دانشگاه شهید باهنر کرمان

یک: تحلیل الگوریتم ها

ساختمان داده ها و الگوریتم

مدرس: دکتر نجمه منصوری

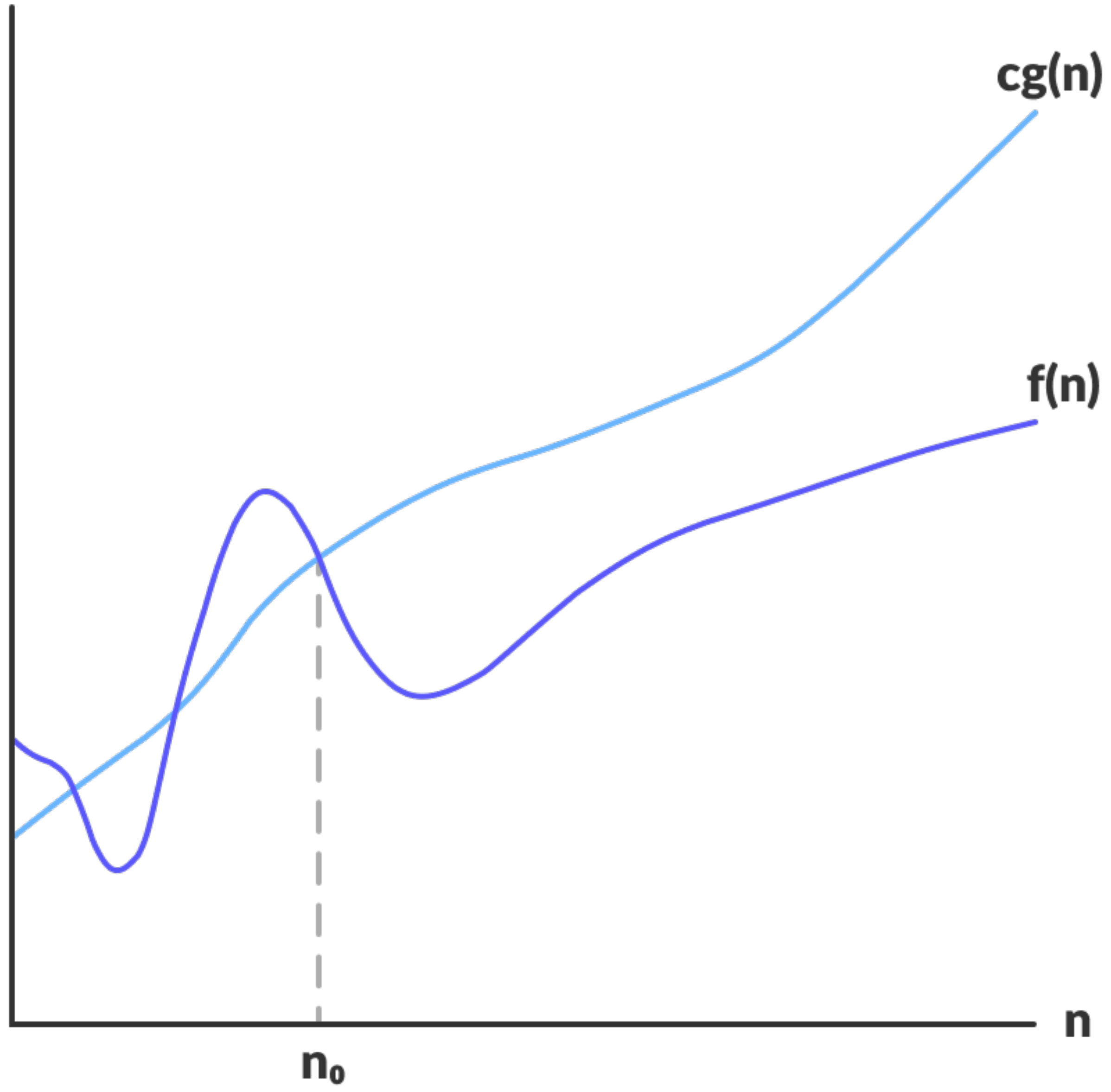
نگارنده: سجاد هاشمیان

نماد O

برای g(n) داده شده، O(g(n)) را مجموعه توابع زیر تعریف می‌کنیم:

$$O(g(n)) = \{f(n) | \exists c > 0 \ \ and \ \ n_o \ \ .s.t \ \forall n > n_o, 0 \leq f(n) \leq c \times g(n)\}$$

در اینجا می‌گوییم g(n) کران بالا مجانبی برای f(n) است.



$f(n) = O(g(n))$

مثال

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

حل.

$$\frac{n(n-1)}{2} \;?\; c \times n^2$$

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

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

یعنی: $n_0 = 0, c = \frac{1}{2}$

مثال

$$5n^2 \in O(n^2)$$

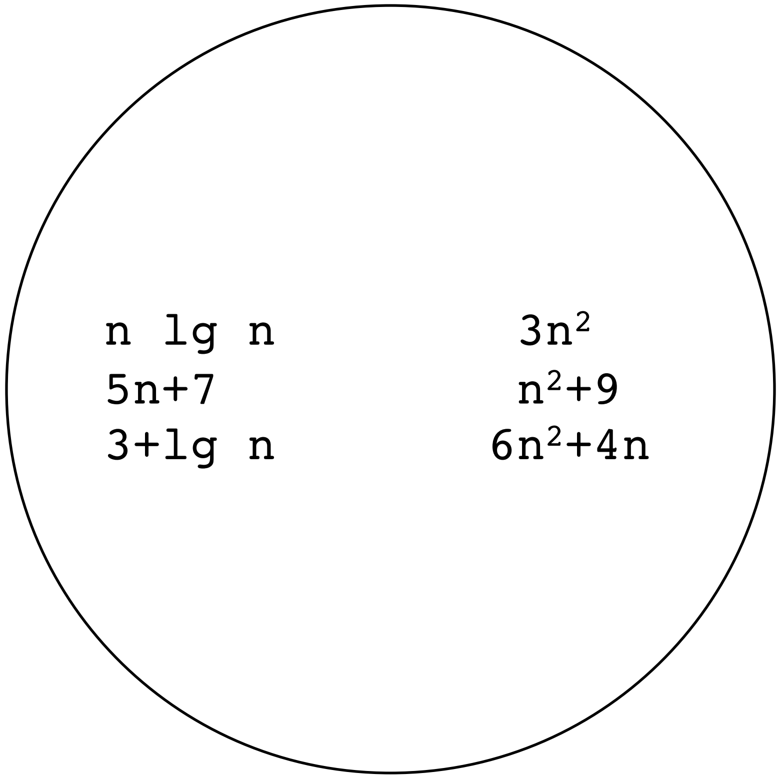
$$5n^2 \leq 5 \times n^2 \Rightarrow N = 0, C = 5$$

$$n^2 \in O(n^2 + 10n)$$

$$N = 10, C = 1$$

$$n \in O(n^2)$$

$$N = 1, C = 1$$



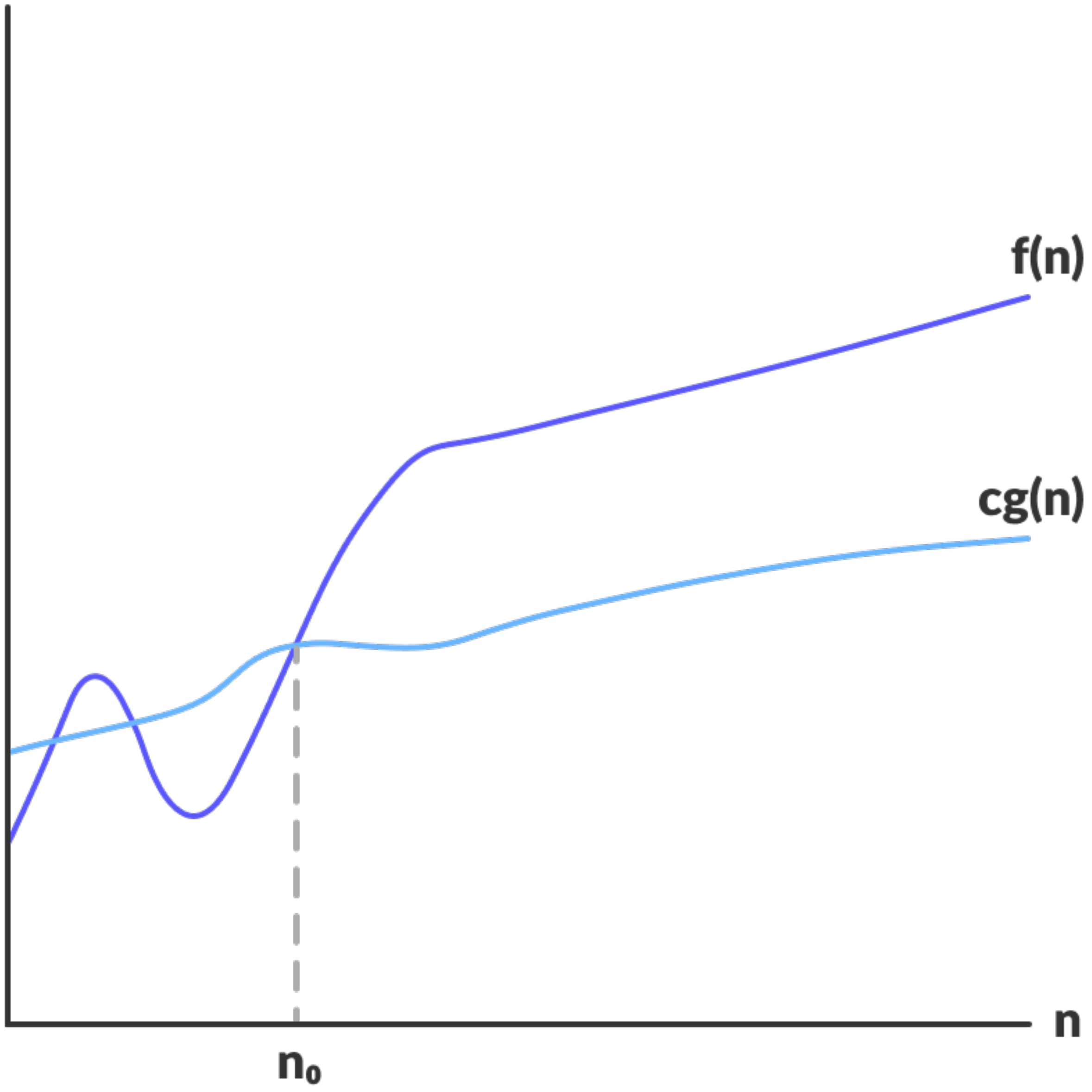
$O(n^2)$

نماد Ω

برای g(n) داده شده، Ω(g(n)) را مجموعه توابع زیر تعریف می‌کنیم:

$$\Omega(g(n)) = \{f(n) | \exists c > 0 \ \ and \ \ n_o \ \ .s.t \ \forall n > n_o, 0 \leq c \times g(n) \leq f(n)\}$$

در اینجا می‌گوییم g(n) کران پایین مجانبی برای f(n) است.



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

مثال

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

حل.

$$\frac{n(n-1)}{2} \; ? \; c \times n^2$$

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

$$\frac{n(n-1)}{2} = \frac{n^2}{4} \implies n = 2$$

یعنی: $n_0 = 2, c = \frac{1}{4}$

مثال

$$5n^2 \in \Omega(n^2)$$

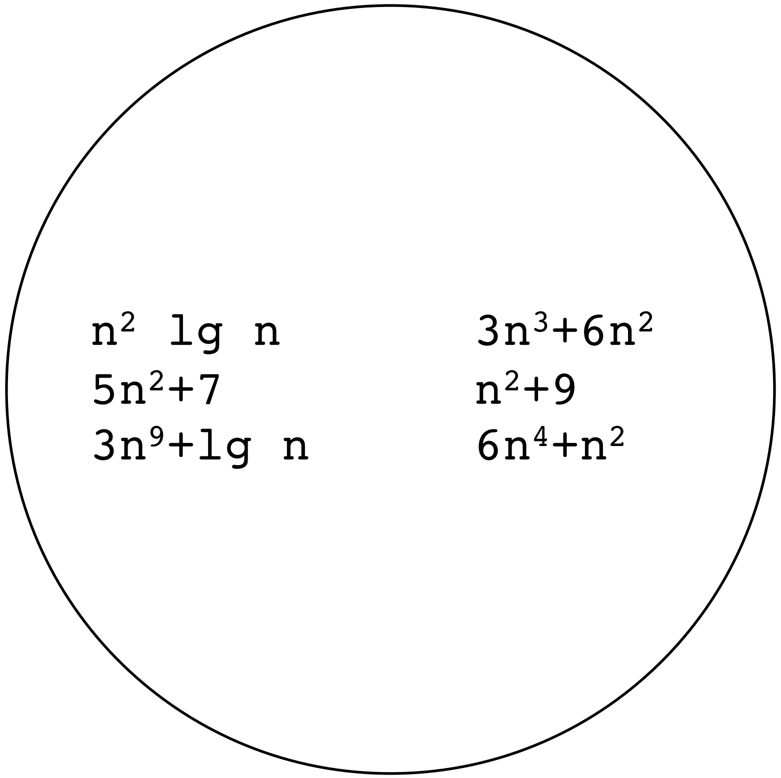
$$5n^2 \geq n^2 \Rightarrow N = 0, C = 1$$

$$n^2 + 10n \in \Omega(n^2)$$

$$N = 0, C = 1$$

$$n^3 \in \Omega(n^2)$$

$$N = 1, C = 1$$



$\Omega(n^2)$

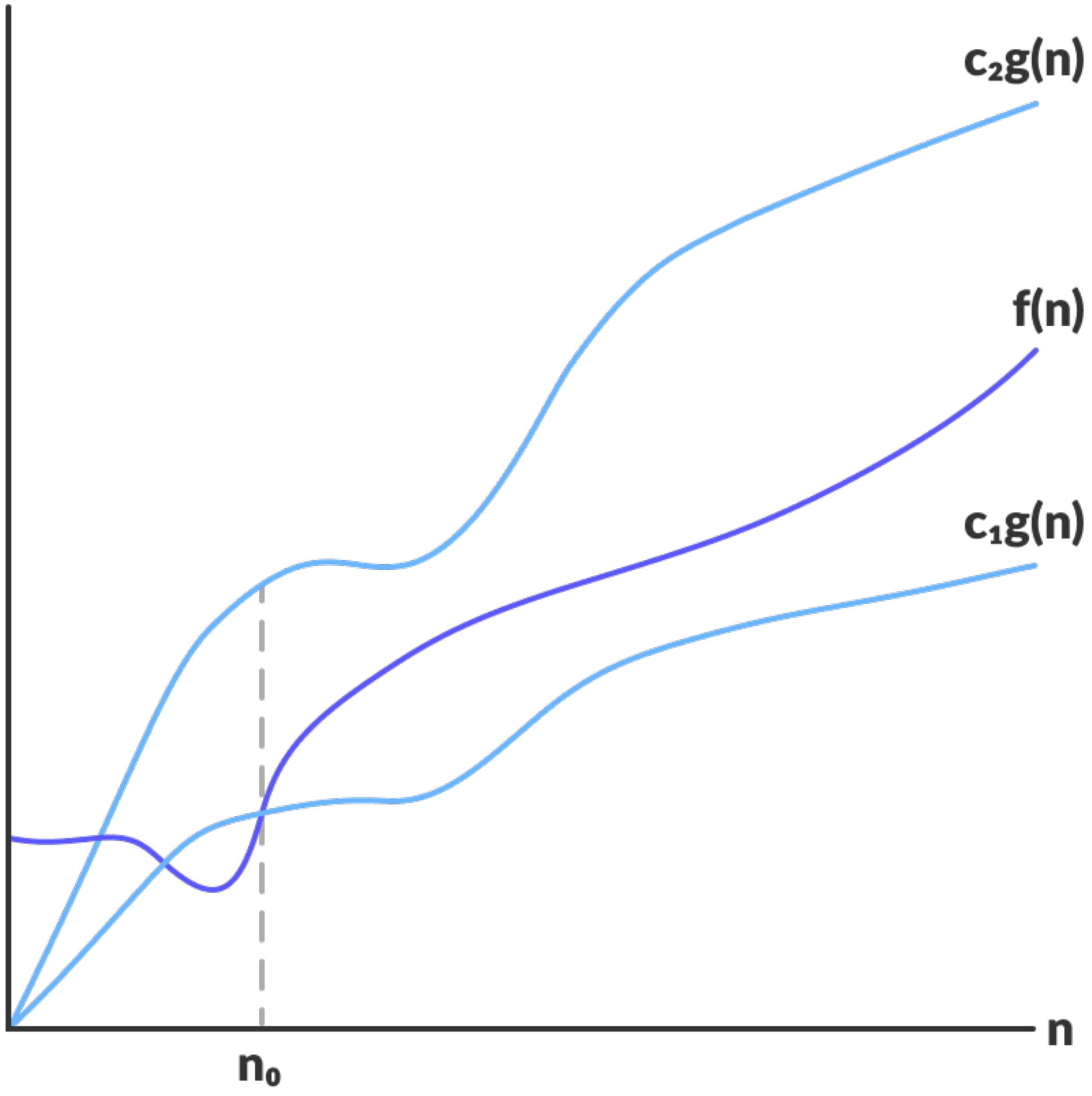
نماد θ

برای g(n) داده شده، θ(g(n)) را مجموعه توابع زیر تعریف می‌کنیم:

$$\theta(g(n)) = \{f(n) | \exists c > 0 \ \ and \ \ n_o \ \ .s.t \ \ \forall n > n_o, c_2 \times g(n) \leq f(n) \leq c_1 \times g(n)\}$$

این به آن معنی است که برای مقادیر بزرگ n درجه رشد دو تابع f و g با هم برابر است. در اینجا می‌گوییم تابع g(n) کران بالا بسته مجانبی برای f(n) است.

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



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

مثال

$۱۰۰n^{۲}+۵n-۴\neq\theta(n^{۳})$

حل. باید نشان دهیم هیچ مقدار مثبت برای C_1 و C_2 و یک مقدار n_0 پیدا نمی شود که رابطه‌ی

$$C_{۱}n^{۳} \leq ۱۰۰n^{۲}+۵n-۴ \leq C_{۲}n^{۳}$$

برای همه مقادیر $n>n_0$ برقرار باشد.

به وضوح به ازای هر مقدار $C_1>0$ و برای nهای بزرگ داریم :

$$C_{۱}n^{۳} \not< ۱۰۰n^{۲}+۵n-۴$$

مثال

$$۱۰۰n^۲+۵n-۴=\theta(n^۲)$$

حل. کافیست در نظر بگیرید $C_1=1$ و $C_2=200$ آنگاه برای همه مقادیر $n>1$ داریم:

$$C_۱n^۲ \le ۱۰۰n^۲+۵n-۴ \le C_۲n^۲$$

نکته) در واقع می‌توان نشان داد که هر چند جمله ای از مرتبه دقیق جمله با بزرگترین توانش است، یعنی:

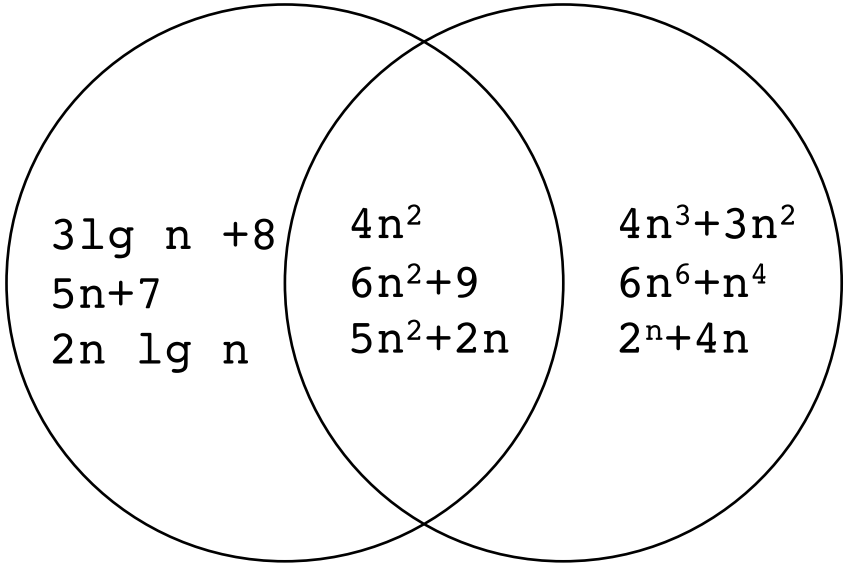
$$a_k n^k + a_{k-1} n^{k-1} + ... = \theta(n^k)$$

نماد θ

تابع θ عطف دو تابع O و Ω است:

$$f(n) = \theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$

یا به عبارتی شرط لازم و کافی برای $f(n)=\theta(g(n))$ آن است که $f(n)=O(g(n))$ و $f(n)=\Omega(g(n))$.



$$\theta(n^2) = \Omega(n^2) \cap O(n^2)$$

مثال

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

حل. در مثال های قبل دو حکم زیر را دیدیم، بنابراین حکم اثبات می‌شود.

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

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

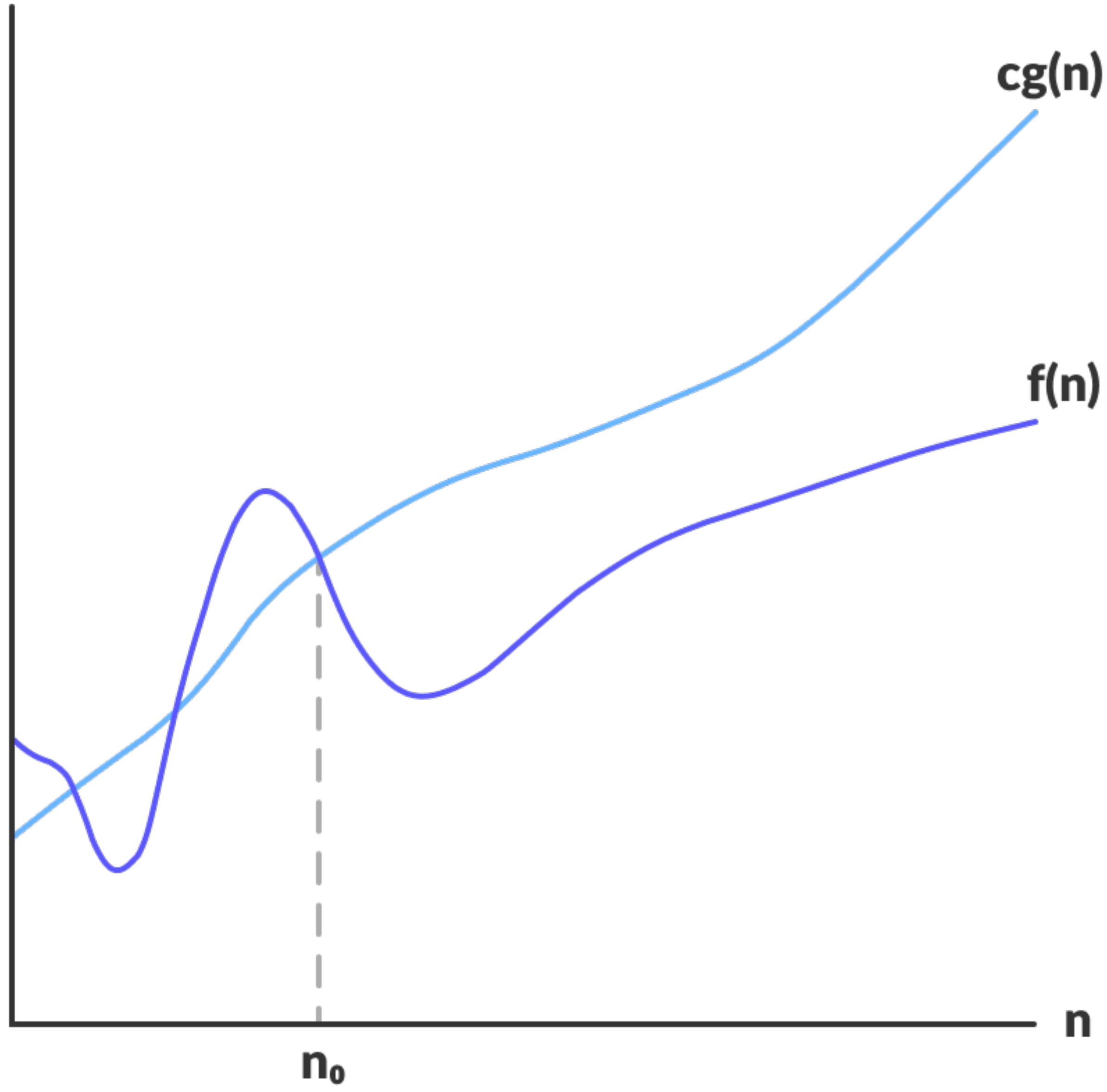
نماد o

برای g(n) داده شده، o(g(n)) را مجموعه توابع زیر تعریف می‌کنیم:

$$o(g(n)) = \{f(n) | \exists c > 0 \ \ and \ \ n_o \ \ .s.t \ \forall n > n_o, 0 < f(n) < c \times g(n)\}$$

این نماد به O شبیه است با این تفاوت که درجه رشد f(n) نمی‌تواند برابر با g(n) باشد و الزاما از آن کوچکتر است، به عبارتی داریم:

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



$$f(n) = o(g(n))$$

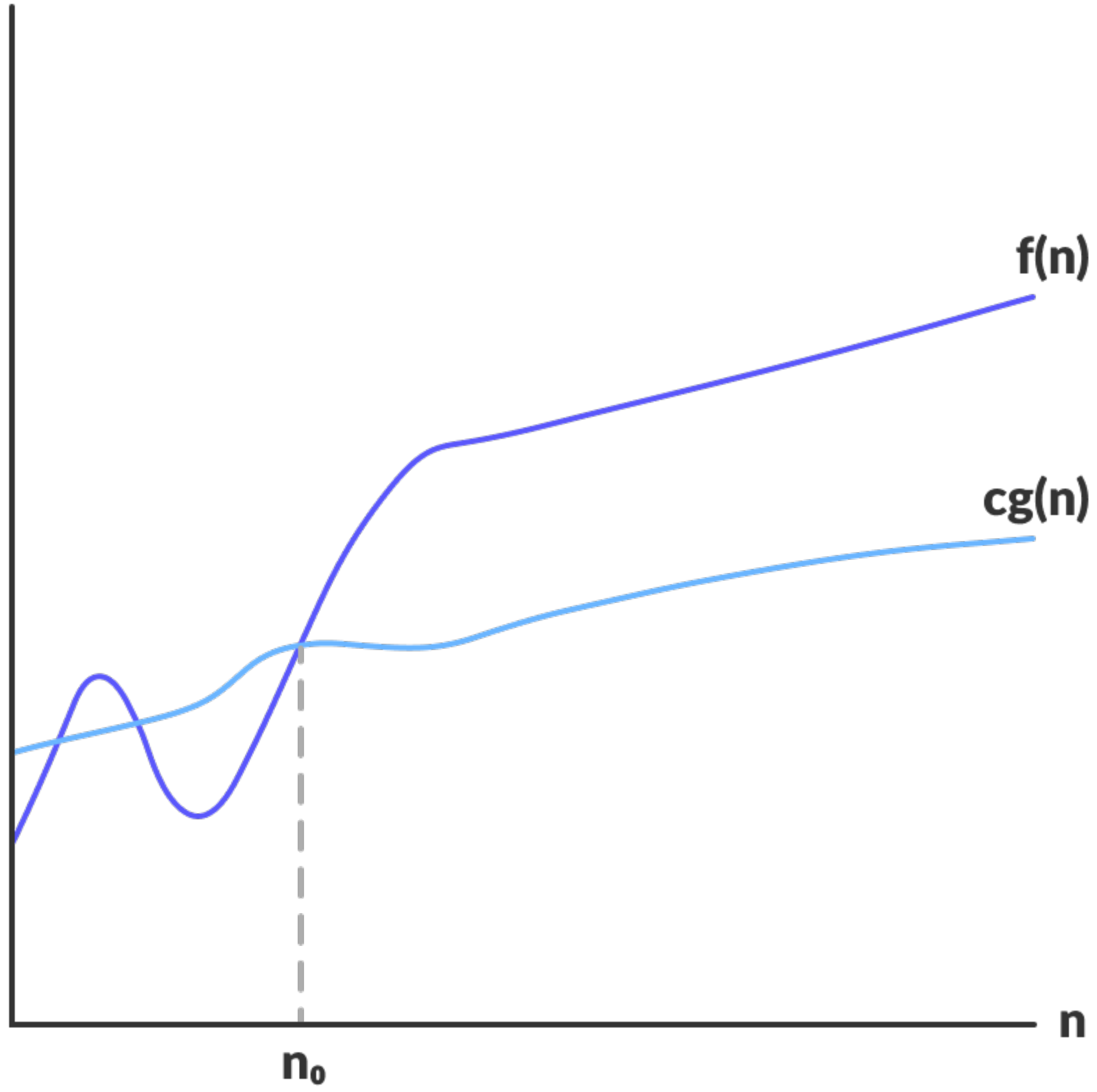
نماد ω

برای g(n) داده شده، ω(g(n)) را مجموعه توابع زیر تعریف می‌کنیم:

$$\omega(g(n)) = \{f(n) | \exists c > 0 \ \ and \ \ n_o \ \ .s.t \ \forall n > n_o, 0 \leq c \times g(n) < f(n)\}$$

این نماد به Ω شبیه است با این تفاوت که درجه رشد f(n) نمی‌تواند برابر با g(n) باشد و الزاما از آن بزرگتر است، به عبارتی داریم:

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



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

خلاصه کلام

خودمانی بگوییم (هرچند نا دقیق از نظر ریاضی)، اگر G درجه رشد تابع g و F درجه رشد تابع f باشد، آنگاه داریم:

$$f = \Theta(g) \iff F = G$$

$$f = O(g) \iff F \leq G$$

$$f = o(g) \iff F < G$$

$$f = \Omega(g) \iff F \geq G$$

$$f = \omega(g) \iff F > G$$

خواص توابع رشد

خاصیت تراگذاری (Transitive)

$$f(n) = \Theta(g(n)), g(n) = \Theta(h(n)) \implies f(n) = \Theta(h(n))$$

$$f(n) = O(g(n)), g(n) = O(h(n)) \implies f(n) = O(h(n))$$

$$f(n) = o(g(n))\ ,\ g(n) = o(h(n)) \implies f(n) = o(h(n))$$

$$f(n) = \Omega(g(n)), g(n) = \Omega(h(n)) \implies f(n) = \Omega(h(n))$$

$$f(n) = \omega(g(n)), g(n) = \omega(h(n)) \implies f(n) = \omega(h(n))$$

خواص توابع رشد

خاصیت بازتابی (Reflexive)

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

$$f(n) = O(f(n))$$

$$f(n) = \Omega(f(n))$$

$$f(n) \neq o(f(n))$$

$$f(n) \neq \omega(f(n))$$

خواص توابع رشد

خاصیت تقارن (Symmetry)

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

خاصیت تقارن ترانهاده (Transpose symmetry)

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

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

استفاده از حد

همانگونه که در تعریف o و ω دیدیم، برای تعیین مرتبه توابع با استفاده از حد داریم:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \implies f(n) \in \Theta(g(n))$$

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

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \implies f(n) \in \omega(g(n))$$

مثال

$$\frac{n^2}{2} \in o(n^3)$$

حل:

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

روش‌های تحلیل الگوریتم‌ها

الگوریتم های ترتیبی:

- یک یا تعداد ثابتی از عبارت های ساده: $\theta(1)$

```
var x=5
var y=2
var a=x
var b=x+a
x=5*x+y
```

روش‌های تحلیل الگوریتم‌ها

فرض کنید T(n) زمان یک برنامه باشد که به k قسمت به صورت زیر تقسیم شده:

$$T_1(n) = O(f_1(n))$$

$$T_2(n) = O(f_2(n))$$

$$\cdots$$

$$T_k(n) = O(f_k(n))$$

در آن صورت داریم:

$$T(n) = \sum_{i=1}^{k} T_i(n) = O(\max_{1 \le i \le k}(f_i(n)))$$

روش‌های تحلیل الگوریتم‌ها

$g(n)$ بار تکرار برنامه ای با زمان اجرای $S(n)=O(f(n))$:

$$T(n)=g(n)S(n)=O(g(n)f(n))$$

ساختار `if` که قسمت های `else` و `then` آن به ترتیب زمان های $T_1(n)=O(f_1(n))$ و $T_2(n)=O(f_2(n))$ داشته باشند:

$$T(n)=\max\{T_1(n),T_2(n)\}=O(\max\{f_1(n),f_2(n)\})$$

مرتبه اجرایی حلقه ها

برای a و b دلخواه، حلقه های for زیر را در نظر بگیرید:

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

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

آنگاه تعداد تکرار های آنها برابر خواهد بود با:

$$\lceil \frac{|b-a|+1}{k} \rceil$$

مثال

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

تعداد تکرار: $\left\lceil \frac{|n-1|+1}{1} \right\rceil = n$

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

تعداد تکرار: $\left\lceil \frac{|n-3|+1}{2} \right\rceil = \frac{n}{2} - 1$

در نتیجه مرتبه اجرایی هر دو حلقه از O(n) است.

مرتبه اجرایی حلقه ها (لگاریتمی)

برای a و b دلخواه، حلقه های for زیر را در نظر بگیرید:

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

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

آنگاه تعداد تکرار های آنها برابر خواهد بود با:

$$\lceil \log_k^b - \log_k^a + 1 \rceil$$

مثال

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

تعداد تکرار: $\lceil \log_2^8 - \log_2^1 + 1 \rceil = 4$

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

تعداد تکرار: $\lceil \log_3^n - \log^2 7_3 + 1 \rceil$

$$= \log_3^n - 2 \in O(\log_3^n)$$

مثال

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

تعداد تکرار: $\lceil \log_2^8 - \log_2^1 + 1 \rceil = 4$

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

تعداد تکرار: $\lceil \log_3^n - \log_3^{27} + 1 \rceil$

$$= \log_3^n - 2 \in O(\log_3^n)$$

مثال

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

تعداد تکرار:

$$\lceil \log_4^n - \log_4^{16} + 1 \rceil = \log_4^n - 1 \in O(\log_4^n)$$

مثال

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

می‌توان نوشت: $i - i/3 = \frac{2}{3}i = \frac{i}{\frac{3}{2}}$ آنگاه داریم:

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

بنابراین تعداد تکرار برابر است با:

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

مرتبه اجرایی حلقه ها (تو در تو)

حلقه های for زیر را در نظر بگیرید:

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

آنگاه هر حلقه nبار تکرار می‌شود، بنابراین در مرتبه اجرایی آن n^2 است.

مثال

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

تعداد تکرار:

$$(\log_2^n + 1) \times n \in O(n \lg n)$$

مثال

```
for(i=1;i<=m;i=i+1)
        some_commands;

for(j=1;j<=n;j=j+1)
        some_commands;
```

تعداد تکرار:

$$O(\max(n, m)) = O(n + m)$$

مثال

```
for(k=1;k<=m;k=k+1)
        some_commands;

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

تعداد تکرار:

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

حلقه های تو در تو وابسته

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

i	1	2	3	...	n
تعداد تکرار	1	2	3	...	n

تعداد تکرار:

$$1+2+3+...+n=\frac{n(n+1)}{2}\in O(n^2)$$

یا به عبارتی دیگر داریم:

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

حلقه های تو در تو وابسته

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

i	1	2	4	…	n
تعداد تکرار	1	2	4	…	n

تعداد تکرار:

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

$$a^0+a^1+a^2+\ldots+a^k = \frac{a^{k+1}-1}{a}$$

مثال

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

تعداد تکرار: در مثال قبل دیدیم که دو حلقه داخلی $O(n)$ مرتبه تکرار می‌شوند، از آنجا که حلقه بیرونی nمرتبه تکرار می‌شود، تعداد کل تکرار برابر است با:

$$n \times O(n) = O(n^2)$$

مثال

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

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

تعداد تکرار:

$$n+\frac{n}{2}+\frac{n}{3}+...+1=n(1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n})\approx n\ln n\in O(n\lg n)$$

$$\sum_{i=1}^{n}\frac{1}{i}=\ln(n)+\gamma$$

مثال

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

i	1	3	9	…	n
تعداد تکرار	n−1+1	n−3+1	n−9+1	…	n−n+1

تعداد تکرار:

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

تغییر مقدار نهایی شمارنده

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

i	1	2	3	…
تعداد تکرار	n/2	n/4	n/8	…

تعداد تکرار:

$$n + \frac{n}{2} + \frac{n}{4} + \ldots = n(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots) = n \times \frac{\frac{1}{2}}{1 - \frac{1}{2}} \in O(n)$$

$$a^0 + a^1 + a^2 + \ldots + a^k = \frac{a^{k+1} - 1}{a}$$

فراخوانی توابع

با فرض آنکه پیچیدگی f(k) برابر با O(k) باشد، پیچیدگی را مشخص کنید:

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

جواب:

$$n \times n = O(n^2)$$

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

جواب:

$$f(1) + f(2) + ... + f(n) = 1 + 2 + ... + n = \frac{n(n+1)}{2} \in O(n^2)$$