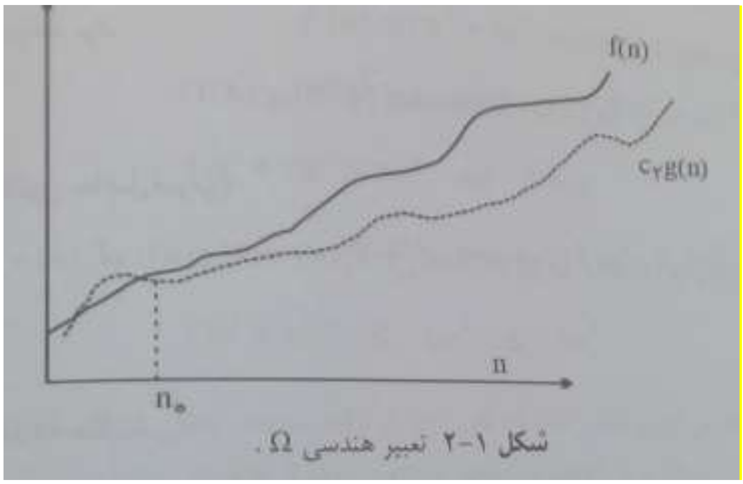
جلسه دوم

تعریف: Ω اومگای بزرگ (بررسی زمان اجرا در بهترین حالت)

گوییم $\mathrm{f(n)} = \Omega(\mathrm{g(n)})$ اگر عدد طبیعی n_0 وعدد حقیقی مثبت c_2 وجود داشته باشند بطوریکه برای هر $\mathrm{n} \geq n_0$، $|\mathrm{f(n)}| \geq c_2|\mathrm{g(n)}|$ (شکل 1-2 را ببینید).



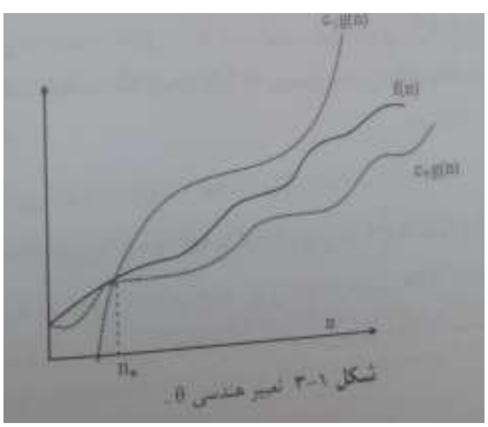
شکل ۱-۲ تعبیر هندسی Ω .

مثال 13: نشان دهید $f(n)=3n^3 + 2n^2 = \Omega(n^3)$.

تعریف: θ (بررسی حالت میانگین زمان اجرا)

گوییم $\mathrm{f(n)} = \theta(\mathrm{g(n)})$ اگر عدد طبیعی n_0 واعداد حقیقی مثبت c_1 و c_2 وجود داشته باشند بطوریکه برای هر $\mathrm{n} \geq n_0$، $c_2 g(n) \leq f(n) \leq c_1 g(n)$

(شکل 1-3 را ببینید).



شکل ۱-۳ تعبیر هندسی θ .

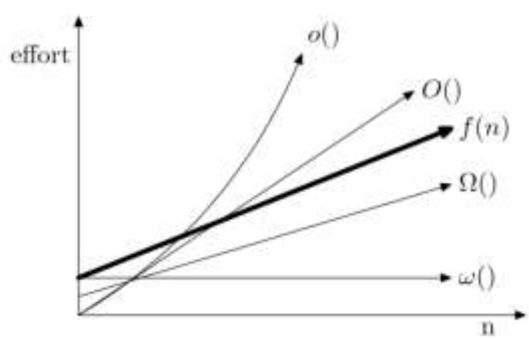
نتیجه 1: $f(n) = \theta(g(n))$ اگر و فقط اگر $f(n) = O(g(n))$ و $f(n) = \Omega(g(n))$.

مثال 14: نشان دهید $f(n)=3n^3 + 2n^2 = \theta(n^3)$

تعریف: (o اوی کوچک) گوییم $f(n) = o(g(n))$ اگر

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

مثال 15: $3n^3 + 2n^2 = o(n^4)$.



Big vs Small o Notation

- Big-O notation says that one function is asymptotically no more than another
  - Think of it as ≤.
- Small-o notation says that one function is asymptotically less than another
  - This of it as <
- A small-o bound is always also a big-O bound (not vice-versa)
  - Just as if x < y then x ≤ y (x ≤ y does not mean that x < y)

تعریف: (ω اومگای کوچک) گوییم $f(n) = \omega(g(n))$ اگر

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty.$$

مثال 16: مثال 13: $3n^3 + 2n^2 = \omega(n^2)$.

مقایسه خواص O، Ω، θ ، o و ω

۱ـ خاصیت تقارنی : این خاصیت را فقط θ دارد :

$$f(n) = \theta(g(n)) \Leftrightarrow g(n) = \theta(f(n))$$

۲ـ خاصیت بازتابی :

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

خاصیت بازتابی را o و ω ندارند.

۳ـ خاصیت ترانهاده تقارنی :

$$f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$$

$$f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$$

۴ـ خاصیت تعدی :

$$f(n)=O(g(n)) \;,\; g(n)=O(h(n)) \Rightarrow f(n)=O(h(n))$$
$$f(n)=\Omega(g(n)) \;,\; g(n)=\Omega(h(n)) \Rightarrow f(n)=\Omega(h(n))$$
$$f(n)=\theta(g(n)) \;,\; g(n)=\theta(h(n)) \Rightarrow f(n)=\theta(h(n))$$
$$f(n)=o(g(n)) \;,\; g(n)=o(h(n)) \Rightarrow f(n)=o(h(n))$$
$$f(n)=\omega(g(n)) \;,\; g(n)=\omega(h(n)) \Rightarrow f(n)=\omega(h(n))$$

Important summations and techniques

- **Constant Series:** For integers *a* and *b*, $a \le b$,

$$\sum_{i=a}^{b} 1 = b - a + 1$$

- **Linear Series (Arithmetic Series):** For $n \ge 0$,

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

- **Quadratic Series:** For $n \ge 0$,

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

- **Cubic Series:** **For $n \geq 0$,**

$$\sum_{i=1}^{n} i^3 = 1^3 + 2^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$

- **Geometric Series:** **For real $x \neq 1$,**

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1}$$

For |x| < 1,

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

- **Linear-Geometric Series:** **For $n \geq 0$, real $c \neq 1$,**

$$\sum_{i=1}^{n} ic^i = c + 2c^2 + \cdots + nc^n = \frac{-(n+1)c^{n+1} + nc^{n+2} + c}{(c-1)^2}$$

- **Harmonic Series:** **n^{th} harmonic number, $n \in I^+$,**

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

$$= \sum_{k=1}^{n} \frac{1}{k} = \ln(n) + O(1)$$

- **Telescoping Series:**

$$\sum_{k=1}^{n} a_k - a_{k-1} = a_n - a_0$$

- **Differentiating Series:** For |x| < 1,

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

- **Approximation by integrals:**
  - For monotonically increasing f(n)

$$\int_{m-1}^{n} f(x)dx \le \sum_{k=m}^{n} f(k) \le \int_{m}^{n+1} f(x)dx$$

  - For monotonically decreasing f(n)

$$\int_{m}^{n+1} f(x)dx \le \sum_{k=m}^{n} f(k) \le \int_{m-1}^{n} f(x)dx$$

- ***n*th harmonic number**

$$\sum_{k=1}^{n} \frac{1}{k} \geq \int_{1}^{n+1} \frac{dx}{x} = \ln(n+1)$$

$$\sum_{k=2}^{n} \frac{1}{k} \leq \int_{1}^{n} \frac{dx}{x} = \ln n$$

$$\Rightarrow \sum_{k=1}^{n} \frac{1}{k} \leq \ln n + 1$$

چند مثال از نمادهای مجانبی

مثال 17: مرتبه زمانی شبه کد زیر کدام است؟

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

مثال 18: کدام گزینه صحیح است؟

۱) $n^{\frac{1}{10}} \in \Omega(\log n)$

۲) $\log(n!) \in O(\log n)$

۳) $8n^2 + 3n - 4 \in O(n \log n)$

۴) $1^2 + 2^n + \ldots + n^n \in O(n^n)$

مثال 19: مرتبه زمانی قطعه کد زیر کدام است؟

```
i := 2
while  i <= n  do
begin
   i := i^2
   x := x+1
end
```

مثال 20: فرض کنید آرایه A[1,..n] داده شده است و هدف مرتب سازی آن به صورت صعودی به کمک الگوریتم مرتب سازی حبابی است.

```
Algorithm Bubble _Sort (A, n)
(a) for i = ۱ to  n-۱  do
      {
(b)       for  j  =  n  downto   i + ۱ do
          {
(c)           if  A[j-۱] > A[j]   then
              {
(d)               temp  =  A[j-۱]
(e)               A[j-۱]  =  A[j]
(f)               A[j]  =  temp
              }
          }
```

الگوریتم فوق نخست کوچک ترین عنصر را پس از n-1 مقایسه در جای اول و عنصر کوچک بعدی را پس از n-2 مقایسه در جای دوم و در نهایت عنصر n-1 ام را با یک مقایسه در جای n-1 قرار می دهد.

الف: تحلیل بدترین پیچیدگی زمانی:

این حالت زمانی اتفاق می افتد که آرایه بصورت نزولی مرتب شده باشد. تعداد اجرای دستورات الگوریتم در جدول زیر خلاصه شده است:

```
Algorithm Bubble _Sort (A, n)
(a) for i = ١ to  n-١ do
       {
(b)        for j = n downto  i+١ do
           {
(c)           if A[j-١] > A[j]  then
              {
(d)              temp = A[j-١]
(e)              A[j-١] = A[j]
(f)              A[j] = temp
              }
       }
```

دستورالعمل	تعداد دفعات اجرا
(a)	$n-١$
(b)	$(n-١)+(n-٢)+...+٢+١$
(c)	$(n-١)+(n-٢)+...+٢+١$
(d)	$(n-١)+(n-٢)+...+٢+١$
(e)	$(n-١)+(n-٢)+...+٢+١$
(f)	$(n-١)+(n-٢)+...+٢+١$

ب: تحلیل بهترین پیچیدگی زمانی:

این حالت زمانی اتفاق می افتد که آرایه بصورت صعودی مرتب شده باشد. تعداد اجرای دستورات الگوریتم در جدول زیر خلاصه شده است:

دستورالعمل	تعداد دفعات اجرا
(a)	$n-۱$
(b)	$(n-۱)+(n-۲)+...+۲+۱$
(c)	$(n-۱)+(n-۲)+...+۲+۱$
(d)	۰
(e)	۰
(f)	۰

ج: تحلیل پیچیدگی در حالت میانگین:

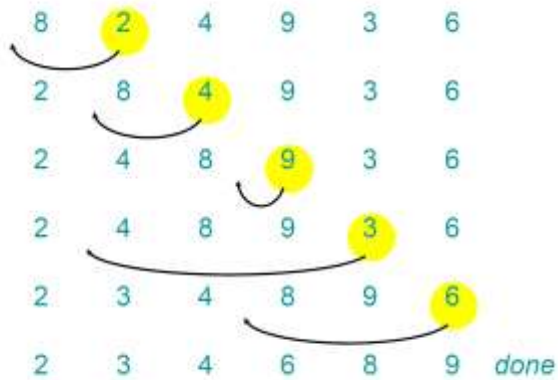
طبق **نتیجه 1** پیچیدگی در این حالت $\theta(n^2)$ است.

مثال 21: تحلیل الگوریتم مرتب سازی درجی

فرض کنید آرایه A[1,..n] داده شده است و هدف مرتب سازی آن به صورت صعودی به کمک الگوریتم مرتب سازی درجی است.

```
INSERTION-SORT (A, n)      ⊳ A[1 . . n]
   for j ← 2 to n
        do key ← A[j]
           i ← j − 1
           while i > 0 and A[i] > key
                do A[i+1] ← A[i]
                   i ← i − 1
           A[i+1] = key
```

با فرض اینکه عناصر 1 تا j-1 مرتب شده اند عنصرj پس از انکه کلیه عناصر سمت چپ و بزرگ تر از آن شیفت داده شدند در جای مناسب درج می شود.



در این الگوریتم پس از از i تکرار، i عنصر اول آرایه مرتب شده هستند. فرض کنید هدف محاسبه تعداد دفعات اجرای دستور مقایسه ای است.

الف: بدترین حالت: بدترین حالت زمانی است که حلقه داخلی همواره اجرا شود. مجموع تعداد اجرای حلقه داخلی $2 + 3 + \cdots n = \frac{n(n+1)}{2} - 1 = O(n^2)$ است.

ب: بهترین حالت: این حالت زمانی اتفاق می افتد که حلقه داخلی اجرا نشود. جمع تعداد اجرای حلقه داخلی $1 + 1 + \cdots + 1 = \Omega(n)$ است.

ج: حالت میانگین: در این حالت لازم است در هر مرحله نصف آرایه تا جایی که مرتب شده مقایسه شود. در نتیجه تعداد اجرای حلقه داخلی $O(n^2)$ است.

Difference

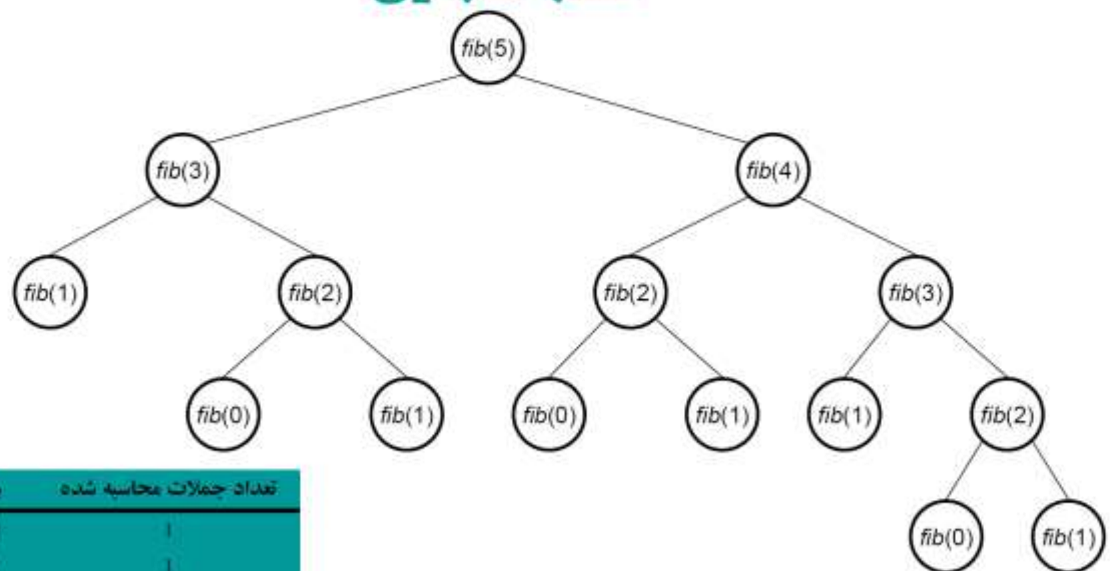
In insertion sort elements are bubbled into the sorted section, while in bubble sort the maximums are bubbled out of the unsorted section.

برخی الگوریتم ها ممکن است بازگشتی باشند. مثال های زیر را ببینید.

مثال 22: تعداد فراخوانی های برنامه زیر را محاسبه کنید:

```
Algorithm Fib (n)
if n ≤ ۱ then  return (n)
    else
        return (Fib (n-۱) + Fib (n-۲))
```

عدم کارآیی



n	تعداد جملات محاسبه شده
0	1
1	1
2	3
3	5
4	9
5	15
6	25

- علت ناکارآیی: محاسبات تکراری
- مثلا در این مثال $fib(2)$ سه بار محاسبه شده است.

فرض کنید $T(n)$ تعداد فراخوانی تابع Fib باشد، پس داریم:

$T(n) = T(n-1) + T(n-2) + 1$

$T(0) = 1$

$T(1)$=1

- هر بار که n به اندازه ۲ واحد افزایش می یابد، تعداد جملات محاسبه شده بیش از ۲ برابر افزایش می یابد، یعنی:
  - $T(n) > 2 * T(n - 2) > 2^{n/2}$ when $n \geq 2$
  - $$\begin{aligned} T(n) &> 2 * T(n - 2) \\ &> 2 * 2 * T(n - 4) \\ &> 2 * 2 * 2 * T(n-6) \\ &\dots \\ &> \underbrace{2 * 2 * \dots * 2}_{n/2 \text{ بار}} * T(0) = 2^{n/2} \end{aligned}$$
- اثبات بوسیله استقراء

- **پایه استقراء:**

$$T(2) = 3 > 2 = 2^{2/2}$$
$$\mathrm{T}(3) = 5 > 2.83 \approx 2^{3/2}$$

- **فرض استقراء:**

$$T(m) > 2^{m/2} \qquad , 2 \leq m < n$$

- **گام استقراء:**

$$\begin{aligned} T(n) &= T(n - 1) + T(n - 2) + 1 \\ &> 2^{(n-1)/2} + 2^{(n-2)/2} + 1 \qquad \text{طبق فرض استقراء} \\ &> 2^{(n-2)/2} + 2^{(n-2)/2} = 2 * 2^{(n-2)/2} = 2^{n/2} \end{aligned}$$

مثال 23: تعداد فراخوانی تابع زیر را حساب کنید:

```
int fact(int n)
{
    if (n == 1) return 1;
else return n * fact(n - 1);
}
```

داریم

$$T(n) = T(n-1) + 1.$$

پس

$$T(n) = T(n-1) + 1 = T(n-2) + 2 = \cdots = 1 + (n-1) = O(n).$$

مثال 24: مساله برج هانوی با سه میله A,B,C و n حلقه را در نظر بگیرید و فرض کنید هدف انتقال حلقه ها از A به B فقط از طریق C است، یعنی نمی توان مستقیم حلقه ها را A به B منتقل کرد. اگر n حلقه داشته باشیم تعداد جابجایی لازم را بدست آورید.

گامهای الگوریتم:

1: ابتدا n-1 حلقه را طبق شرایط بالا از میله A به میله C انتقال می دهیم.

2: حلقه آخر را از میله A به حلقه B انتقال می دهیم.

3: n-1 حلقه میله C را با رعایت شرایط بالا به میله B منتقل می کنیم.

بنابراین تعداد کل فراخوانی های تابع برابر است با

$$T(1) = 1$$

$$T(n) = 2T(n-1) + 1 = \cdots = 2 + 2(2 + 2 + \cdots 2^{n-1}) = 2^n - 1.$$