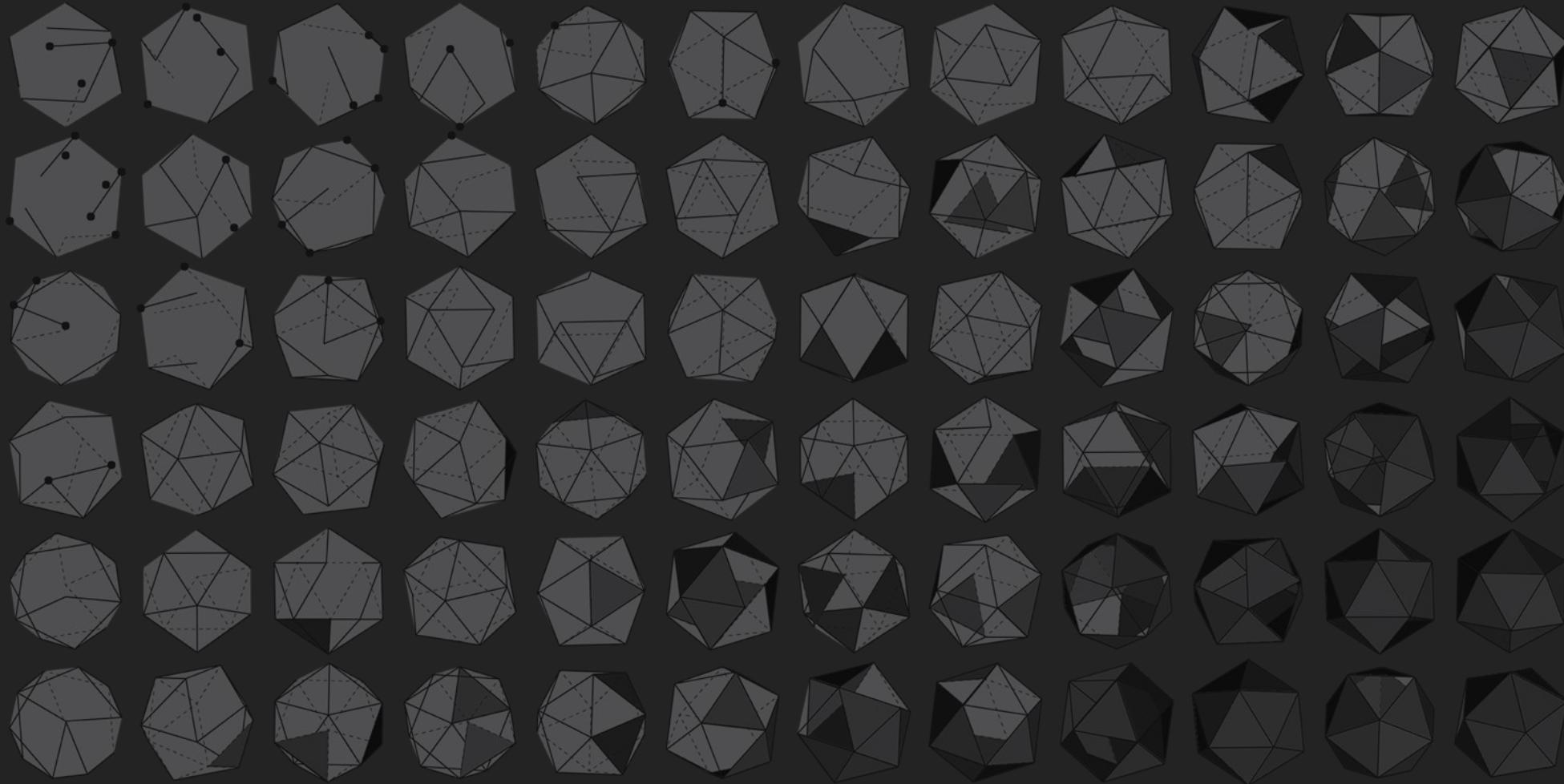
حدس	$T(n) = O(n) \longrightarrow T(n) \leq cn \quad \text{X}$
-----	---

$$\begin{aligned} T(n) &\leq 2(c \lfloor n/2 \rfloor) + n \\ &\leq cn + n \\ &= O(n), \quad \leftarrow \text{wrong!!} \end{aligned}$$

تغییر متغیر برای حل استقراء

- در برخی موارد یک تغییر متغیر ساده شما را به یک رابطه بازگشتی آشنا می‌سازد

$$\left. \begin{array}{l} T(n) = 2T(\sqrt{n}) + \lg n \\ m = \lg n \end{array} \right\} \quad \left. \begin{array}{l} T(2^m) = 2T(2^{m/2}) + m \\ S(m) = T(2^m) \end{array} \right\} \quad \left. \begin{array}{l} S(m) = 2S(m/2) + m \\ S(m) = O(m \lg m) \end{array} \right\} \quad \downarrow \quad \boxed{T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)}$$



روش‌های حل رابطه بازگشتی: قضیه اصلی Master Theorem

فصل ۵.۴ کتاب

قضیه اصلی Mater Theorem

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

رابطه بازگشت $T(n)$ مسئله را به a زیرمسئله مساوی به اندازه n/b تقسیم میکند و هزینه تقسیم و ترکیب زیرمسئله‌ها برابر $f(n)$ است

n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$

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

$$f(n) \quad \square \quad n^{\log_b a}$$

مثال قضیه اصلی - ۱

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

$$f(n) \quad \square \quad n^{\log_b a}$$

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

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

$$n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$$

$$f(n) = O(n^{\log_3 9 - \epsilon}), \text{ where } \epsilon = 1$$

$$T(n) = \Theta(n^2)$$

مثال قضیه اصلی - ۲

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

$$f(n) \quad \square \quad n^{\log_b a}$$

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

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

$$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$$

$$f(n) = \Theta(n^{\log_b a}) = \Theta(1)$$

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

مثال قضیه اصلی - ۳

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

$$f(n) \quad \square \quad n^{\log_b a}$$

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

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

$$n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$$

$$f(n) = \Omega(n^{\log_4 3 + \epsilon}), \text{ where } \epsilon \approx 0.2$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n) \text{ for } c = 3/4$$

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

مثال قضیه اصلی - ۴

$$T(n) = 2T(n/2) + n \lg n$$

$$n^{\log_b a} = n \quad \boxed{\quad} \quad f(n) = n \lg n$$

X

is not *polynomially* larger

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

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

The ratio $f(n)/n^{\log_b a} = (n \lg n)/n = \lg n$ is asymptotically less than n^ϵ for any positive constant ϵ

the recurrence falls into the gap between case 2 and case 3

Exercise 4.6-2

Show that if $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$, then the master recurrence has solution $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$. For simplicity, confine your analysis to exact powers of b .

مثال قضیه اصلی - ۵

$$T(n) = 2T(n/2) + \Theta(n)$$

مرتبه زمانی الگوریتم‌های تقسیم و حل MSA و Mergesort

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

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

$$n^{\log_b a} = n^{\log_2 2} = n$$

$$f(n) = \Theta(n)$$

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

مثال قضیه اصلی - ۶

$$T(n) = 2T(n/4) + \sqrt{n}$$

$$f(n) \quad \boxed{n^{\log_b a}}$$

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

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

$$n^{\log_b a} = n^{\log_4 2} = \sqrt{n}$$

$$\sqrt{n} = \Theta(n^{\log_4 2}) \quad \boxed{\text{case 2}}$$

$$T(n) = \Theta(\sqrt{n} \lg n)$$

مثال قضیه اصلی - ۷

$$T(n) = T(\sqrt{n}) + 1$$

استفاده از تغییر متغیر

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

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

$$m = \lg n$$

$$T(2^m) = T(2^{m/2}) + 1$$

$$S(m) = T(2^m)$$

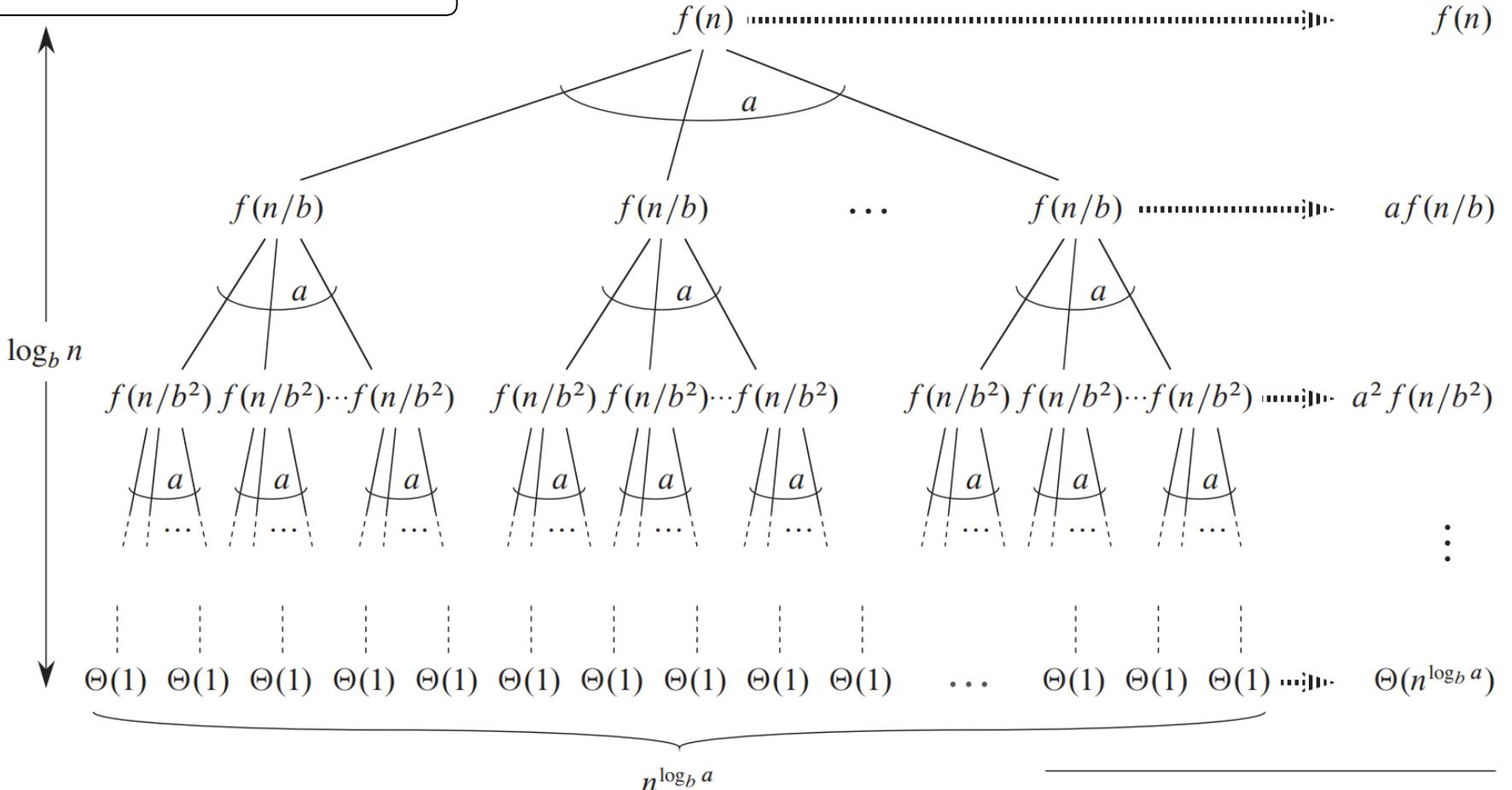
$$S(m) = S(m/2) + 1$$

$$\left\{ \begin{array}{l} n^{\log_b a} = n^{\log_2 1} = n^0 = 1 \quad f(n) = 1 \\ f(n) = \Theta(1) \\ \text{case 2 applies and } S(m) = \Theta(\lg m) \\ T(n) = \Theta(\lg \lg n) \end{array} \right.$$

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

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

درک قضیه اصلی



مثال قضیه اصلی - ۴

Exercise 4.6-2

Show that if $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$, then the master recurrence has solution $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$. For simplicity, confine your analysis to exact powers of b .

$$\begin{aligned}
 T(n) = aT(n/b) + f(n) &\longrightarrow T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n-1} a^j f(n/b^j) \\
 f(n) = \Theta(n^{\log_b a} \lg^k n) &\longrightarrow T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n-1} a^j \Theta\left((n/b^j)^{\log_b a} \lg^k(n/b^j)\right) \\
 &= \Theta(n^{\log_b a}) + \Theta\left(\sum_{j=0}^{\log_b n-1} a^j (n/b^j)^{\log_b a} \lg^k(n/b^j)\right) \\
 &= \Theta(n^{\log_b a}) + \Theta\left(n^{\log_b a} \sum_{j=0}^{\log_b n-1} \left(\frac{a}{b^{\log_b a}}\right)^j \lg^k(n/b^j)\right) \\
 &= \Theta(n^{\log_b a}) + \Theta\left(n^{\log_b a} \sum_{j=0}^{\log_b n-1} \lg^k(n/b^j)\right)
 \end{aligned}$$

مثال قضیه اصلی - ۴

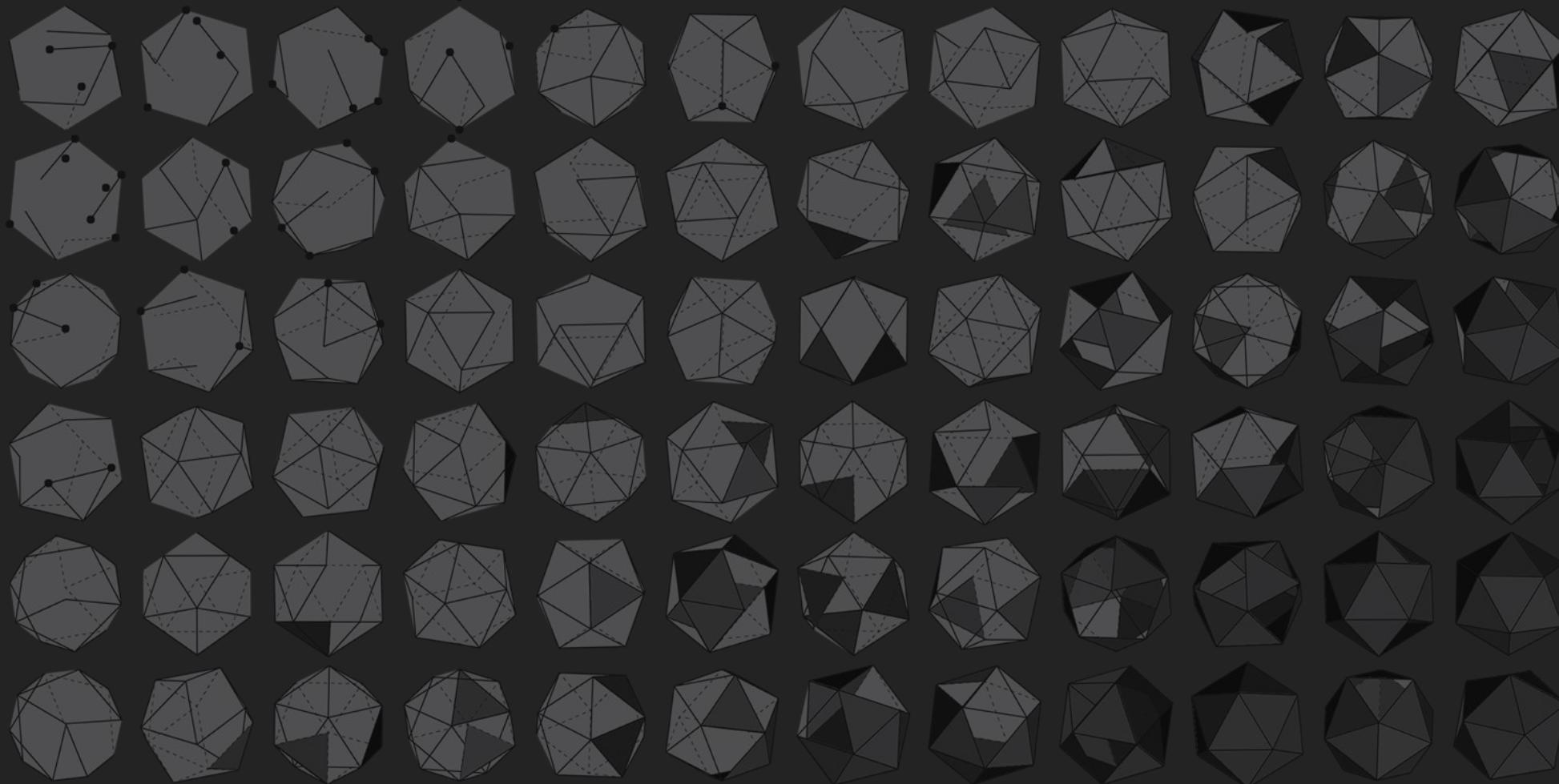
$$= \Theta(n^{\log_b a}) + \Theta\left(n^{\log_b a} \sum_{j=0}^{\log_b n - 1} \lg^k(n/b^j)\right)$$



$$\begin{aligned} \lg^k(n/b^j) &= (\lg n - \lg b^j)^k \\ &= \lg^k n - \lg^k b^j \end{aligned}$$



$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + \Theta\left(n^{\log_b a} \sum_{j=0}^{\log_b n - 1} \lg^k(n/b^j)\right) \\ &= \Theta(n^{\log_b a}) + \Theta\left(n^{\log_b a} \sum_{j=0}^{\log_b n - 1} \lg^k n\right) \\ &= \Theta(n^{\log_b a}) + \Theta(n^{\log_b a} \log_b n \lg^k n) \\ &= \Theta(n^{\log_b a}) + \Theta(n^{\log_b a} \lg^{k+1} n) \\ &= \Theta(n^{\log_b a} \lg^{k+1} n) \end{aligned}$$

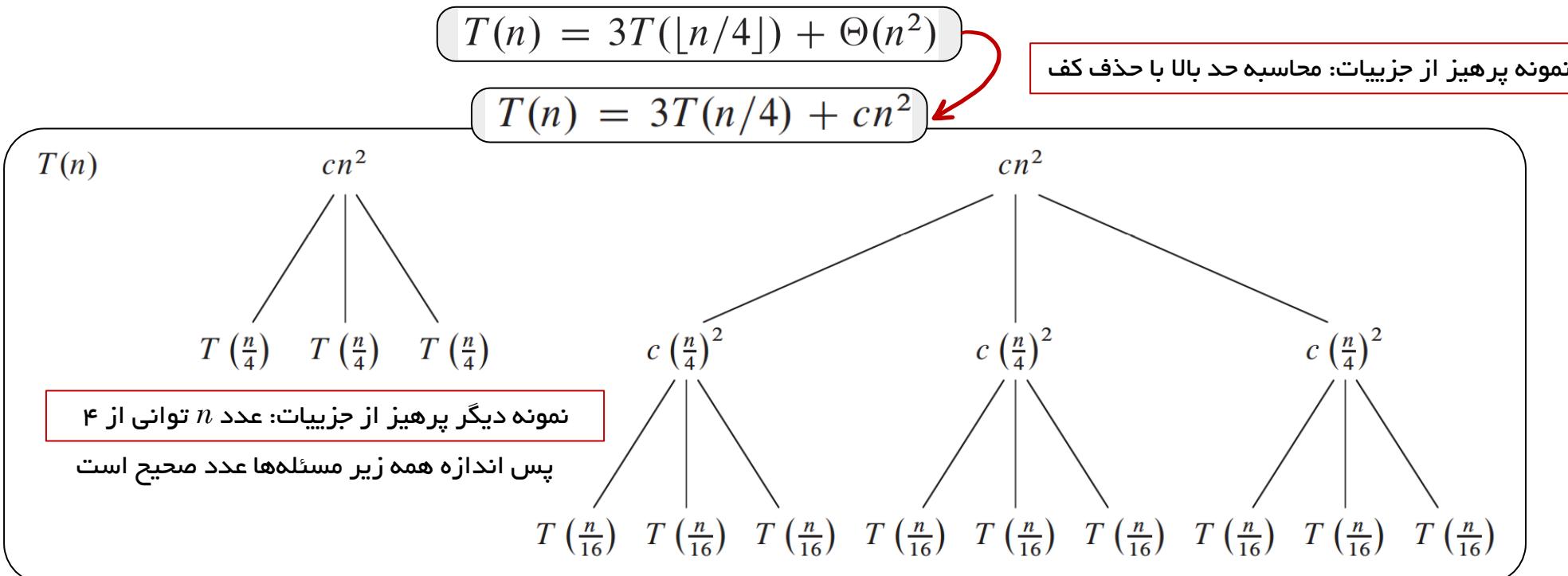


روش‌های حل رابطه بازگشتی: درخت بازگشت

فصل ۴.۴ کتاب

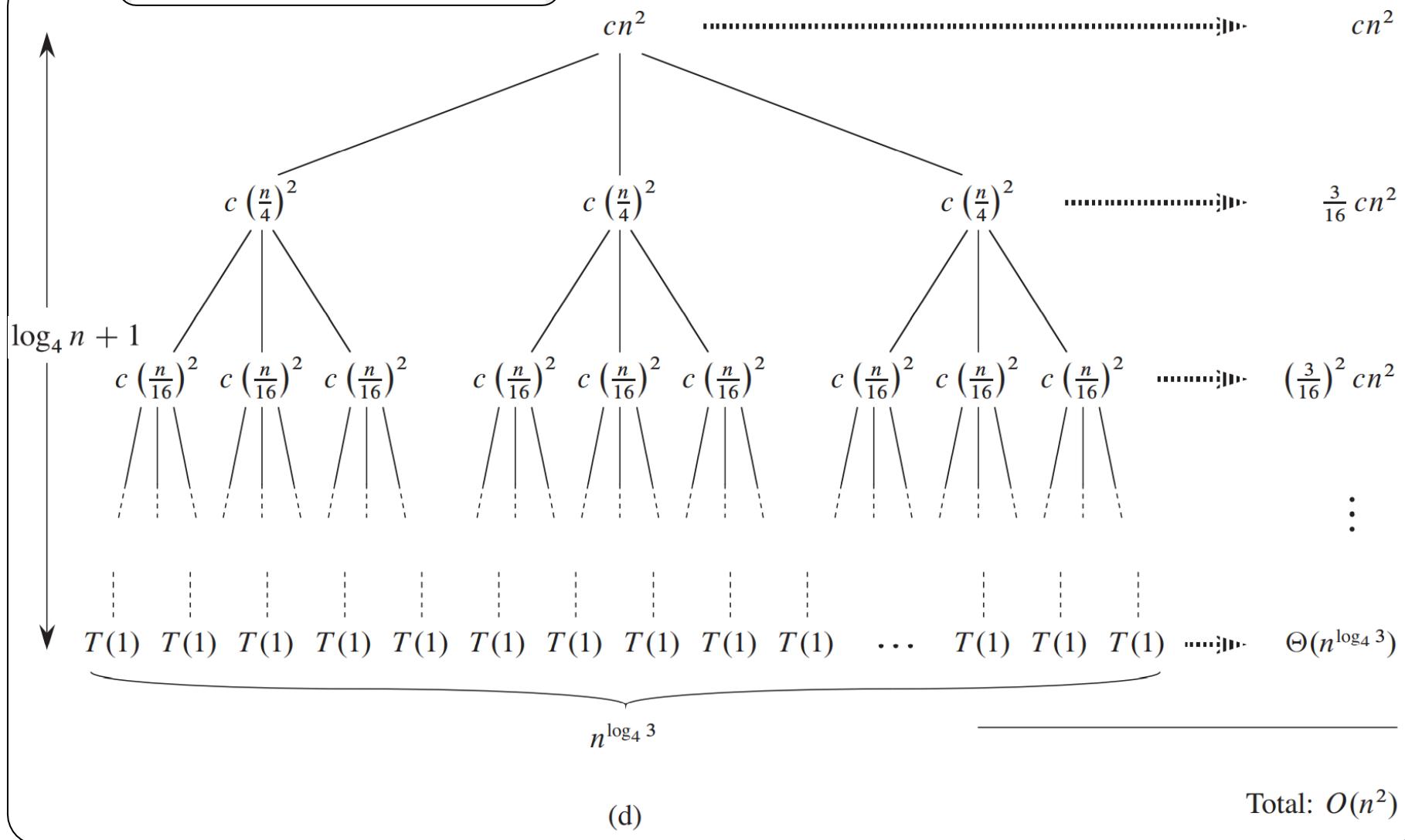
روش درخت بازگشتی

- روش جایگذاری و استقرار مناسب برای حل روابط بازگشتی
- اما پیدا کردن حدس مناسب همیشه راحت نیست! ← درخت بازگشتی
- در محاسبات درخت بازگشت میتوان از برخی جزییات پرهیز کرد
- در آن صورت اثبات حدس به دست آمده از طریق استقرار ضرورت خواهد داشت



روش درخت بازگشتی

$$T(n) = 3T(n/4) + cn^2$$



محاسبه مجموع هم سطح‌ها

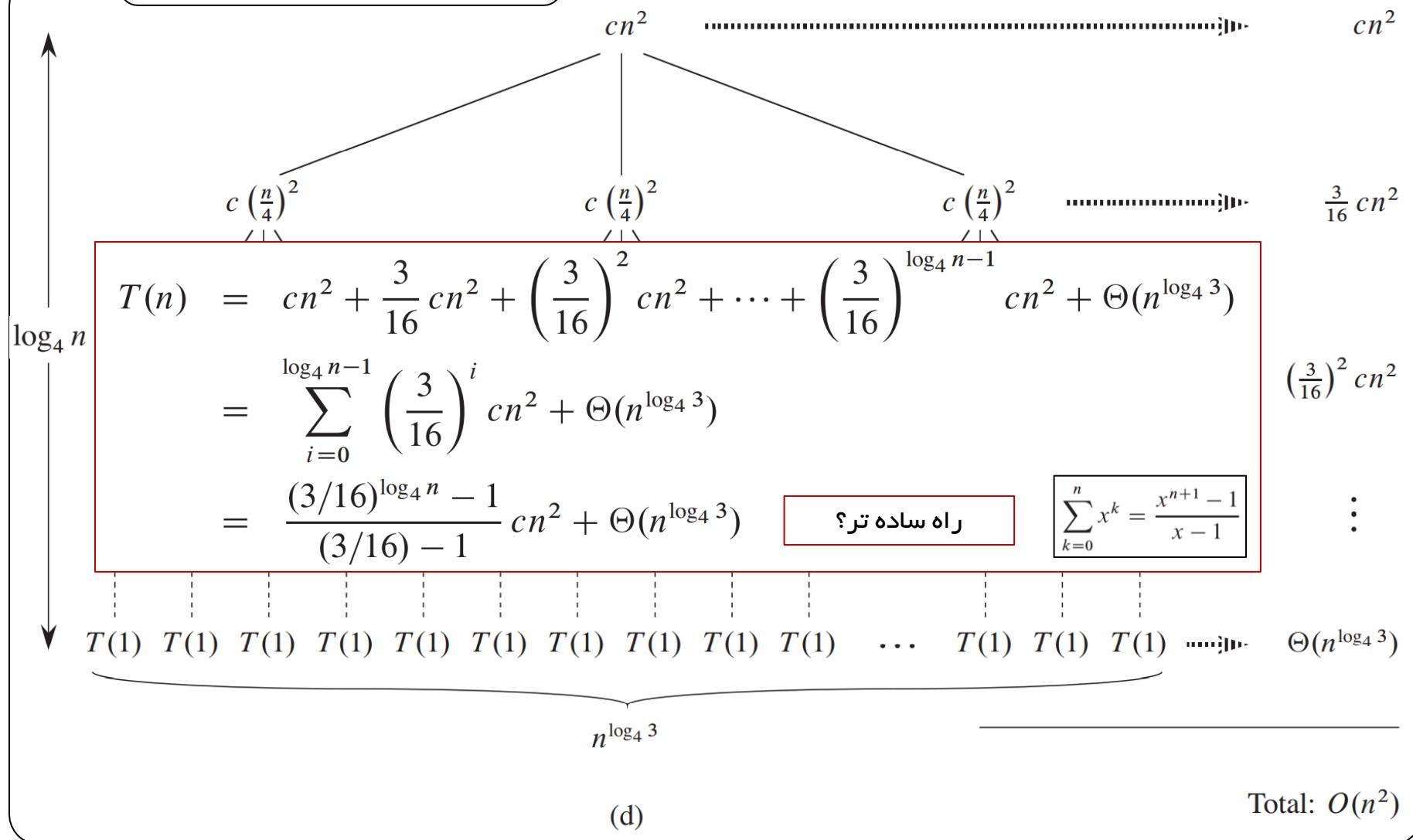
نماینده کاربردی

$$n/4^i = 1$$

محاسبه مجموع سطوح

روش درخت بازگشتی

$$T(n) = 3T(n/4) + cn^2$$



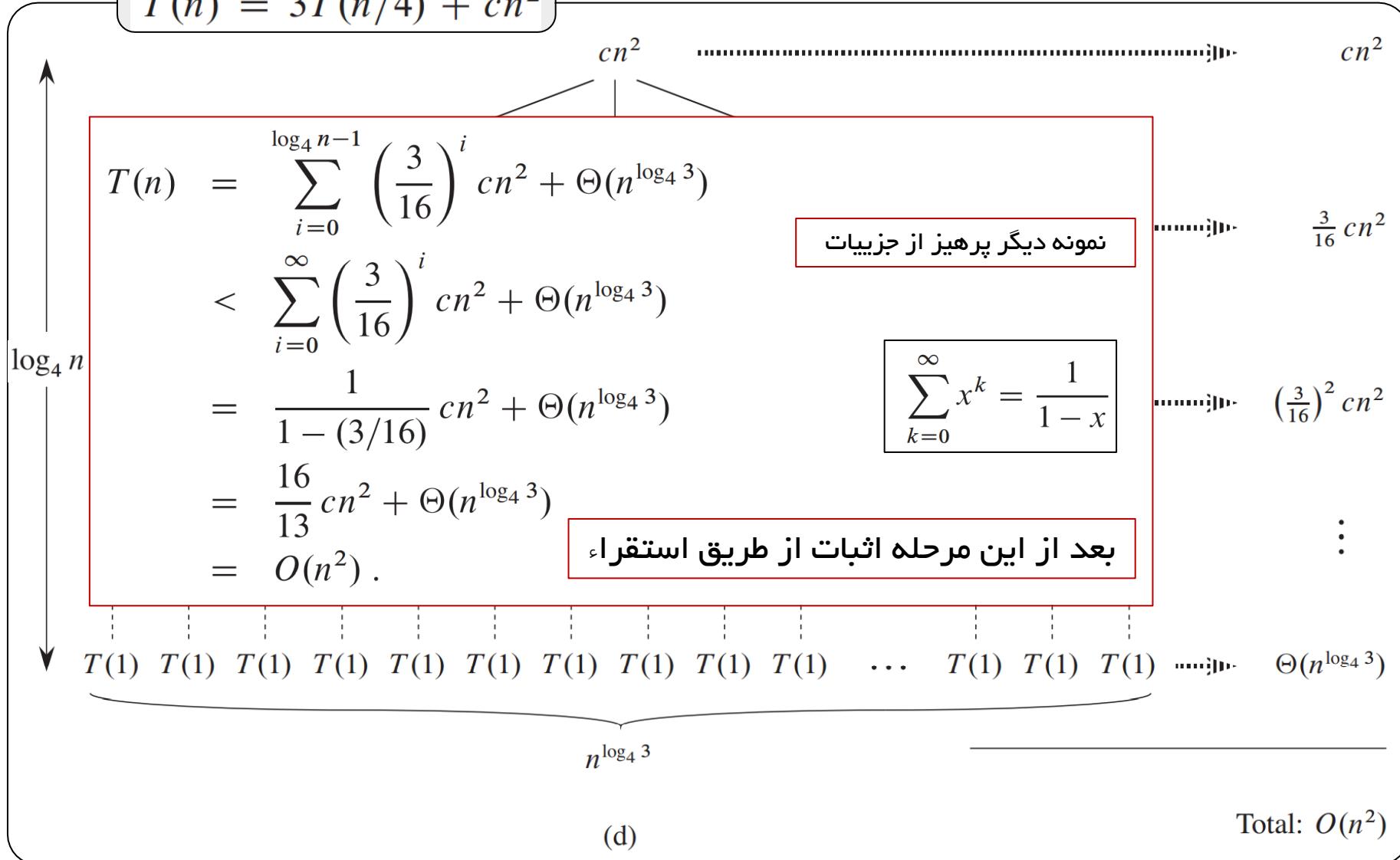
محاسبه مجموع هم سطح‌ها

آنالیز روش بازگشتی

محاسبه مجموع سطوح

روش درخت بازگشتی

$$T(n) = 3T(n/4) + cn^2$$



روش درخت بازگشتی - مثال ۲

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

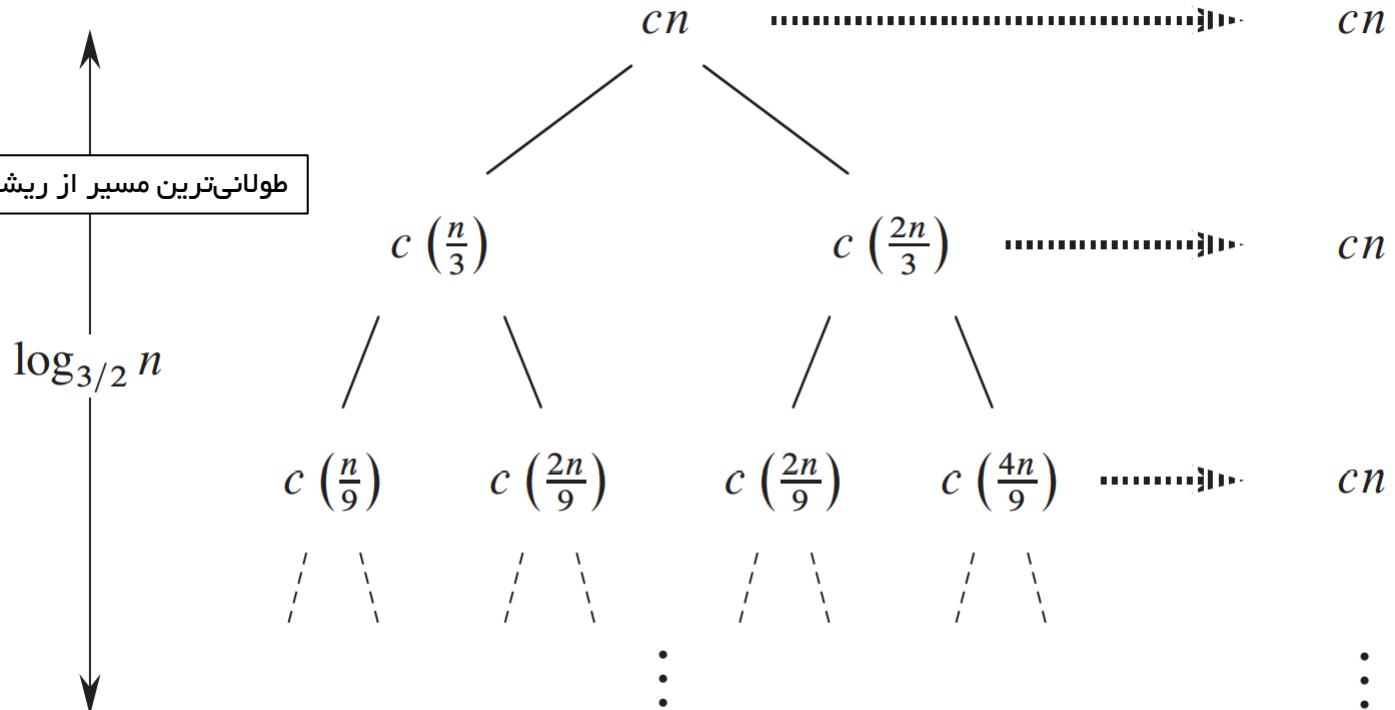
طولانی‌ترین مسیر از ریشه به برگ

محاسبه مجموع هم سطح‌ها

محاسبه مجموع سطوح

$$(2/3)^k n = 1 \text{ when } k = \log_{3/2} n$$

Total: $O(n \lg n)$

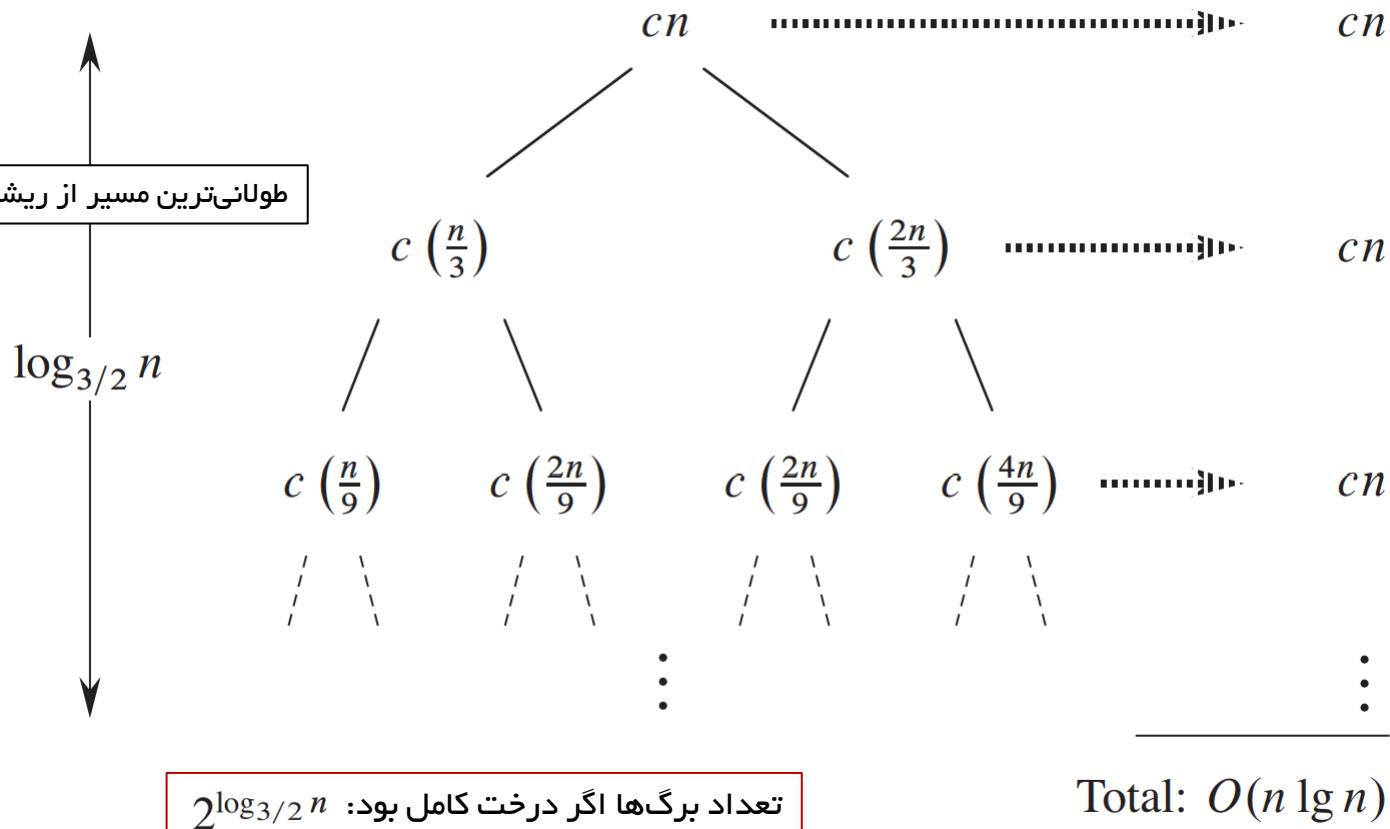


محاسبه مجموع سطوح

روش درخت بازگشتی - مثال ۲

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

طولانی‌ترین مسیر از ریشه به برگ



تعداد برگ‌ها اگر درخت کامل بود: $2^{\log_{3/2} n}$

Total: $O(n \lg n)$

محاسبه مجموع هم سطح‌ها

آیا درخت کامل است؟

آیا مجموع سطوح همیشه cn است؟

همه اینها را می‌توان دقیق محاسبه کرد اما...

نمونه‌هایی از پرهیز در محاسبه جزئیات

محاسبه مجموع سطوح

اثبات حدس روش درخت بازگشتی

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

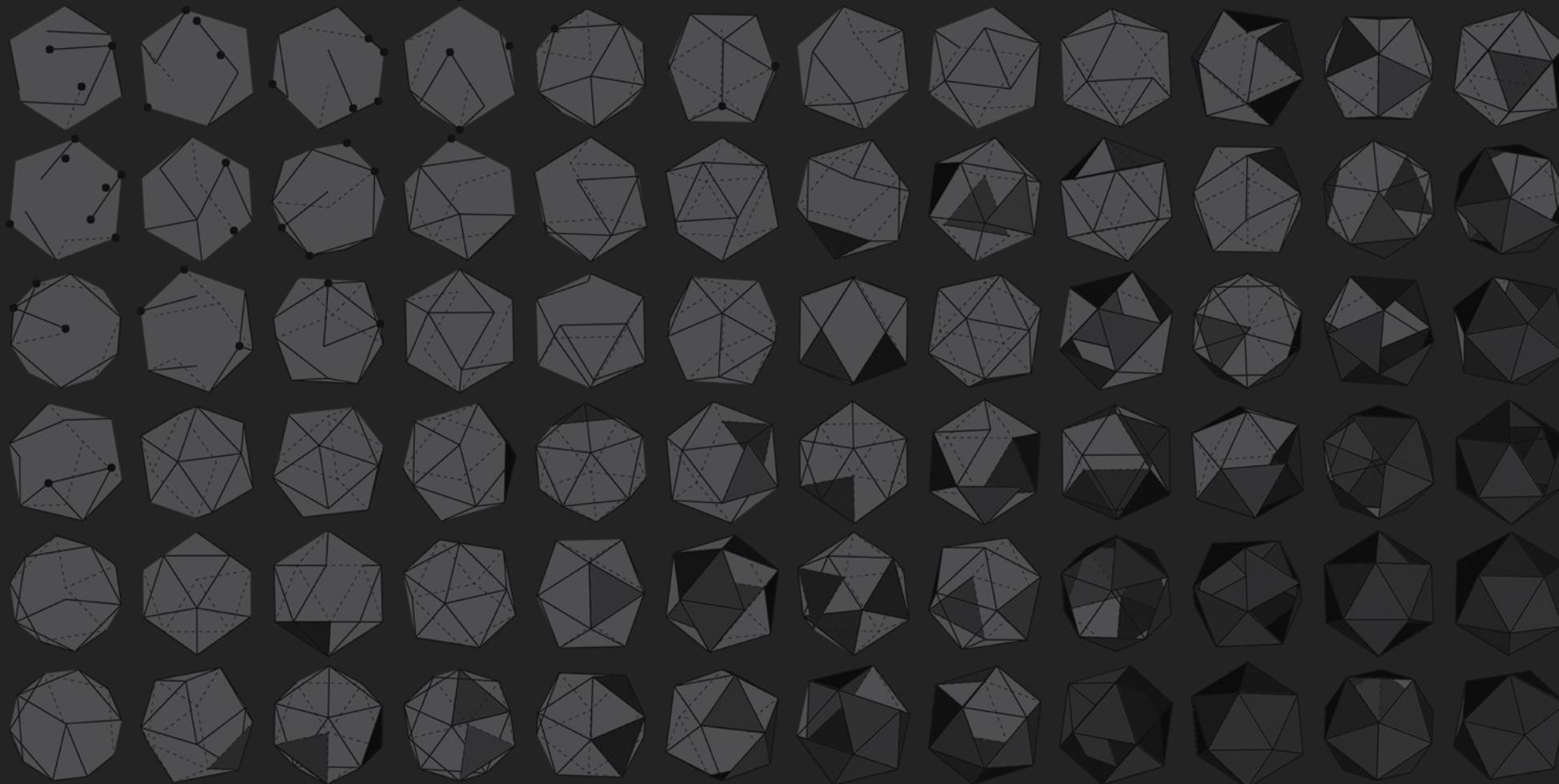
Total: $O(n \lg n)$

$$T(n) \leq dn \lg n$$

$$\begin{aligned} T(n) &\leq T(n/3) + T(2n/3) + cn \\ &\leq d(n/3) \lg(n/3) + d(2n/3) \lg(2n/3) + cn \\ &= (d(n/3) \lg n - d(n/3) \lg 3) \\ &\quad + (d(2n/3) \lg n - d(2n/3) \lg(3/2)) + cn \\ &= dn \lg n - d((n/3) \lg 3 + (2n/3) \lg(3/2)) + cn \\ &= dn \lg n - d((n/3) \lg 3 + (2n/3) \lg 3 - (2n/3) \lg 2) + cn \\ &= dn \lg n - dn(\lg 3 - 2/3) + cn \\ &\leq dn \lg n , \end{aligned}$$

$$d \geq c / (\lg 3 - (2/3))$$

اثبات حالت‌های پایه:
چه بازه‌ای برای d در نظر گرفته شود



ضرب ماتریسی یا Matrix Multiplication

فصل ۴.۲ کتاب

ضرب داخلی آرایه اعداد

Dot product. Given two length- n vectors a and b , compute $c = a \cdot b$.

Grade-school. $\Theta(n)$ arithmetic operations.

$$a \cdot b = \sum_{i=1}^n a_i b_i$$

$$a = [.70 \quad .20 \quad .10]$$

$$b = [.30 \quad .40 \quad .30]$$

$$a \cdot b = (.70 \times .30) + (.20 \times .40) + (.10 \times .30) = .32$$

Remark. “Grade-school” dot product algorithm is asymptotically optimal.

ضرب ماتریسی

Matrix multiplication. Given two n -by- n matrices A and B , compute $C = AB$.

Grade-school. $\Theta(n^3)$ arithmetic operations.

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

$$\begin{bmatrix} .59 & .32 & .41 \\ .31 & .36 & .25 \\ .45 & .31 & .42 \end{bmatrix} = \begin{bmatrix} .70 & .20 & .10 \\ .30 & .60 & .10 \\ .50 & .10 & .40 \end{bmatrix} \times \begin{bmatrix} .80 & .30 & .50 \\ .10 & .40 & .10 \\ .10 & .30 & .40 \end{bmatrix}$$

ضرب ماتریسی

Matrix multiplication. Given two n -by- n matrices A and B , compute $C = AB$.

Grade-school. $\Theta(n^3)$ arithmetic operations.

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

$$\begin{bmatrix} .59 & .32 & .41 \\ .31 & .36 & .25 \\ .45 & .31 & .42 \end{bmatrix} = \begin{bmatrix} .70 & .20 & .10 \\ .30 & .60 & .10 \\ .50 & .10 & .40 \end{bmatrix} \times \begin{bmatrix} .80 & .30 & .50 \\ .10 & .40 & .10 \\ .10 & .30 & .40 \end{bmatrix}$$

SQUARE-MAT-MULT(A, B, n)

```

let  $C$  be a new  $n \times n$  matrix
for  $i = 1$  to  $n$ 
    for  $j = 1$  to  $n$ 
         $c_{ij} = 0$ 
        for  $k = 1$  to  $n$ 
             $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$ 
return  $C$ 
```