

ساختمان داده‌ها

دکتر امین گلزاری اسکوئی

a.golzari@azaruniv.ac.ir

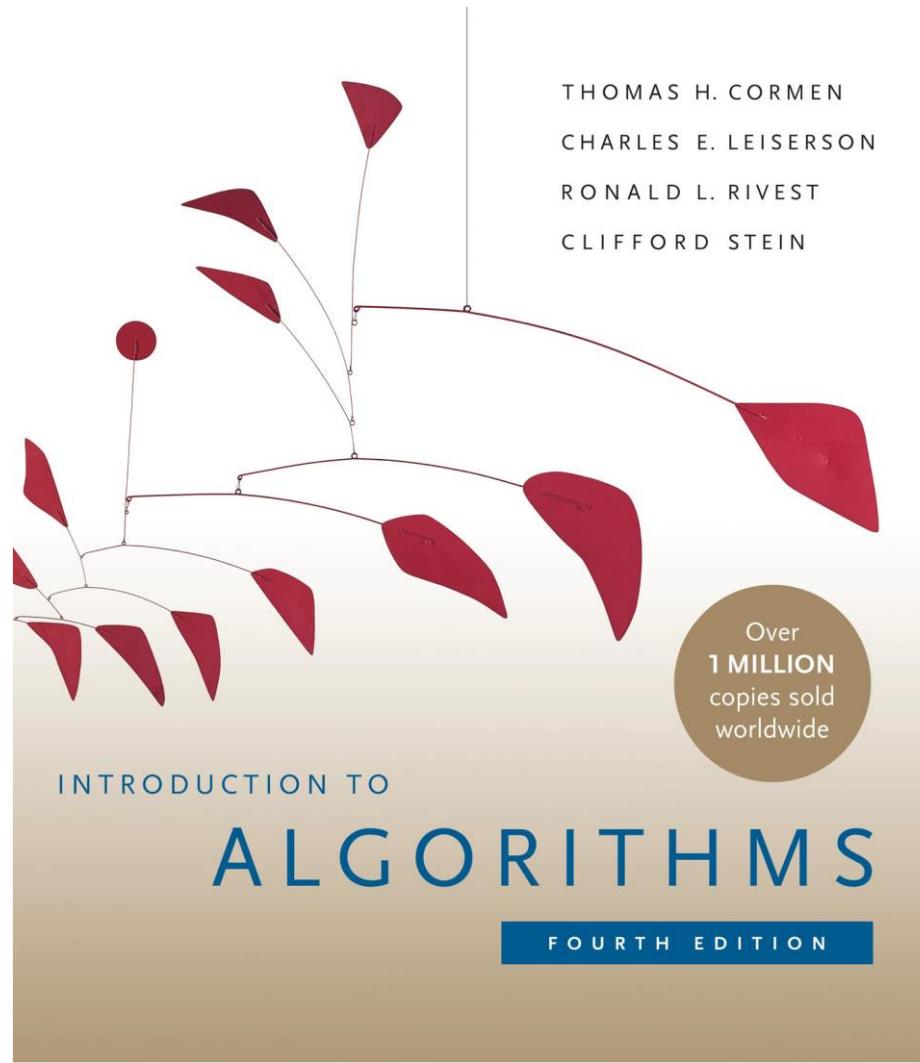
a.golzari@tabrizu.ac.ir

<https://github.com/Amin-Golzari-Oskouei>

دانشگاه شهید مدنی آذربایجان
پاییز ۱۴۰۲



منابع



❖ کورمن

❖ دکتر قدسی

❖ شیرا فکن

نمرات و آزمون‌ها



در صورت غایب بودن در امتحان‌های میان ترم و امتحان‌های کوییز (یا در یکی از امتحان‌ها)، درصد امتحان پایانی برای شخص مورد نظر، براساس میزان درصد امتحان مذکور، افزایش خواهد یافت.

اکیدا تاکید می‌کنم در امتحانات میان ترم و کوییزها شرکت کنید.



دستیاران



امیرحسین میری

دانشجو سال سوم مهندسی کامپیوتر



محمد ظهیری

دانشجو سال سوم مهندسی کامپیوتر

سر فصل

- ❖ مرتبه اجرایی (پیچیدگی اجرایی)
- ❖ توابع بازگشتی
- ❖ آرایه
- ❖ صف و پیشنه
- ❖ لیست پیوندی
- ❖ درخت
- ❖ گراف
- ❖ مرتبسازی
- ❖ درهمسازی

فصل ۱

مرتبه اجرایی (پیچیدگی اجرایی)

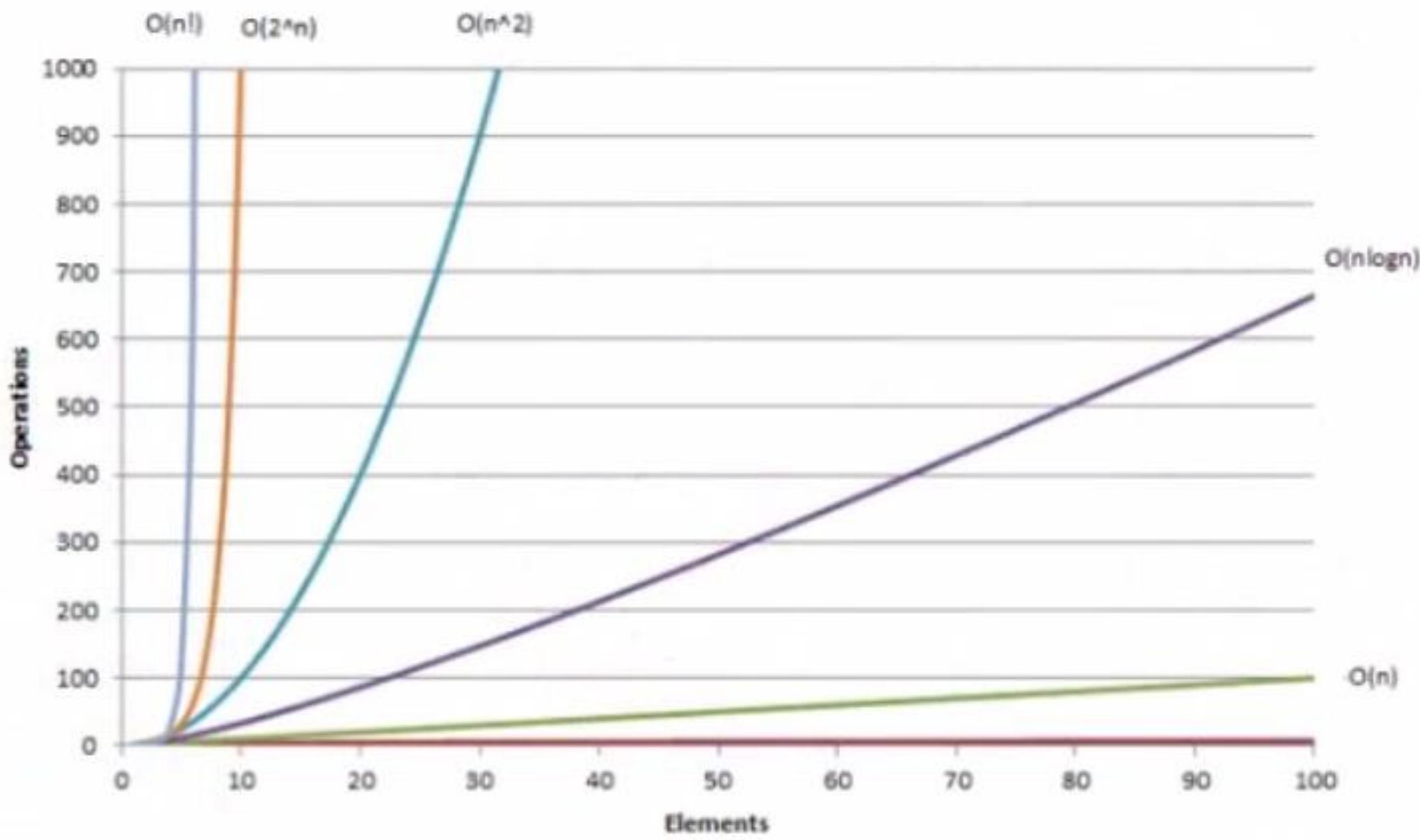
مطالب این فصل

تعریف پیچیدگی و زمان اجرا الگوریتمها

یادگیری نمادهای مختلف برای بررسی پیچیدگی الگوریتمها

پیچیدگی اجرایی

پیچیدگی یک الگوریتم، تابعی است که مدت زمان اجرای استفاده شده توسط الگوریتم را بر حسب تعداد داده‌های ورودی n اندازه می‌گیرد.



notation	name
$O(1)$	constant
$O(n)$	linear
$O(\log n)$	logarithmic
$O(n^2)$	quadratic
$O(n^c)$	polynomial
$O(c^n)$	exponential
$O(n!)$	factorial

مقایسه نرخ رشد توابع مختلف

بزرگ‌ترین مسئله‌ی قابل حل در چند دقیقه				Nexx رشد
۲۰۰۰ به بعد	دهه‌ی ۹۰	دهه‌ی ۸۰	دهه‌ی ۷۰	
هر	هر	هر	هر	1
هر	هر	هر	هر	$\log N$
چند میلیارد	چند صد میلیون	چند ده میلیون	چند میلیون	N
چند صد میلیون	چند ده میلیون	چند میلیون	چند صد هزار	$N \log N$
چند ده هزار	چند هزار	هزار	چند صد	N^2
چند هزار	هزار	چند صد	صد	N^3
سی	بیست و چند	بیست و چند	بیست	2^N

مقایسه نرخ رشد توابع مختلف

n	$\log n$	n	$n \log n$	n^2	n^3	2^n
8	3	8	24	64	512	256
16	4	16	64	256	4,096	65,536
32	5	32	160	1,024	32,768	4,294,967,296
64	6	64	384	4,096	262,144	1.84×10^{19}
128	7	128	896	16,384	2,097,152	3.40×10^{38}
256	8	256	2,048	65,536	16,777,216	1.15×10^{77}
512	9	512	4,608	262,144	134,217,728	1.34×10^{154}

مرتبه اجرایی توابع چند جمله‌ای

در صورتی که داشته باشیم:

$$f(n) = n^m + n^{m-1} + \dots + n^2 + n + c \Rightarrow f(n) = O(n^m)$$

مثال:

$$f(n) = 5n^2 - 3n + 4 \Rightarrow O(n^2)$$

$$O(1) < O(\lg n) < O(n) < O(n \lg n) < O(n^2) < O(2^n) < O(n!) < O(n^n)$$

$$\log_2^n = \lg n$$

نمادهای پیچیدگی اجرایی

اوی بزرگ

عبارت $f(n) \in O(g(n))$ یعنی: برای تابع پیچیدگی مفروض $g(n)$ به مجموعه ای از توابع اشاره دارد که برای آنها ثابت‌های C و n_0 وجود دارند، بطوریکه برای همه $n \geq n_0$ داریم:

امگا بزرگ

عبارت $f(n) \in \Omega(g(n))$ یعنی: برای تابع پیچیدگی مفروض $g(n)$ به مجموعه ای از توابع اشاره دارد که برای آنها ثابت‌های C و n_0 وجود دارند، بطوریکه برای همه $n \geq n_0$ داریم:

تنا

عبارت $f(n) \in \Theta(g(n))$ یعنی: $f(n) \in O(g(n))$ و $f(n) \in \Omega(g(n))$

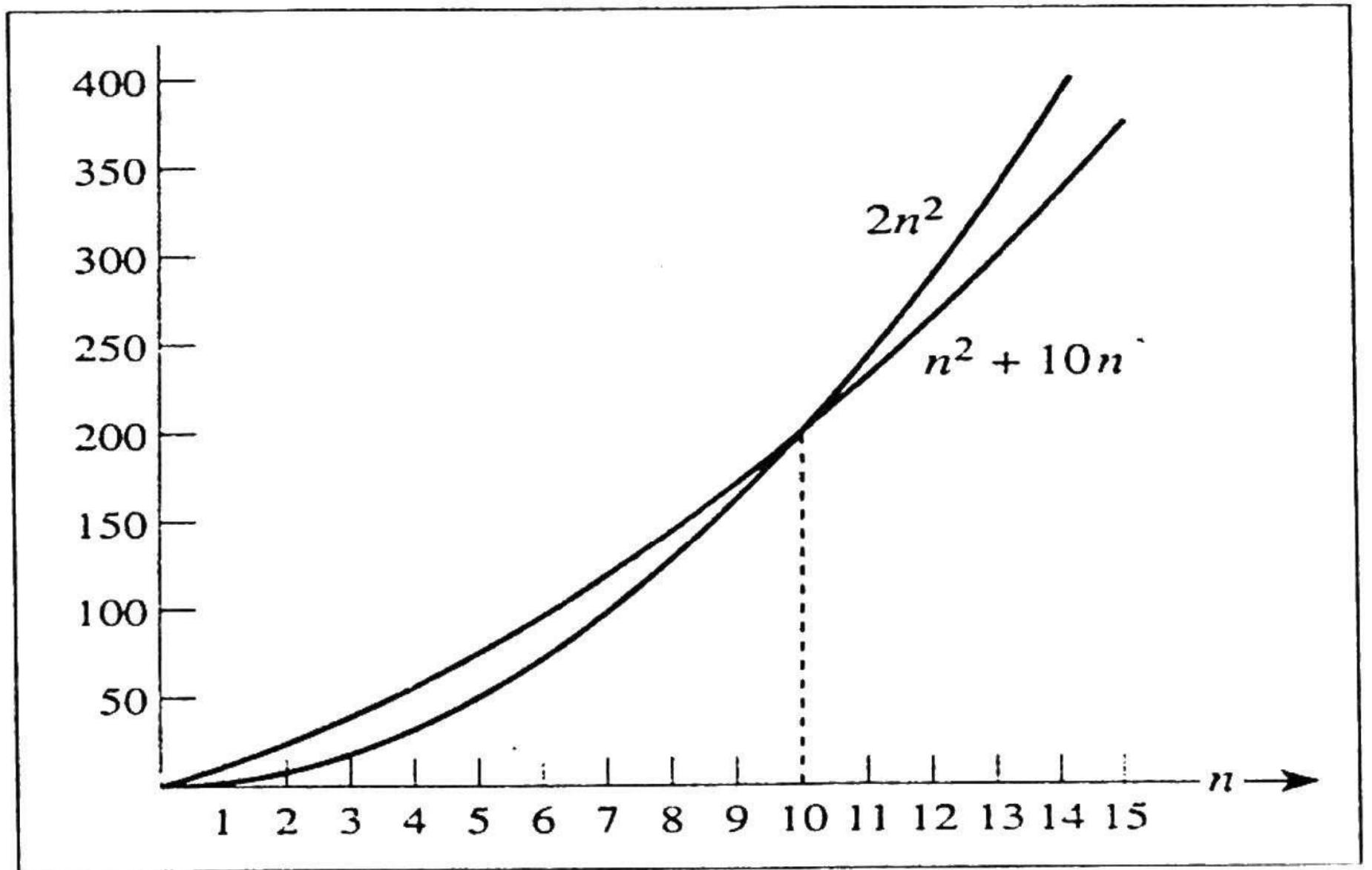
مثال

نشان دهید: $n^2 + 10n \in O(n^2)$

$$n^2 + 10n \leq cn^2$$

$$n^2 + 10n \leq 2n^2$$

$$n^2 + 10n = 2n^2 \Rightarrow 10n = n^2 \Rightarrow n = 10$$



مثال

$$\frac{n(n-1)}{2} \in O(n^2)$$

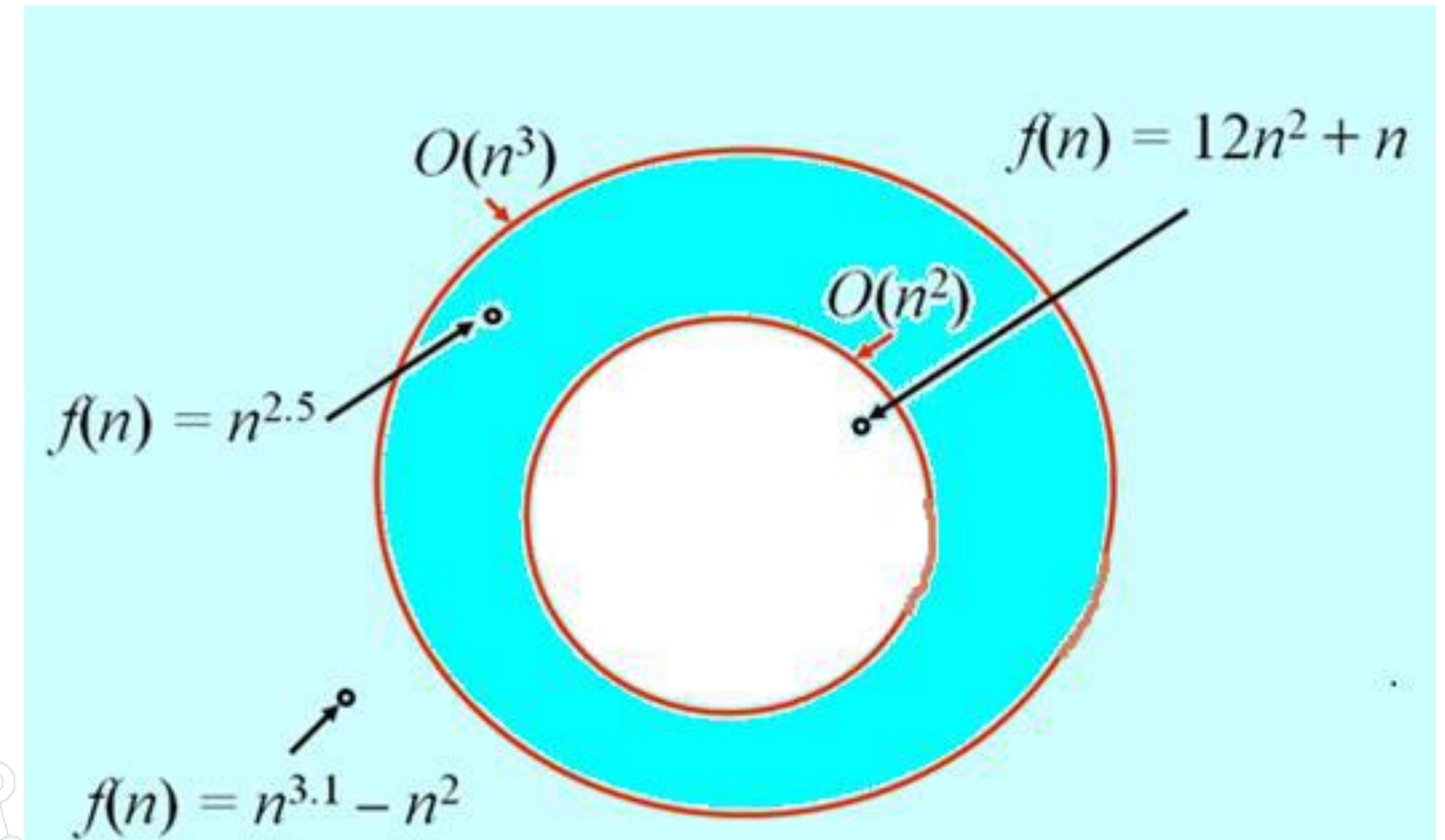
$$\frac{n(n-1)}{2} \leq cn^2$$

$$\frac{n(n-1)}{2} \leq \frac{n^2}{2}$$

$$\frac{n(n-1)}{2} = \frac{n^2}{2} \Rightarrow n = 0$$

$$n_0 = 0, c = \frac{1}{2} \quad \text{يعني :}$$

مثال



مثال

$$\frac{n(n-1)}{2} \geq cn^2$$

$$\frac{n^2}{2} - \frac{n}{2} \geq \frac{1}{4}n^2$$

$$\frac{n^2}{2} - \frac{n}{2} = \frac{n^2}{4} \Rightarrow n = 2$$

نشان دهید : $\frac{n(n-1)}{2} \in \Omega(n^2)$: حل :

$$n_0 = 2, c = \frac{1}{4}$$
 يعني:

مثال

نشان دهید که :
جواب:

در مثال های قبل دیدیم که $\frac{n(n-1)}{2} \in \Omega(n^2)$ و $\frac{n(n-1)}{2} \in O(n^2)$ بنابراین :

$$\frac{n(n-1)}{2} \in \theta(n^2)$$

خواص توابع رشد

۱) بازتابی : $f(n) = O(f(n)), \quad f(n) = \Omega(f(n)), \quad f(n) = \theta(f(n))$

۲) تراگذاری .
مثلا برای تنا :
$$\begin{cases} f(n) = \theta(g(n)) \\ g(n) = \theta(h(n)) \end{cases} \Rightarrow f(n) = \theta(h(n))$$

۳) تنا خاصیت تقارن دارد: $f(n) = \theta(g(n)) \Leftrightarrow g(n) = \theta(f(n))$

۴) خاصیت تقارن ترانهاده دارد: $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

خواص توابع رشد

$$\begin{cases} f(n) \in O(g(n)) \\ h(n) \in O(k(n)) \end{cases} \Rightarrow f(n).h(n) \in O(g(n).k(n))$$

$$\begin{cases} f(n) \in O(g(n)) \\ h(n) \in O(k(n)) \end{cases} \Rightarrow f(n) + h(n) \in O(\max(g(n), k(n)))$$

پیچیدگی دستور if

به دستور if زیر توجه کنید:

```
if(cond)
    block1
```

```
else
    block2
```

زمان اجرا در بدترین حالت برابر است با:

$$\text{Max(time(block1) , time(block2))}$$

مرتبه اجرایی حلقه های ساده

```
for (i=a ; i<=b ; i=i+k )  
    S;
```

```
for (i=b ; i>=a ; i=i-k )  
    S;
```

حلقه زیر را در نظر بگیرید: $(a \leq b)$

$$\frac{b-a+1}{k} \quad \text{تعداد تکرار:}$$

اگر این مقدار اعشاری بود، حد بالای آن را در نظر بگیرید.

منظور از S ، توالی از دستورها (sequence of statement) است.

مثال

تعداد تکرار دستور داخل حلقه را مشخص کنید.

```
for ( i=1 ; i<=n ; i=i+1 )  
    S;
```

جواب:

$$\frac{n - 1 + 1}{1} = n \quad \text{تعداد تکرار :}$$

مرتبه اجرایی : $O(n)$

مثال

تعداد تکرار دستور داخل حلقه را مشخص کنید.

```
for ( i=3 ; i<=n ; i=i+2 )  
    s;
```

جواب:

$$\frac{n - 3 + 1}{2} = \frac{n}{2} - 1$$

مرتبه اجرایی : **O(n)**

مثال

```
for ( i=9 ; i<3n+4 ; i=i+5 )  
s;
```

تعداد تکرار دستور داخل حلقه را مشخص کنید.

جواب:

$$\frac{3n + 4 - 9}{5} = \frac{3n}{5} - 1 \quad \text{تعداد تکرار:}$$

مرتبه اجرایی: $O(n)$

مرتبه ثابت

مرتبه اجرایی دستور $x=x+1$ را مشخص کنید.

```
i=n;  
while(i>1)  
{  
    i=i % 2;  
    x=x+1;  
}
```

$O(1)$ جواب:

مرتبه لگاریتمی

حلقه زیر را در نظر بگیرید:

```
for ( i=a ; i<=b ; i=i*k )  
    s;
```

```
for ( i=b ; i>=a ; i=i/k )  
    s;
```

$$\log_k^b - \log_k^a + 1 \quad \text{تعداد تکرار:}$$

اگر این مقدار صحیح نبود، حد بالای آن را در نظر بگیرید.

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=8 ; i=i*2 )  
    s;
```

$$\log_2^8 - \log_2^1 + 1 = 4$$

جواب:

مثال

```
for ( i=27 ; i<=n ; i=i*3 )  
    s;
```

تعداد تکرار را مشخص کنید.

جواب:

$$\log_3^n - \log_3^{27} + 1 = \log_3^n - 2 \Rightarrow O(\log_3^n)$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=n ; i>=16 ; i=i/4 )  
    s;
```

جواب:

$$\log_4^n - \log_4^{16} + 1 = \log_4^n - 1 \Rightarrow O(\log_4^n)$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=n ; i>=1; i=i - i/3 )
```

```
    s;
```

```
for ( i=n ; i>=1; i= i / (3/2) )
```

```
    s;
```

جواب:

$$i = \frac{2}{3}i = \frac{i}{\frac{3}{2}}$$

می توان نوشت :

بنابراین تعداد تکرار برابر است با :

$$\log_{3/2}^n - \log_{3/2}^1 + 1 = \log_{3/2}^n + 1$$



حلقه های تودر تو

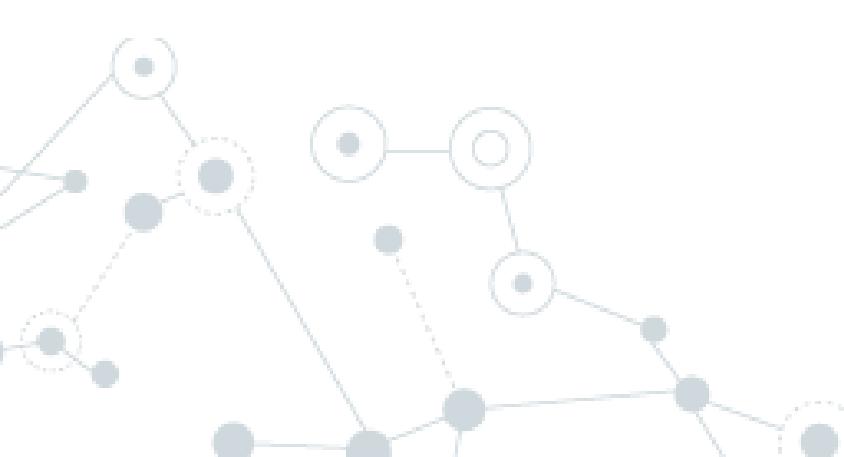
تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i++ )  
    for ( j=1 ; j<=n ; j++ )  
        S;
```

حل:

تعداد تکرار هر حلقه: n

تعداد تکرار دستور S : n^2



مثال

تعداد تکرار را مشخص کنید.

```
for ( i=2 ; i<=n ; i=i+4 )  
    for ( j=n ; j>3 ; j=j-2 )  
        s;
```

جواب:

$$\frac{n-2+1}{4} \times \frac{n-3}{2} \Rightarrow O(n^2)$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i=i*2 )
    for ( j=1 ; j<=n ; j++ )
        s;
```

جواب:

$$(\lg n + 1) \times n \Rightarrow O(n \lg n)$$

حلقه های پشت سر هم

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i++) {  
    s;  
}  
  
for ( j=1 ; j<=m ; j++) {  
    s;  
}
```

$O(\max(n, m))$ جواب:

$O(n + m)$ يا

ترکیب حلقه های تودر تو و پشت سرهم

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i++) {  
    for ( j=1 ; j<=n ; j++) {  
        s;  
    }  
}  
for ( k=1 ; j<=n ; k++) {  
    s;  
}
```

$$O(\max(n^2, n)) = O(n^2)$$

جواب:

حلقه های تودر تو وابسته

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i++ )  
  for ( j=1 ; j<=i ; j++ )  
    S;
```

i	1	2	3	...	n
تعداد تکرار	1	2	3	...	n

جواب:

$$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2} \Rightarrow O(n^2)$$

مجموع تعداد تکرار :

$$\sum_{i=1}^n \sum_{j=1}^i 1 = \sum_{i=1}^n i = \frac{n(n+1)}{2}$$

روش دوم:

مثال

تعداد تکرار را مشخص کنید.

```
for ( i=1 ; i<=n ; i=i*2 )  
    for ( j=1 ; j<=i ; j++ )  
        S;
```

جواب:

i	1	2	4	...	n
تعداد تکرار	1	2	4	...	n

مجموع تعداد تکرار :

$$1 + 2 + 4 + \dots + n = 2^0 + 2^1 + 2^2 + \dots + 2^{\log_2^n} = \frac{2^{\lg n + 1} - 1}{2 - 1} = 2^{\lg n} \times 2 - 1 = 2n - 1 \Rightarrow O(n)$$

$$\alpha^0 + \alpha^1 + \alpha^2 + \alpha^3 + \dots + \alpha^k = \frac{\alpha^{k+1} - 1}{\alpha - 1}$$

مثال

تعداد تکرار را مشخص کنید.

```
for ( k=1 ; k<=n ; k++ )  
    for ( i=1 ; i<=n ; i=i*2 )  
        for ( j=1 ; j<=i ; j++ )  
            S;
```

جواب:

در مثال قبل دیدیم که دو حلقه داخلی $(2n-1)$ مرتبه تکرار می شود. چون حلقه اول n مرتبه تکرار می شود، تعداد کل تکرار برابر است با:

$$n(2n - 1) \Rightarrow O(n^2)$$

مثال

```
for ( i=1 ; i<=n ; i++ )  
    for ( j=1 ; j<=n ; j=j+i )  
        S;
```

تعداد تکرار را مشخص کنید.

جواب:

i	1	2	3	...	n
تعداد تکرار	n	n/2	n/3	...	1

مجموع تعداد تکرار :

$$n + \frac{n}{2} + \frac{n}{3} + \dots + 1 = n\left(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}\right) \approx n \ln n \Rightarrow O(n \lg n)$$

مثال

```
for ( i=1 ; i<=n ; i=i*3 )  
    for ( j=i ; j<=n ; j++ )  
        S;
```

تعداد تکرار را مشخص کنید.

جواب:

i	1	3	9	...	n
تعداد تکرار	n-1+1	n-3+1	n-9+1	...	n-n+1

مجموع تعداد تکرار :

$$(\log_3^n + 1)(n + 1) - (1 + 3 + 9 + \dots + n) = (\log_3^n + 1)(n + 1) - \frac{3n - 1}{2} \Rightarrow O(n \lg n)$$

$$1 + 3 + 9 + \dots + n = 3^0 + 3^1 + 3^2 + \dots + 3^{\log_3^n} = \frac{3^{\log_3^n + 1} - 1}{3 - 1} = \frac{3n - 1}{2}$$

مثال

```
for ( i=1 ; i<=n ; i++ )  
    for ( j=1 ; j<=log(i) ; j++ )  
        S;
```

تعداد تکرار را مشخص کنید.

جواب:

i	1	2	...	n
تعداد تکرار	$\log(1)$	$\log(2)$...	$\log(n)$

مجموع تعداد تکرار :

$$\log(1) + \log(2) + \dots + \log(n) = \log(1 \times 2 \times \dots \times n) = \log(n!)$$

$$\log(n!) = \theta(n \lg n)$$

تغییر مقدار نهایی شمارنده

```
for ( i=1 ; i<=n ; i++ )  
    for ( j=1 ; j<=n ; j++ )  
    {   s;  
        n=n-1; }
```

تعداد تکرار را مشخص کنید.

جواب:

i	1	2	3	...
تعداد تکرار	$n/2$	$n/4$	$n/8$...

$$\frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \dots = n\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots\right) = n \times \frac{1/2}{1 - 1/2} = n \Rightarrow O(n)$$

مجموع تعداد تکرار :

پیچیدگی در حالت فراخوانی تابع

مثال : با فرض اینکه پیچیدگی زمانی تابع $f(k)$ برابر $O(k)$ باشد، پیچیدگی را مشخص کنید.

```
for ( i=1 ; i<=n ; i++ )  
    f(n);
```

$n \times n \Rightarrow O(n^2)$ جواب:

```
for ( i=1 ; i<=n ; i++ )  
    f(i);
```

مثال :

$$f(1) + f(2) + \dots + f(n) = 1 + 2 + \dots + n = \frac{n(n+1)}{2} \Rightarrow O(n^2)$$

جواب:

مسئله خرس قطبی

در یک زمستان سرد ، خرس قطبی n قطعه گوشت (دقیقا به اندازه های ۱، ۲، تا n) را در غاری ذخیره کرده است. او هر روز یکی از این قطعه گوشت ها را به صورت تصادفی انتخاب می کند. اگر اندازه ی گوشت عدد فردی بود، آن را کاملا می خورد. اگر زوج بود، آن را دقیقا نصف می کند، یک نصف آن را می خورد و نصف دیگر را مجددا در غار قرار می دهد. اگر گوشتی موجود نباشد، خرس می میرد. با این الگوریتم، برای n های خیلی بزرگ، روزهای باقیمانده از عمر خرس چگونه خواهد بود؟

حل:

با فرض $n=4$ ، قطعه های گوشت برابر ۱ و ۲ و ۳ و ۴ می باشند. با هر ترتیبی که بخورد بعد از ۷ روز گوشت ها تمام می شود. برای $n=8$ جواب ۱۵ است.

روش دوم :

$$n + \frac{n}{2} + \frac{n}{4} + \dots = n\left(1 + \frac{1}{2} + \frac{1}{4} + \dots\right) \approx n \times \frac{1}{1 - \frac{1}{2}} \approx 2n = O(n)$$

تابع $\lg^* n$

خروجی این تابع برابر است با تعداد دفعاتی که اگر از n لگاریتم گرفته شود، حاصل برابر یک(یا کوچکتر) خواهد شد. بنابراین رشد این تابع بسیار کند است.

$$\lg^* 4 = 2$$

$$4 \rightarrow 2 \rightarrow 1$$

$$\lg^* 16 = 3$$

$$16 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

$$\lg^* 65536 = 4$$

$$65536 \rightarrow 16 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

تعریف نماد اوی کوچک و امگای کوچک

عبارت $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$ برقرار است در صورتی که : $f(n) = o(g(n))$

عبارت $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$ برقرار است در صورتی که : $f(n) = \omega(g(n))$

$$\lim_{n \rightarrow \infty} \frac{n^2/2}{n^3} = \lim_{n \rightarrow \infty} \frac{1}{2n} = 0$$

مثال: رابطه $\frac{n^2}{2} \in o(n^3)$ برقرار است:



PYTHON TIME

مثال

```
def max(lst):
    x = lst[0]
    for i in range(1, len(lst)):
        if lst[i] > x:
            x = lst[i]
    return x
```

❖ تعداد تکرار: $n-1$

❖ پیچیدگی اجرایی: $O(n)$

مثال

```
def f(x, y, n):
    if x > y:
        for i in range(n):
            print(i)
    else:
        for i in range(n):
            for j in range(n):
                print(i, j)
```

$$\max(n, n^2) = n^2 \text{ پیچیدگی اجرایی:}$$

مثال

```
def unique(S):
    for j in range(len(S)):
        for k in range(j+1, len(S)):
            if S[j] == S[k]:
                return False
    return True
```

$$1+2\dots+(n-1)=n^2$$

مقایسه دو الگوریتم

(وش کند)

```
def avg(S):
    n = len(S)
    A = [0] * n
    for j in range(n):
        total = 0
        for i in range(j + 1):
            total += S[i]
        A[j] = total / (j+1)
    return A
```

(وش سریع)

```
def avg2(S):
    n = len(S)
    A = [0] * n
    total = 0
    for j in range(n):
        total += S[j]
        A[j] = total / (j+1)
    return A
```

پیچیدگی اجرایی: n^2

پیچیدگی اجرایی: n

مقایسه دو الگوریتم

(وش سریع)

```
def disjoint2(A, B, C):
    for a in A:
        for b in B:
            if a == b :
                for c in C:
                    if a == c:
                        return False
    return True
```

```
def disjoint1(A, B, C):
    for a in A:
        for b in B:
            for c in C:
                if a == b == c:
                    return False
    return True
```

پیچیدگی اجرایی: n^3

پیچیدگی اجرایی: n^2

مقایسه دو الگوریتم

(وش کند)

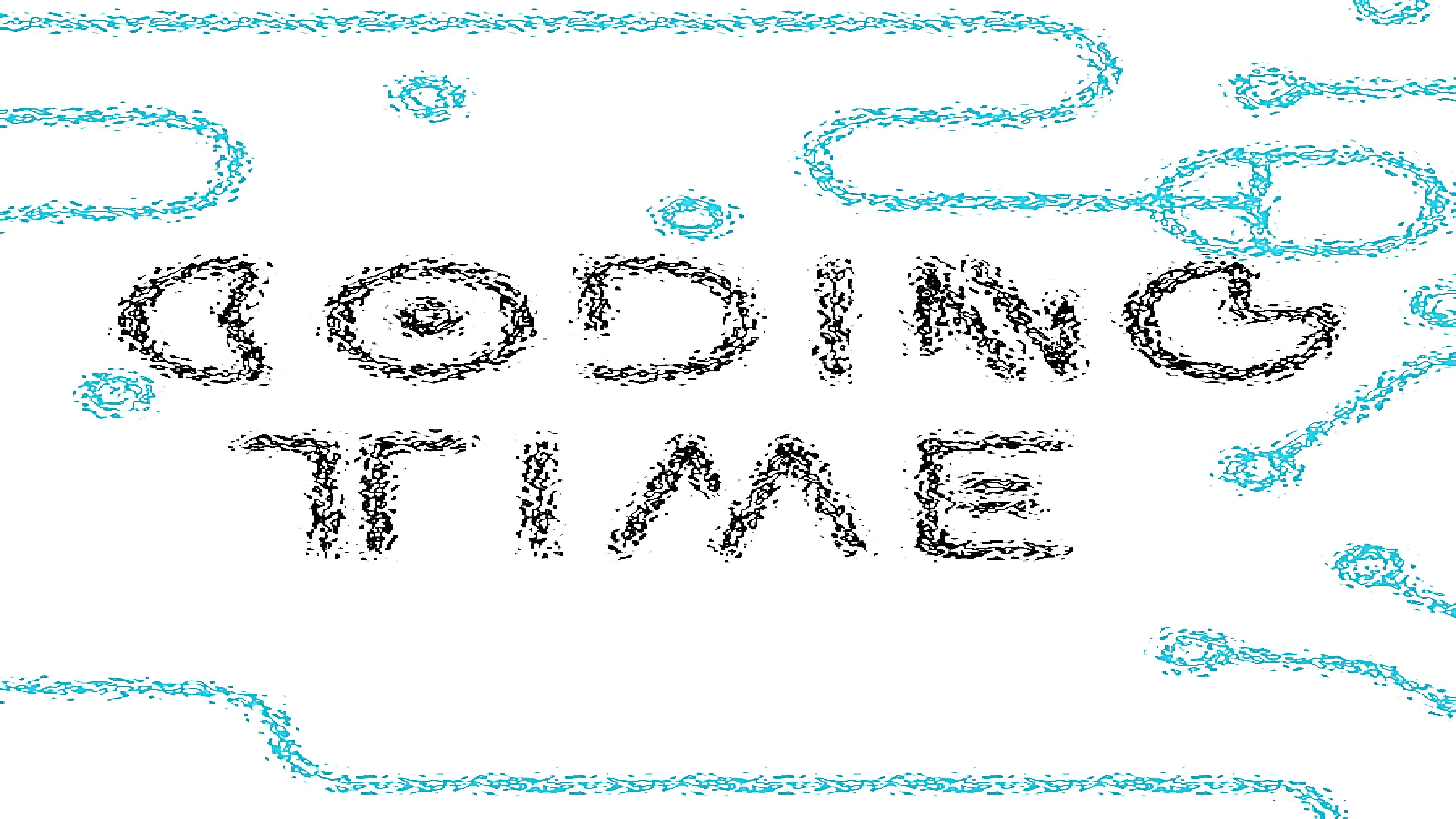
```
def r1(a):
    n = len(a)
    x = [None]*n
    for i in range(n):
        x[n-1-i] = a[i]
    return (x)
```

(وش سریع)

```
def r2(a):
    i=0
    n = len(a)
    while i < n//2:
        a[i], a[n-1-i] = a[n-1-i], a[i]
        i+=1
    return (a)
```

تعداد تکرار: n

تعداد تکرار: $\frac{n}{2}$





پرسنل

سوال

a.golzari@azaruniv.ac.ir

a.golzari@tabrizu.ac.ir

<https://github.com/Amin-Golzari-Oskouei>