

MSA + Brute force

طراحی الگوریتم‌ها – جلسه پنجم

Introduction to Algorithm

استاد: جوانمردی

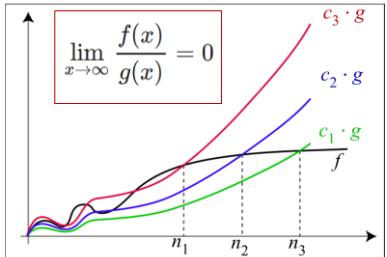
۱۳۹۹/۷/۱۴

حائلزای
برهانی
اسلام

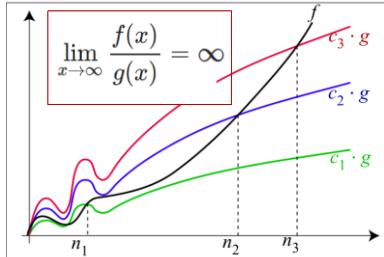
مرور جلسه قبل

نمادهای o و ω کوچک

The o -Notation



The ω -Notation



$$T_A(n) = o(T_B(n))$$

← الگوریتم A بهتر است الگوریتم 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

یکنواختی تابع
Monotonicity

بازتابی
Reflexivity

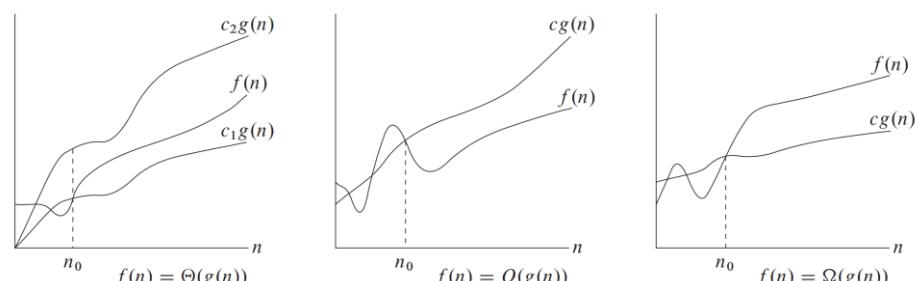
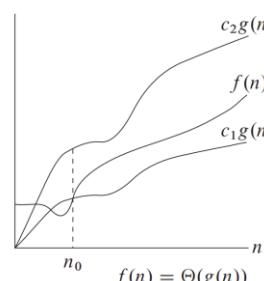
تقارنی
Symmetry

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

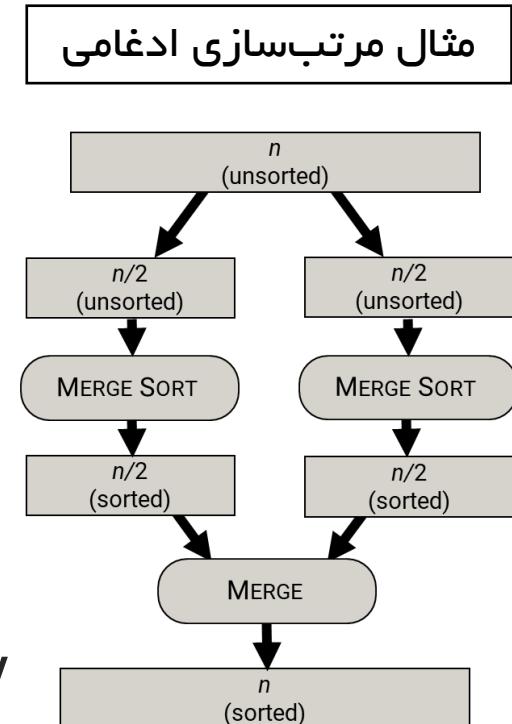
- مسئله زیرآرایه بیشینه یا Maximum Subarray
- انواع روش‌های حل معادله بازگشتی
- جایگذاری و استقرای ریاضی
- درخت بازگشت
- قضیه اصلی
- الگوریتم استراسن برای ضرب ماتریسی

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

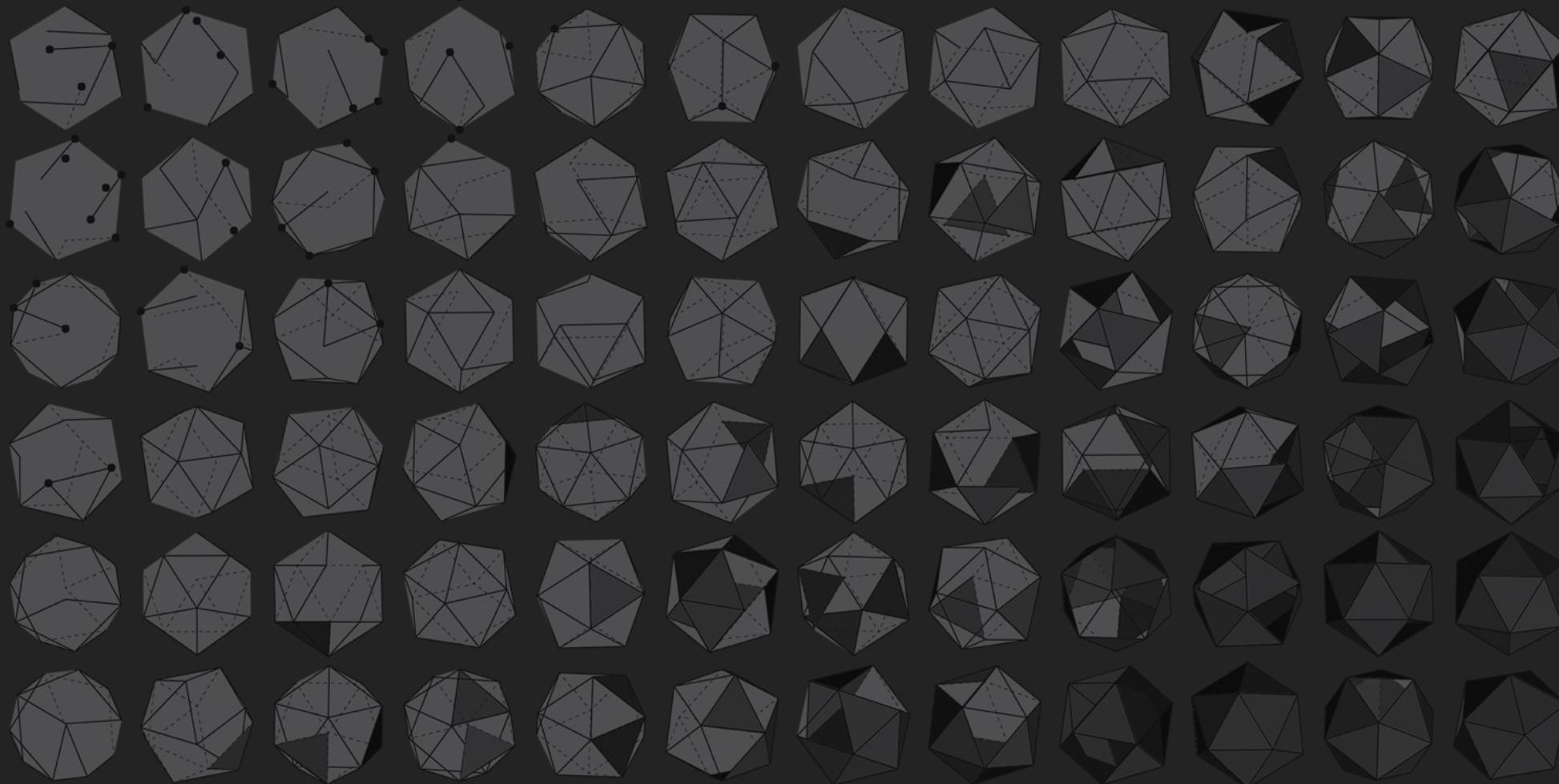
- **The divide-and-conquer approach:**
 - **Divide** the problem into a number of subproblems (that are smaller instances of the same problem)
 - **Conquer** the subproblems by solving them recursively
 - **Base case:** If the subproblem sizes are small enough, solve it in a straightforward manner
 - **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



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:

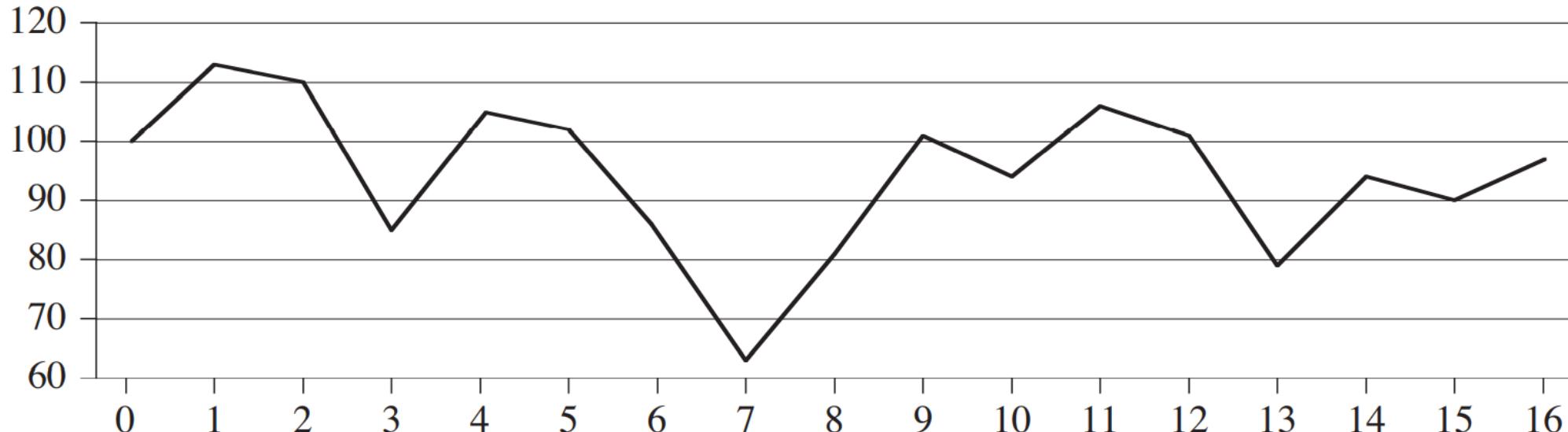


مسئله زیر آرایه پیشینه یا

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

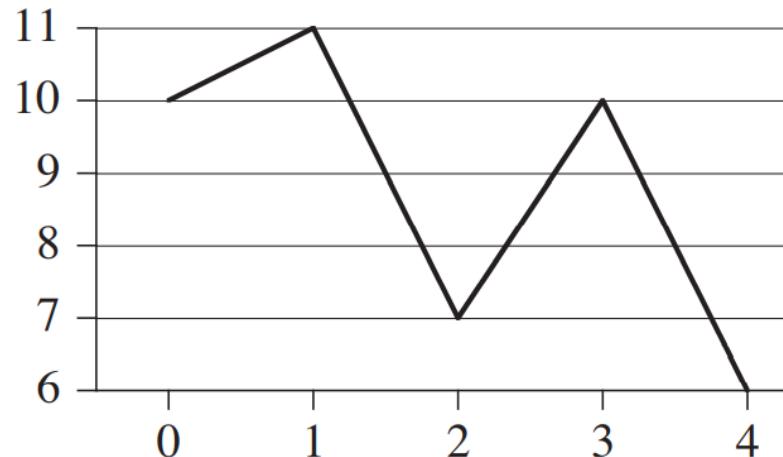
Goal: Maximize your profit



“buy low, sell high” ??

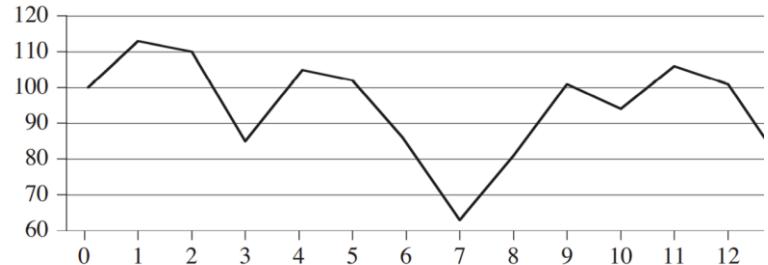
کسب حد اکثر سود در خرید و فروش سهام

- 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:



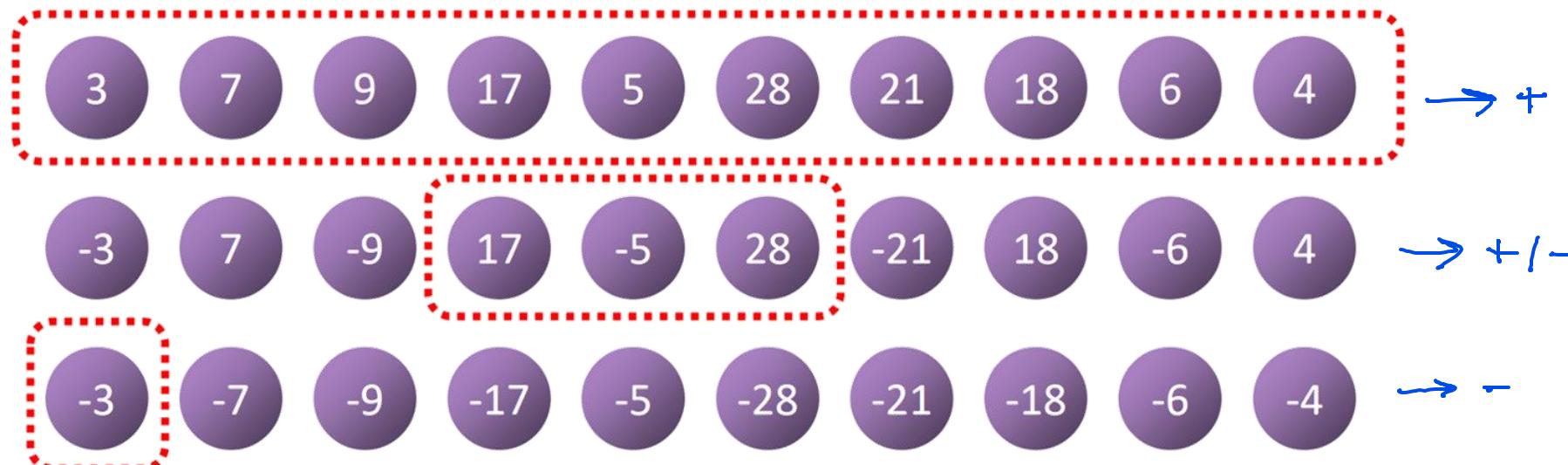
Day	0	1	2	3	4
Price	10	11	7	10	6
Change		1	-4	3	-4

مسئله زیرآرایه بیشینه یا Maximum Subarray



- 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 فروش

ردیف	نام	پیشنهاد	کیا و متوجه شد																	
۱۸۱۴	۱۵۸۵	۸۸۱	۱۱۰۸	۷۰۵	۷۹۸	۷۷۱	۱۶۰۲	۷۰۰	۷۰۰	۷۷۰	۱۶۰۱	۷۷۲	۷۷۱	۷۷۰	۷۷۰	۷۷۰	۷۷۰	۷۷۰	۷۷۰	ستندج
۹۲۴	۱۲۸۵	۲۸۰	۱۲۷۵	۷۰۰	۱۶۵۱	۱۲۹۱	۴۹۸	۴۰۰	۳۸۰	۱۱۲۲	۱۶۷۸	۱۲۶۵	۱۱۴۴	۵۹۶	۱۳۱۹	۹۷۷	۷۰۵	۹۰۰	۱۴۰۹	شاہرود
۱۲۷۷	۱۰۶۱	۹۲۳	۵۶۴	۹۰۸	۱۰۵۶	۱۳۲۲	۱۶۹۵	۷۹۲	۷۷۲	۷۱۹	۱۳۱۷	۱۰۵۹	۱۳۲۲	۱۰۴	۱۱۷۸	۱۱۳۴	۳۹۲	۱۰۵۷	۹۷۲	شهرکرد
۱۳۲۵	۶۱۹	۱۳۰۴	۳۰۴	۱۲۸۹	۵۱۵	۱۸۰۲	۱۶۷۶	۱۱۷۷	۱۱۰۲	۱۱۰۰	۱۱۱۲	۹۵۹	۱۸۱۳	۴۸۵	۱۰۰۹	۱۰۱۵	۷۷۲	۱۶۲۸	۵۹۴	شیراز
۱۶۹۲	۱۶۰۵	۵۳۰	۱۰۵۰	۲۲۵	۱۰۰۷	۷۰۰	۱۰۰۲	۷۹۹	۷۷۹	۶۱۷	۱۰۷۵	۸۰۷	۶۱۸	۴۸۰	۷۸۲	۴۰۱	۳۰۲	۳۷۶	۱۰۰۵	قزوین
۱۹۹۵	۱۱۷۲	۵۱۲	۸۷۵	۴۹۷	۸۹۰	۱۰۱۱	۱۰۰۷	۷۸۱	۷۶۱	۵۸۸	۱۱۲۸	۷۱۵	۹۷۰	۷۷۹	۱۰۷۲	۷۷۲	۱۱۷۲	۵۷۸	۸۷۸	قم
۱۲۹۶	۱۰۴۸	۶۲۶	۷۹۲	۶۱۱	۹۷۲	۱۱۰۲	۱۱۸۸	۹۰۰	۹۰۰	۷۸۲	۱۱۲۸	۹۳۵	۱۰۴۴	۱۰۷۲	۱۱۰۲	۱۱۰۲	۱۱۰۲	۱۱۰۲	۱۱۰۲	کاشان
۹۹۹	۴۸۰	۱۶۱۴	۸۷۵	۱۰۰۲	۱۰۰۵	۱۹۱۷	۱۷۹۰	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	۱۲۸۷	کرمان
۱۸۰۰	۱۱۷۹	۹۰۸	۹۷۲	۸۲۰	۶۹۲	۸۸۰	۱۰۷۸	۷۷۰	۷۰۰	۱۰۰۷	۱۰۰۷	۹۰۸	۹۰۸	۹۰۸	۹۰۸	۹۰۸	۹۰۸	۹۰۸	۹۰۸	کرمانشاه
۱۰۰۰	۱۷۷۱	۶۷	۱۶۲۵	۵۰۷	۱۶۶۵	۱۲۷۴	۰۰۰	۱۹۳	۱۷۲	۱۱۷	۱۱۷	۱۱۷۰	۹۷۱	۸۷۴	۱۲۴	۱۲۴	۹۷۰	۱۳۹۴	۱۳۹۴	گرگان
۱۵۵۸	۳۷۰	۱۶۹۸	۵۹۸	۱۵۰۳	۸۷۹	۲۱۶۷	۲۱۶۷	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	۱۰۷۲	لار
۷۷۰	۱۶۹۳	۹۳۰	۱۶۹۷	۱۱۰۶	۱۶۹۲	۱۷۷۲	۱۰۳	۷۸۰	۷۸۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱۰۰	لطف آباد
۷۸۱	۱۳۷۶	۶۳۶	۱۶۹۸	۱۱۰۷	۱۶۹۷	۱۷۷۲	۲۲۲	۷۶۱	۷۶۱	۱۰۰۴	۱۳۷۶	۱۷۶۸	۱۰۰۵	۱۲۲۲	۱۸۰۱	۱۳۳۳	۱۱۷۷	۱۲۰۵	۱۳۷۶	مشهد
۱۹۹۱	۱۸۸۵	۱۰۶۷	۱۰۰۸	۷۰۵	۱۰۰۸	۴۷۷	۱۰۰۹	۹۳۶	۹۱۶	۹۲۰	۲۱۶۱	۹۲۲	۰۰۰	۹۲۷	۱۲۲	۴۷۵	۹۰۰	۵۱۲	۱۰۴۹	مهاباد
۵۰۰۴	۸۲۷	۱۶۷۱	۱۶۸۸	۲۰۱۵	۱۶۹۹	۲۰۲۰	۱۲۷۶	۱۱۹۵	۱۱۷۶	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	۱۰۰۷	سیروان
۲۱۱۰	۲۲۲۱	۱۱۷۷	۱۷۲۸	۰۰۹	۱۶۶۸	۳۷۷	۱۶۰۲	۹۲۲	۹۲۲	۹۷۰	۲۲۸۷	۱۲۷۱	۱۱۷	۱۱۷۲	۱۱۷۲	۱۱۷۲	۱۱۷۲	۱۱۷۲	۱۱۷۲	نوروز
۱۶۷۷	۱۶۲۱	۷۱۷	۱۰۴۴	۴۹۱	۸۱۳	۸۸۹	۱۲۲۹	۵۸۶	۵۶۶	۳۷۷	۱۰۷۷	۹۷۸	۸۷۸	۴۹۶	۹۱۰	۹۷۷	۵۹۰	۷۶۱	۷۶۱	همدان
۱۶۰۵	۷۹۳	۱۱۱۸	۷۸۱	۱۱۰۵	۹۹۲	۱۶۱۷	۱۰۰۰	۹۱۷	۹۱۷	۹۷۷	۱۰۹۷	۱۰۹۷	۹۱۳	۱۰۰۱	۱۰۰۰	۲۰۰	۱۳۷۲	۱۳۷۲	۱۳۷۲	پاسوج
۸۹۷	۸۰۷	۱۰۰۷	۷۷۵	۱۰۰۷	۹۳۷	۱۰۰۹	۱۰۰۹	۹۲۶	۹۰۶	۹۷۸	۹۱۳	۱۰۰۱	۱۰۰۰	۲۰۰	۱۳۷۲	۱۳۷۲	۱۳۷۲	۱۳۷۲	۱۳۷۲	یزد

نمره ۵۰۲

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)

```

$O(n^2)$

$O(n)$

$O(n^2)$

$O(n^2)$

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

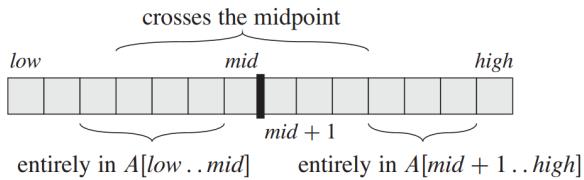
حل زیرآرایه بیشینه با روش تقسیم و حل

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

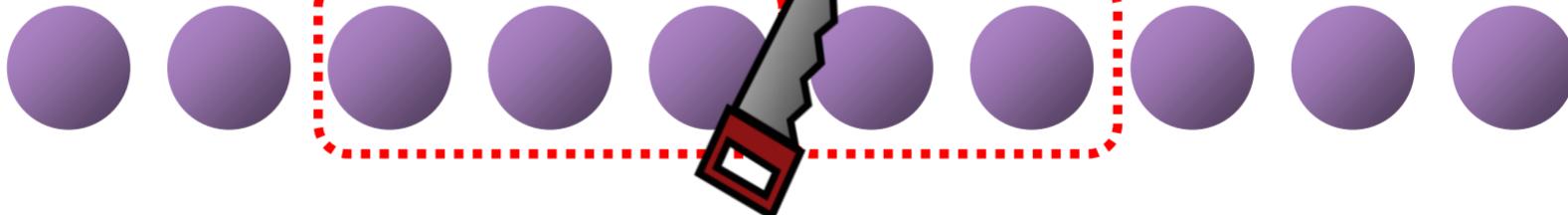
i



Case 2: right



Case 3: cross the middle



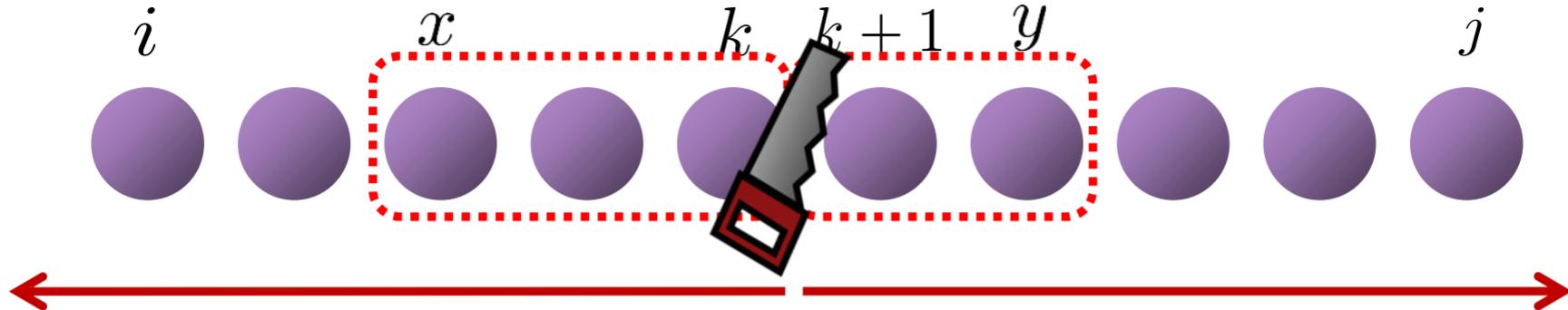
Case 1: $\text{MaxSub}(A, i, j) = \text{MaxSub}(A, i, k)$

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

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

حالت سوم: آرایه‌ای که از وسط عبور می‌کند!

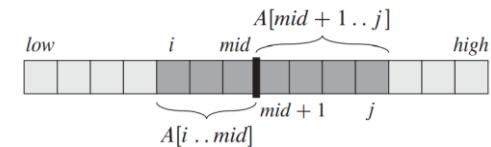
- Goal: find the maximum subarray that crosses the middle



(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)



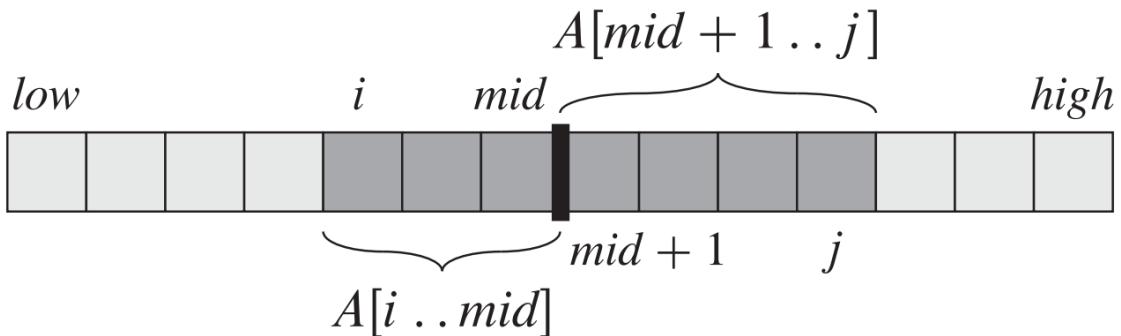
- Observation
 - The sum of $A[x \dots k]$ must be the maximum among $A[i \dots k]$ (left: $i \leq k$)
 - The sum of $A[k + 1 \dots y]$ must be the maximum among $A[k + 1 \dots j]$ (right: $j > k$)
 - Solvable in linear time $\rightarrow \Theta(n)$

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)
    
```



$\Theta(n)$

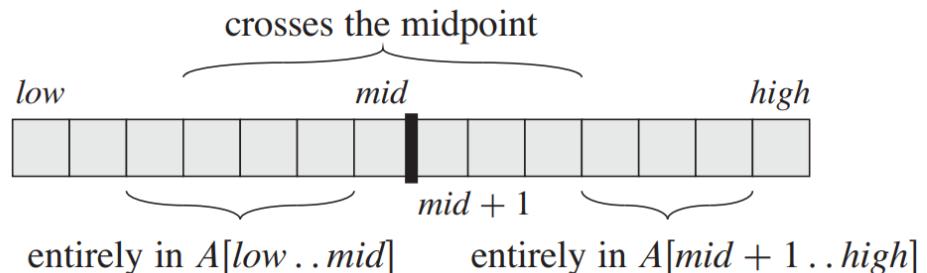
حل زیرآرایه بیشینه با روش تقسیم و حل

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 ( $cross-low, cross-high, cross-sum$ )

```



* مطالعه درست
max را تابعی بنویس

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

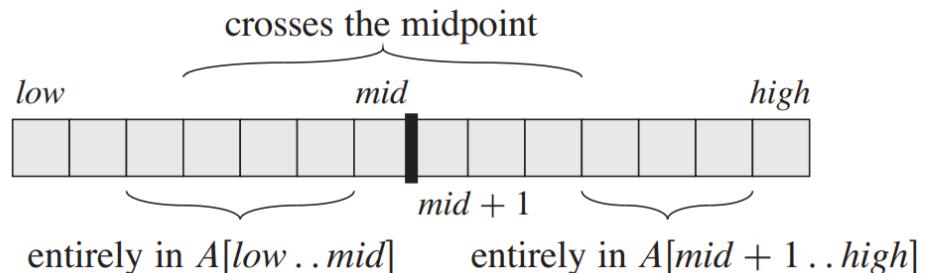
حل زیرآرایه بیشینه با روش تقسیم و حل

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($A, low, high$)

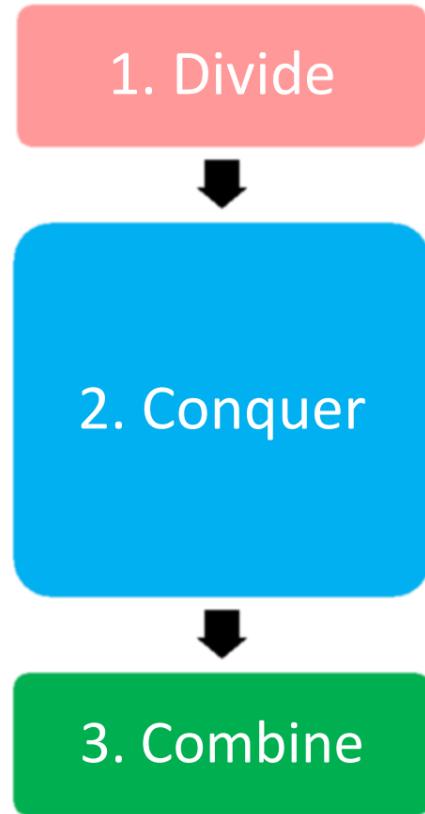
```

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  $left-sum \geq cross-sum$   $\cancel{O(1)}$ 
11       return ( $left-low, left-high, left-sum$ )
12     elseif  $right-sum \geq left-sum$  and  $right-sum \geq cross-sum$ 
13       return ( $right-low, right-high, right-sum$ )
14     else return ( $cross-low, cross-high, cross-sum$ )

```

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

تحليل زمانی روش تقسیم و حل MSA



1. Divide

- Divide a list of size n into 2 subarrays of size $n/2$

$\Theta(1)$

2. Conquer

- Recursive case ($n > 1$)
 - find **MaxSub** for each subarrays
- Base case ($n = 1$)
 - Return itself

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

$\Theta(1)$

3. Combine

- Find **MaxCrossSub** for the original list
- Pick the subarray with the maximum sum among 3 subarrays

$\Theta(n)$

$\Theta(1)$

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

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

تحلیل زمانی روش تقسیم و حل

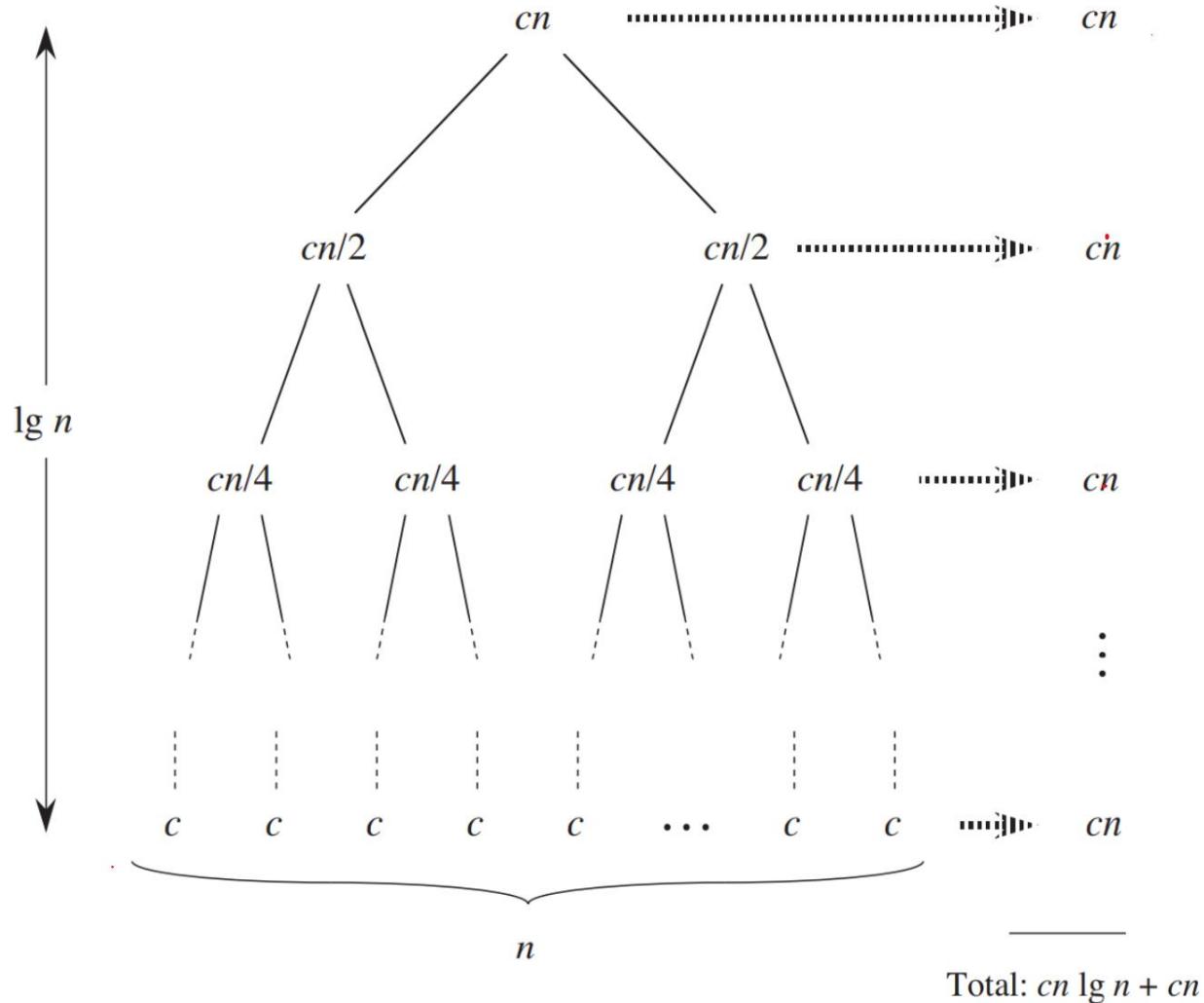
$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + O(n) & \text{if } n \geq 2 \end{cases}$$

ساده‌سازی: تعداد ورودی‌ها توانی از ۲

تعداد ورودی زیرمسئله‌ها
همیشه عدد صحیح

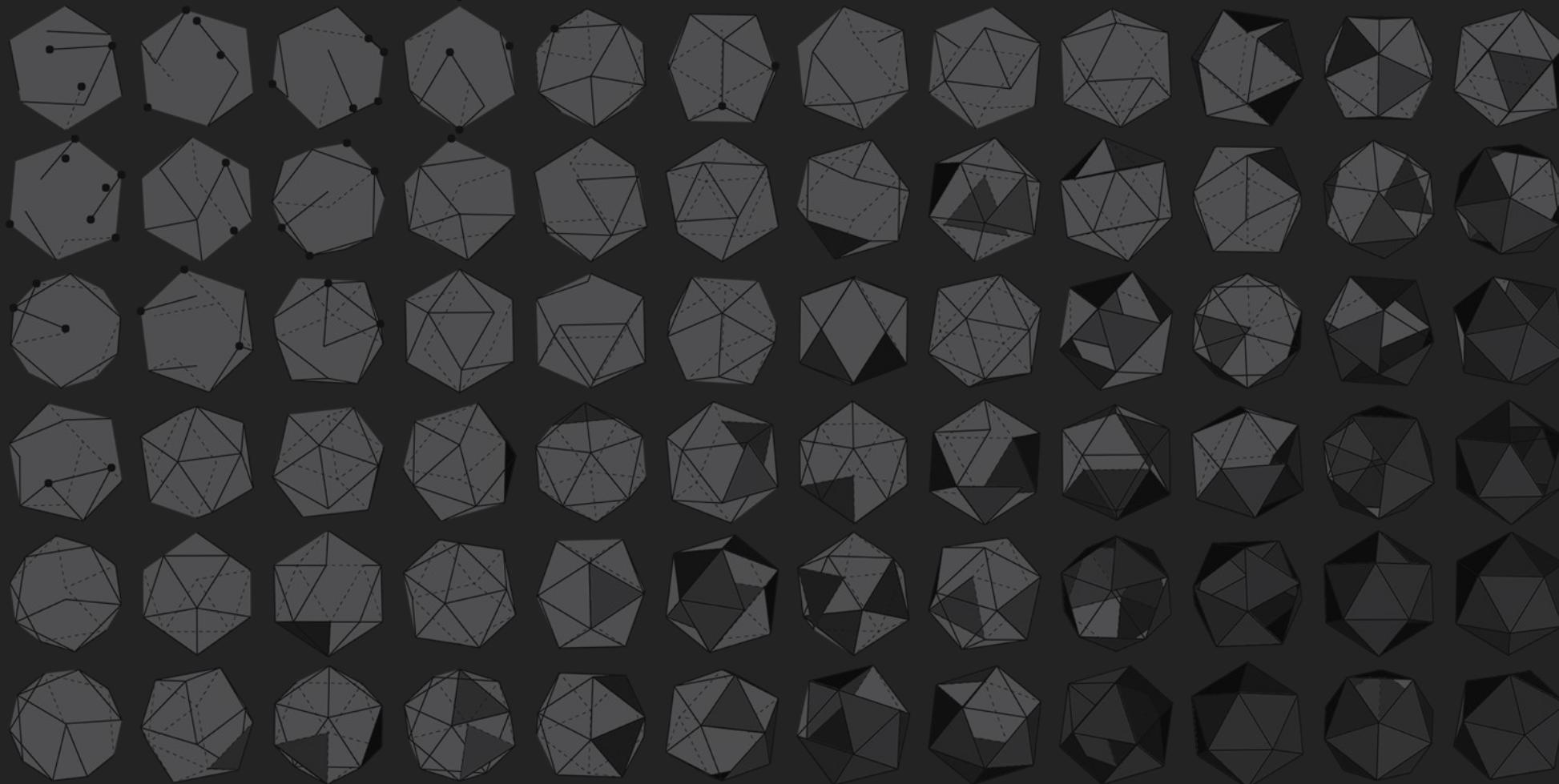
Merge Sort

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



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

مروی 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

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

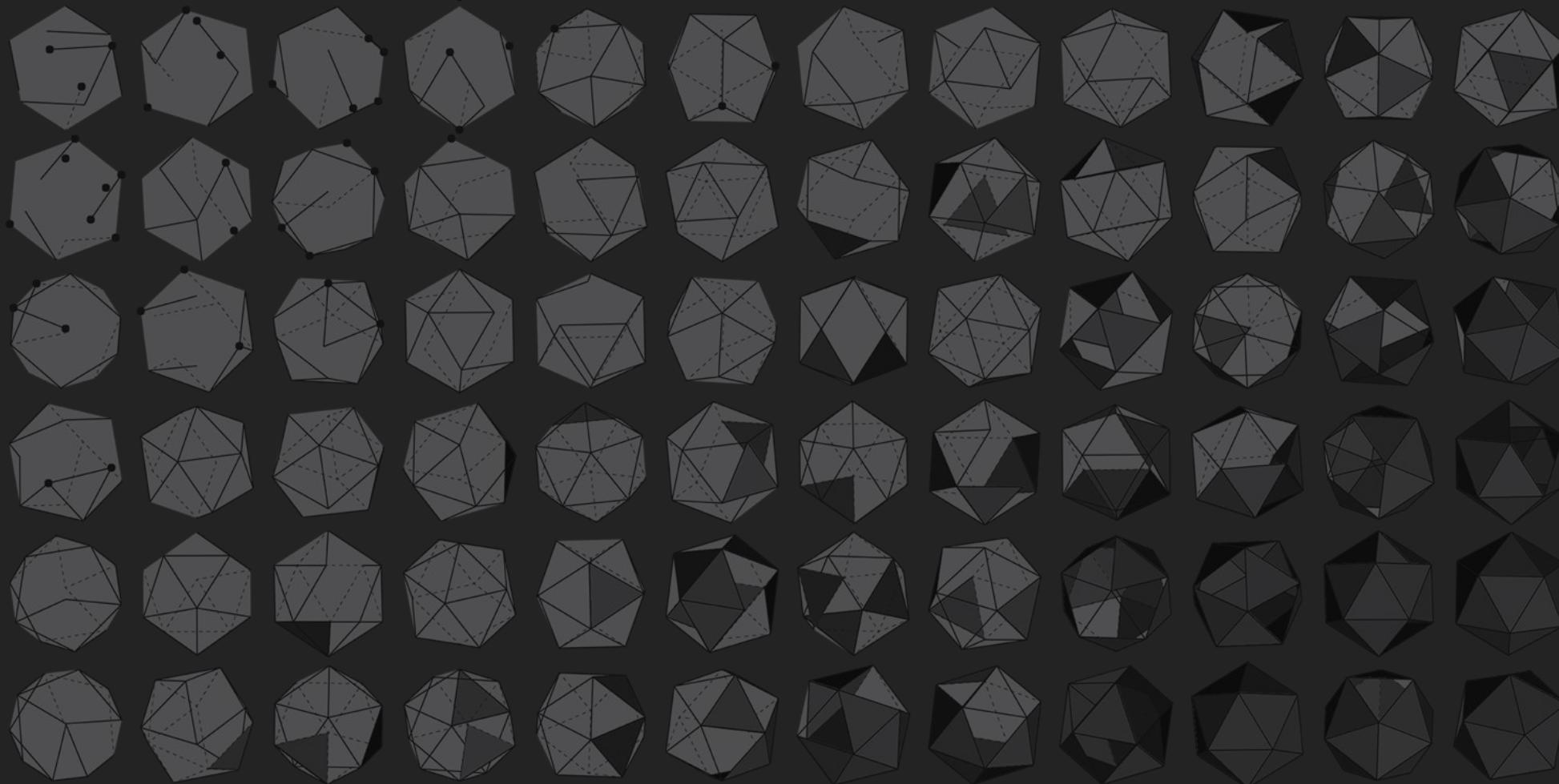
جزیيات فنی حل رابطه بازگشت

- 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:
 1. Guess the form of the solution
 2. 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

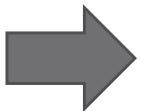
- Example:

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

- Guess:

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

حدس از کجا؟ میتوانیم از روش درخت بازگشت بهره بگیریم



- 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 نگرانی از بابت حالت پایه وجود ندارد زیرا قطعا:
- 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

$$\begin{aligned} T(n) &= 2T(n/2) + cn \leq 2d \cdot n/2 \cdot \lg(n/2) + cn \\ &\leq d \cdot n \cdot \lg(n/2) + cn \\ &\leq d \cdot n \cdot [\lg(n) - \lg(2)] + cn \\ &\leq d \cdot n \cdot \lg(n) + cn - dn \end{aligned}$$

$$T(n) \leq d \cdot n \cdot \lg(n)$$

دسته‌ای از d که رابطه را صحیح می‌کند

$$cn - dn \leq 0 \rightarrow d \geq c$$

اثبات از طریق استقراء در روش جایگذاری

- حالت پایه لزوماً باید $n = 1$ باشد

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

تمام اعداد توان ۲ هستند

$$O(g(n)) = \{f(n) : \exists c > 0, \boxed{n_0 > 0} \text{ s.t. } \forall n \geq n_0 : f(n) \leq c \cdot g(n)\}$$

$$T(n) \leq d \cdot n \cdot \lg(n) \quad \text{if } n=1$$

$$\quad \quad \quad \text{if } n=2$$

مثال روش جایگذاری

- روش جایگذاری میتواند برای تعیین حد بالا و یا پایین کاربرد داشته باشد

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

$c > 0$ 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

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

$$\left. \right\} c \geq 1$$

مثال روش جایگذاری - حالت پایه

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

- برای حل حالت پایه فرض میکنیم $T(1) = 1$

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

- در استقراء حالت پایه باید به همه حالت‌های بعدی قابل تعمیم باشد

برای $n > 3$ همه $T(n)$ ها با استفاده از $T(2)$ و $T(3)$ قابل نمایش خواهند بود

پس بجای اینکه $n = 1$ را حالت پایه در نظر بگیریم، $n = 2$ و $n = 3$ را حالت پایه در نظر میگیریم

$$T(3)$$

$$c \geq 2$$

$$T(2)$$

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!

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

برای n های بزرگ $\lfloor n/2 \rfloor + 17$ و $\lfloor n/2 \rfloor$ تفاوت چندانی نخواهد داشت!

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

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

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

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

$$= cn + 1, \times$$

تصحیح حدس زده شده

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

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

فرض $T(n) = O(n) \rightarrow T(n) \leq cn \times \rightarrow T(n) \leq cn - d$, 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: به حکم یک جمله از درجه کمتر اضافه میکنیم
- درجه جمله اضافه با حکم برابر باشد: یک فاکتور لگاریتم در حکم کمتر حدس زدیم
- 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

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

فرض	$T(n) = O(n) \rightarrow T(n) \leq cn$
-----	--

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

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

خطاهای احتمالی در استقراء

- خطأ کردن در استقراء خیلی هم سخت نیست!

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

فرض $T(n) = O(n) \rightarrow T(n) \leq cn \quad \text{X}$

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

تغییر متغیر برای حل استقراء

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

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

↓

$$S(m) = O(m \lg m)$$

↓

$$T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$$