

## چند تست کنکور 98

۷۰- مرتبه زمانی الگوریتم زیر کدام است؟

**f(n)**

```
{
    x = ۲; y = ۱;
    while(y <= n){
        y = y * x;
        x = x * x;
    }
}
```

$\log(\log(\log n))$  (۴)       $\log(\log n)$  (۳)       $\log n \log n$  (۲)       $n \log n$  (۱)

۷۲- فرض کنید زیر برنامه  $C = \text{aux}(A, B)$  دو ماتریس  $A$  و  $B$  با اندازه  $n \times n$  را ضرب کرده و نتیجه را در  $C$  می‌گرداند. مقدار برگشتی تابع زیر برای ماتریس  $M$  کدام است؟

**Mat(M)**

```
{
    P = M ; Q = M
    for i = ۱ to n-۱ do{
        P = aux(P, M)
        Q = aux(Q, P)
    }
    return(Q)
}
```

$M^{\frac{n(n+1)}{2}}$  (۴)       $M^{\frac{n(n-1)}{2}}$  (۳)       $M^{n-1}$  (۲)       $M^n$  (۱)

۷۹- اگر  $T(n) = ۲T(\frac{n}{۴}) + \theta(۱)$  از مرتبه  $\theta(\sqrt{n})$  باشد، آنگاه  $T(n) = ۲T(\frac{n}{۴}) + \theta(\sqrt{n})$  از کدام مرتبه است؟

$\theta(\sqrt{n} \log n)$  (۴)       $O(n\sqrt{\log n})$  (۳)       $O(\log^2 n)$  (۲)       $O(n)$  (۱)

۸۰- اگر  $U(m) = T(۲^m)$  باشد و  $T(n) = T(\frac{n}{۲}) + \log n$ ، آنگاه رابطه بازگشتی  $U$  کدام است؟

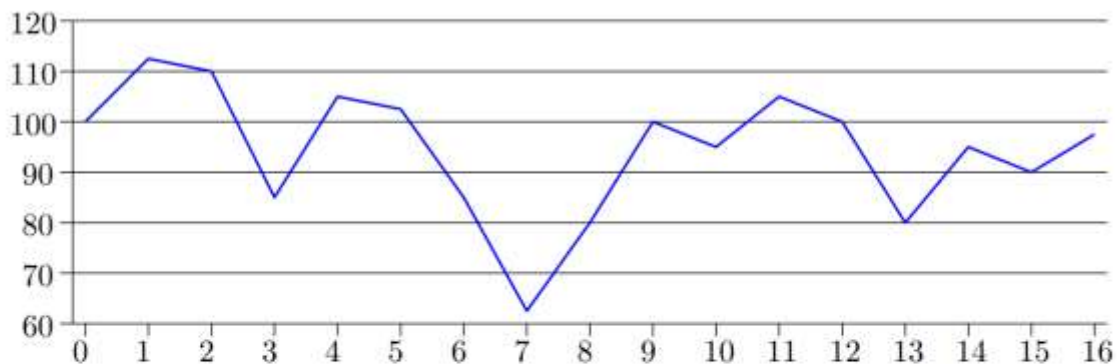
$U(m) = U(m-۱) + m$  (۴)       $U(m) = U(m-۱) + ۱$  (۱)

$U(m) = U(m-۱) + ۲^m$  (۴)       $U(m) = U(m-۱) + \log m$  (۳)

## Maximum Subarray Problem

**Example:** I am giving you perfect stock market predictions of a company ABC for the next thirty days. But under a condition — you can buy and sell the stocks only once. That's it. You can't do more than one "buy-sell" transaction. What would be the best day to buy the stocks? And when do you need to sell them to make the maximum profit?

### Buying and Selling



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

### Problem

*When were the best times to buy and sell?*

- A Brute-force Solution

- Try every possible pair of buy and sell dates (buy precedes sell date)
- Period of  $n$  days has  $\binom{n}{2}$  pairs of dates  $\binom{n}{2}$  is  $\Theta(n^2)$
- Best we could hope is to evaluate each pair in constant time and approach would take  $\Omega(n^2)$
- Can we do better?

- Difficulty: We don't necessarily buy at the lowest price or sell at the highest price.

- Example:



## Problem Transformation

- Consider the array  $A[1 \dots 16]$  of change in price.

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

Maximum subarray  $A[8 \dots 11]$

- The *maximum subarray* is the contiguous subarray whose elements have the largest sum. Here, it is  $A[8 \dots 11]$ .
- So the best times to buy and sell are days 7 and 11.
- We reduced our original problem to:

### Problem

Given an array  $A[1 \dots n]$  of  $n$  numbers, find the indices  $p, q$  such that  $1 \leq p \leq q \leq n$  and  $\sum_{i=p}^q A[i]$  is maximum.

## The MAXIMUM SUBARRAY Problem

### Brute-force approach to MAXIMUM SUBARRAY

```
1: procedure MAXIMUM-SUBARRAY( $A, 1, n$ )
2:    $\text{max} \leftarrow -\infty$ 
3:   for  $i \leftarrow 1, n$  do
4:     for  $j \leftarrow i, n$  do
5:        $\text{sum} \leftarrow 0$ 
6:       for  $k \leftarrow i, j$  do
7:          $\text{sum} \leftarrow \text{sum} + A[k]$ 
8:       if  $\text{sum} > \text{max}$  then
9:          $\text{max} \leftarrow \text{sum}, p \leftarrow i, q \leftarrow j$ 
10:  return  $p, q, \text{max}$ 
```

- Running time?  $\Theta(n^3)$ .

## Applications

The maximum subarray problem has several applications. Some of the well-known applications are in genomic sequence analysis and computer vision. They are used in genomic sequence analysis to identify important segments of protein sequences like GC-rich regions, and regions of high charge. In computer vision, they find their use in detecting the brightest area in bitmap images.

مثال ۱-۱۹ الگوریتم زیر برای تعیین عنصر بیشینه آرایه  $A[1..n]$  مفروض است. این الگوریتم، اندیس عنصر بیشینه را توسط  $j$  برمی گرداند. با فرض آن که عناصر آرایه  $A$  اعداد صحیح متمایز و متساوی الاحتمال هستند، میانگین زمان اجرای دستورالعمل  $d$  یعنی  $j = i$  را به دست آورید.

**Algorithm FindMax** (  $A, n, j$  )

- a)  $j = n$
- b) **for**  $i = n-1$  **downto**  $1$  **do**
- c)     **if**  $A[i] > A[j]$  **then**
- d)          $j = i$

حل. تعداد دفعات اجرای دستورالعمل های الگوریتم فوق در بدترین حالت، بهترین حالت و حالت میانگین، در زیر آمده است.

تعداد دفعات اجرا در بهترین حالت	تعداد دفعات اجرا در حالت میانگین	تعداد دفعات اجرا در بدترین حالت	دستورالعمل
۱	۱	۱	a
$n-1$	$n-1$	$n-1$	b
$n-1$	$n-1$	$n-1$	c
*	$A_n = ?$	$n-1$	d

با توجه به جدول مزبور مشاهده می شود که  $A_n$  یعنی میانگین زمان اجرای دستورالعمل  $j = i$  در رابطه  $0 \leq A_n \leq n-1$  صادق است. برای روشن شدن مطلب به بررسی حالت خاص  $n = 3$  می پردازیم. با توجه به متمایز بودن عناصر آرایه  $A$ ، حالت های امکان پذیر برای آرایش عناصر آن به صورت های زیر هستند.