ساختمان داده‌ها و الگوریتم‌ها

(ریاضی)

نیمسال اول ۱۳۹۰-۱۳۸۹

فصل اول: روشهای تحلیل الگوریتم

زمان اجرای الگوریتم‌ها

زمان اجرای الگوریتم‌ها

همانطور که می‌دانیم الگوریتم عبارتست از:

مجموعه‌ای از دستورات و دستورالعمل‌ها برای حل مسئله، که شرایط زیر را می‌بایست دارا باشد:

- دقیق باشد
- مراحل آن به‌ترتیب انجام پذیرد
- پایان‌پذیر باشد

عوامل دخیل در زمان اجرای برنامه عبارتند از:

- سرعت سخت‌افزار
- نوع کامپایلر
- اندازه داده ورودی
- ترکیب داده‌های ورودی
- پیچیدگی زمانی الگوریتم
- پارامترهای دیگر که تأثیر ثابت در زمان اجرا دارند.

برای بررسی یک الگوریتم، تابعی به‌نام $T(n)$ که تابع زمانی الگوریتم نامیده می‌شود، در نظر می‌گیریم. که در آن n اندازهٔ ورودی مسئله است. مسئله ممکن است شامل چند دادهٔ ورودی باشد. به‌عنوان مثال اگر ورودی یک گراف باشد علاوه بر تعداد راس‌ها (n)، تعداد یال‌ها (m) هم یکی از مشخصه‌های دادهٔ ورودی می‌باشد. دراینصورت زمان اجرای الگوریتم را با $T(n,m)$ نمایش می‌دهیم

برای بررسی یک الگوریتم، تابعی به‌نام $T(n)$ که تابع زمانی الگوریتم نامیده می‌شود، در نظر می‌گیریم. که در آن n اندازهٔ ورودی مسئله است. مسئله ممکن است شامل چند دادهٔ ورودی باشد. به‌عنوان مثال اگر ورودی یک گراف باشد علاوه بر تعداد راس‌ها (n)، تعداد یال‌ها (m) هم یکی از مشخصه‌های دادهٔ ورودی می‌باشد. دراینصورت زمان اجرای الگوریتم را با $T(n,m)$ نمایش می‌دهیم

برای محاسبه تابع زمانی $T(n)$ برای یک الگوریتم موارد زیر را باید در محاسبات در نظر بگیریم:

- زمان مربوط به اعمال جایگزینی که مقدار ثابت می‌باشند.
- زمان مربوط به انجام اعمال محاسبات که مقدار ثابتی دارند.
- زمان مربوط به تکرار تعدادی دستور یا دستورالعمل (حلقه‌ها)
- زمان مربوط به توابع بازگشتی

مثال ۱:

```
(1) x = 0 ;
(2) for (i = 0 ;  i < n ; i++)
(3)       x++ ;
```

مثال ۱:

```
(1) x = 0 ;
(2) for (i = 0 ;  i < n ; i++)
(3)       x++ ;
```

سطر	زمان	تعداد
1	$C_۱$	۱
2	$C_۲$	$n+۱$
3	$C_۳$	n

مثال ۱:

```
(1) x = 0 ;
(2) for (i = 0 ;  i < n ; i++)
(3)       x++ ;
```

سطر	زمان	تعداد
1	$C_۱$	۱
2	$C_۲$	$n+۱$
3	$C_۳$	n

$$T(n) = C_۱ + C_۲(n+۱) + C_۳n$$

$$T(n) \leq C(۲n+۲) \qquad C = \max\{C_۱, C_۲, C_۳\}$$

مثال ۲:

```
(1)  x=0  ;
(2)  for (i=0 ;  i < n ; i++)
(3)        for (j=0 ;  j < n  ; j++)
(4)              x++ ;
```

مثال ۲:

```
(1)  x=0  ;
(2)  for (i=0 ;  i < n ; i++)
(3)        for (j=0 ;  j < n  ; j++)
(4)              x++ ;
```

سطر	هزینه	تعداد
1	C_1	1
2	C_2	$n+1$
3	C_3	$n(n+1)$
4	C_4	$n \times n$

مثال ۲:

```
(1)  x=0  ;
(2)  for (i=0 ;  i < n ; i++)
(3)        for (j=0 ;  j < n  ; j++)
(4)              x++ ;
```

سطر	هزینه	تعداد
1	$C_۱$	۱
2	$C_۲$	$n+۱$
3	$C_۳$	$n(n+۱)$
4	$C_٤$	$n \times n$

$$T(n) = C_۱ + C_۲(n+۱) + C_۳n(n+۱) + C_۴n^۲$$

$$T(n) \leq C(۲n^۲ + ۲n + ۲) \quad C = \max\{C_۱, C_۲, C_۳, C_۴\}$$

مثال ۳:

```
(1)     int     Factorial(int   n)
        {
(2)             int   fact= 1 ;
(3)             for( int  i=2  ; i<= n ; i++ )
(4)                     fact*= i ;
(5)             return fact ;
        }
```

مثال ۳:

```
(1)     int    Factorial(int  n)
        {
(2)             int  fact= 1 ;
(3)             for( int  i=2  ; i<= n ; i++ )
(4)                     fact*= i ;
(5)             return fact ;
        }
```

سطر	هزینه	تعداد
2	C_1	۱
3	C_2	n
4	C_3	n – ۱
5	C_4	۱

مثال ۳:

```
(1)     int    Factorial(int  n)
        {
(2)             int  fact= 1 ;
(3)             for( int  i=2  ; i<= n ; i++ )
(4)                     fact*= i ;
(5)             return fact ;
        }
```

سطر	هزینه	تعداد
2	$C_۱$	۱
3	$C_۲$	n
4	$C_۳$	n – ۱
5	$C_٤$	۱

$$T(n) = C_۱ + C_۲n + C_۳(n - ۱) + C_۴$$

$$T(n) \leq C(۲n + ۱)$$

مثال ۴:

```
    void    Add(a, b, c, int m,n)
            {
(1)                 for(int i=0 ; i<n ; i ++)
(2)                     for(int j=0 ; j<m ; j ++)
(3)                         c[i][j]= a[i][j] + b[i][j] ;
            }
```

مثال ۴:

```
void   Add(a, b, c, int m,n)
       {
(1)        for(int i=0 ; i<n ; i ++)
(2)            for(int j=0 ; j<m ; j ++)
(3)                c[i][j]= a[i][j] + b[i][j] ;
       }
```

سطر	هزینه	تعداد
1	C_1	$n+1$
2	C_2	$n(m+1)$
3	C_3	nm

مثال ۴:

```
void   Add(a, b, c, int m,n)
       {
(1)          for(int i=0 ; i<n ; i ++)
(2)              for(int j=0 ; j<m ; j ++)
(3)                  c[i][j]= a[i][j] + b[i][j] ;
       }
```

سطر	هزینه	تعداد
1	$C_۱$	$n+۱$
2	$C_۲$	$n(m+۱)$
3	$C_۳$	nm

$$T(n) = C_۱(n+۱) + C_۲n(m+۱) + C_۳nm$$

$$T(n) \leq C(۲nm + ۲n + ۱)$$

نماد $\mathcal{O}$ (Big-Oh)

تعریف: گوئیم $T(n) \in O(f(n))$ اگر و فقط اگر ثابت C و ثابت صحیح n_0 وجود داشته باشند که برای همه مقادیر $n \geq n_0$، داشته باشیم $T(n) \leq Cf(n)$ (رابطه $T(n) \in O(f(n))$ را بخوانید $T(n)$ متعلق به اوی بزرگ $f(n)$).

تعریف بالا به‌صورت زیر نیز بیان می‌شود:

$$T(n) \in O(f(n)) \Leftrightarrow \exists C, n_0 > 0 \text{ به‌طوری‌که } \forall n \geq n_0 \quad T(n) \leq Cf(n)$$

نماد $\mathcal{O}$ (Big-Oh)

تعریف: گوئیم $T(n) \in O(f(n))$ اگر و فقط اگر ثابت C و ثابت صحیح n_0 وجود داشته باشند که برای همه مقادیر $n \geq n_0$، داشته باشیم $T(n) \leq Cf(n)$ (رابطه $T(n) \in O(f(n))$ را بخوانید $T(n)$ متعلق به اوی بزرگ $f(n)$).

تعریف بالا به‌صورت زیر نیز بیان می‌شود:

$$T(n) \in O(f(n)) \Leftrightarrow \exists C, n_0 > 0 \text{ به‌طوری‌که } \forall n \geq n_0 \quad T(n) \leq Cf(n)$$

در حالت کلی $f(n)$ مرتبه زمانی اجرای الگوریتم نامیده می‌شود (اصطلاحاً پیچیدگی زمانی الگوریتم هم گفته می‌شود) و با $O(f(n))$ نمایش داده می‌شود.

مثال :

$$T(n) \leq C(۲n + ۲) \quad \Rightarrow T(n) \leq ۳n(C = ۱) \quad \Rightarrow T(n) \in \mathcal{O}(n)$$

مثال :

$$T(n) \leq C(۲n + ۲) \quad \Rightarrow T(n) \leq ۳n(C = ۱) \quad \Rightarrow T(n) \in \mathcal{O}(n)$$

$$T(n) \leq C(۲n^۲ + ۲n + ۲) \quad \Rightarrow T(n) \leq ۴n^۲ \Rightarrow T(n) \in \mathcal{O}(n^۲)$$

مثال :

$$T(n) \leq C(۲n + ۲) \quad \Rightarrow T(n) \leq ۳n(C = ۱) \quad \Rightarrow T(n) \in \mathcal{O}(n)$$

$$T(n) \leq C(۲n^۲ + ۲n + ۲) \quad \Rightarrow T(n) \leq ۴n^۲ \Rightarrow T(n) \in \mathcal{O}(n^۲)$$

$$T(n) \in \mathcal{O}(n^۳)??$$

مثال :

$$T(n) \leq C(۲n + ۲) \quad \Rightarrow T(n) \leq ۳n(C = ۱) \quad \Rightarrow T(n) \in \mathcal{O}(n)$$

$$T(n) \leq C(۲n^۲ + ۲n + ۲) \quad \Rightarrow T(n) \leq ۴n^۲ \Rightarrow T(n) \in \mathcal{O}(n^۲)$$

$$T(n) \in \mathcal{O}(n^۳)??$$

$$T(n) \leq C(۲nm + ۲n + ۱)$$

اگر $m \leq n \Rightarrow T(n) \in \mathcal{O}(n^2)$

تمرین:

i) $T(n) = (۲n + ۱) \in O(n^۲)$

ii) $T(n) = (۵n^۲ + n + ۱) \in O(n)$

iii) $T(n) = (۴ * ۲^n + n^۲) \in O(۲^n)$

قضیه ۱-۱: اگر $T(n)=a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0$ زمان اجرای یک الگوریتم باشد آنگاه $T(n) \in O(n^m)$.

i) $T(n) = (n+1)^2 \in O(n^2)$

ii) $T(n) = 3n^3 + 2n^2 \in O(n^3)$

iii) $T(n) = 5^n \notin O(2^n)$

نماد Ω (Big-Omega)

تعریف: گوئیم $T(n) \in \Omega(f(n))$ اگر و فقط اگر ثابت صحیح C و n_0 وجود داشته باشد که به ازای همه مقادیر $n \geq n_0$ داشته باشیم $T(n) \geq Cf(n)$ (رابطه $T(n) \in \Omega(f(n))$ را بخوانید $T(n)$ امگای بزرگ $f(n)$).

تعریف بالا را به‌صورت زیر نیز ارائه می‌دهند:

$$T(n) \in \Omega(f(n)) \Leftrightarrow \exists C, n_0 > 0 \quad \forall n \geq n_0 \qquad Cf(n) \leq T(n)$$

اگر دقت کنید ملاحظه می‌کنید که تعریف بالا یک کران پایین زمان اجرا برای $T(n)$ ارائه می‌دهد. بنابراین، در حالت کلی می‌توان گفت که $\Omega(f(n))$ بهترین حالت اجرا برای یک الگوریتم می‌باشد.

مثال :

$$T(n) = an^۲ + bn + c \quad (a, b, c > ۰)$$

$$T(n) = an^۲ + bn + c \quad > an^۲ + b + c \quad > an^۲ \Longrightarrow T(n) \in \Omega(n^۲)$$

مثال :

$$T(n) = an^۲ + bn + c \quad (a, b, c > ۰)$$

$$T(n) = an^۲ + bn + c \quad > an^۲ + b + c \quad > an^۲ \Longrightarrow T(n) \in \Omega(n^۲)$$

$$T(n) = n^۴ + ۵n^۲ \quad \geq n^۴ \Longrightarrow T(n) \in \Omega(n^۴)$$

تمرین:

i) $T(n) = ٦n + ٤ \in \Omega(n)$

ii) $T(n) = ٣n + ٢ \notin \Omega(n^{٢})$

iii) $T(n) = ٥^{n} + n^{٢} \in \Omega(٢^{n})$

قضیه ۲-۱: اگر $T(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0$ زمان اجرای یک الگوریتم بوده و $a_m > 0$ باشد آنگاه $T(n) \in \Omega(n^m)$.

نماد Θ

تعریف: گوئیم $T(n) \in \theta(f(n))$ اگر و فقط اگر ثابتهای C_1، C_2 و ثابت صحیح n_0 وجود داشته باشد به‌طوری‌که برای همه مقادیر $n \geq n_0$:

$$C_1 f(n) \leq T(n) \leq C_2 f(n)$$

تعریف بالا به‌صورت زیر نیز ارائه می‌شود:

$$\exists C_1, C_2, n_0 > 0 \text{ به‌طوری‌که } \forall n \geq n_0 \quad C_1 f(n) \leq T(n) \leq C_2 f(n)$$

$$\Leftrightarrow T(n) \in \theta(f(n))$$

توسط نماد بالا تابع $T(n)$ هم از بالا و هم از پایین محدود می‌شود. توجه داشته باشید که، درجه رشد تابع $f(n)$ و $T(n)$ یکسان است.

مثال :

$$T(n) = \frac{۱}{۲} n^{۲} - ۳n$$

مثال :

$$T(n) = \frac{1}{2}n^2 - 3n$$

حل:

نشان می‌دهیم که $T(n) \in \theta(n^2)$.

طبق تعریف:

$$T(n) \in \theta(n^2) \quad \Leftrightarrow$$

(۱) $C_1 n^2 \le \frac{1}{2}n^2 - 3n \le C_2 n^2$ $\forall n \ge n_0$ به‌طوری‌که $\exists C_1 و C_2 و n_0$.

حال رابطه (۱) را به n^2 تقسیم می‌کنیم:

$$C_1 \le \frac{1}{2} - \frac{3}{n} \le C_2$$

با توجه به عبارت بالا، طرف راست، به ازای $C_2 \ge 1/2$ و $n \ge 1$ برقرار است.

به‌همین ترتیب برای طرف چپ عبارت بالا به ازای $n \ge 7$، $C_1 \le 1/4$ حاصل می‌شود.

بنابراین به ازای $C_1 = 1/4$ و $C_2 = 1/2$ و $n_0 \ge 7$ عبارت $T(n) \in \theta(n^2)$ خواهد بود.

مثال ۱۱-۱: فرض کنید $T(n) = 7n^3$ باشد. آیا $T(n) \in \theta(n^2)$ می‌تواند باشد.

حل: طبق تعریف θ باید داشته باشیم:

$$\exists C_1, C_2, n_0 > 0 \text{ به‌طوری‌که } \forall n \geq n_0 \quad C_1 n^2 \leq 7n^3 \leq C_2 n^2$$

طرف چپ رابطه بالا همیشه برقرار است اما طرف راست رابطه بالا در صورتی برقرار است که $n \leq C_2/7$ باشد و این با تعریف θ که در آن رابطه برای هر n بزرگ‌تر از n_0 برقرار است منافات دارد، لذا $T(n)$ نمی‌تواند متعلق به $\theta(n^2)$ باشد.

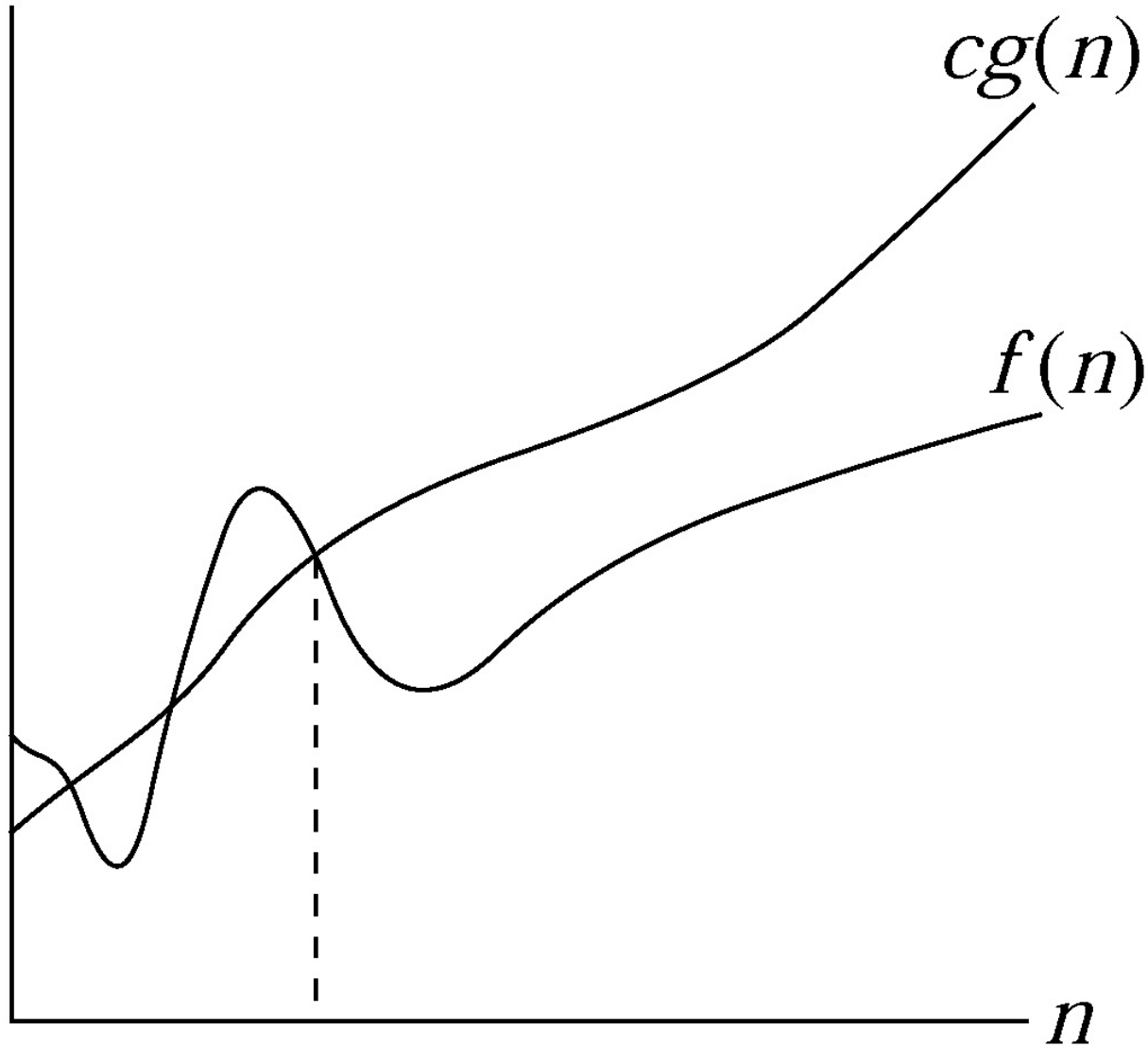
تمرین:

i) $T(n) = ٦n + ٤ \in \theta(n)$

ii) $T(n) = ٣n + ٦ \notin \theta(n^{٢})$

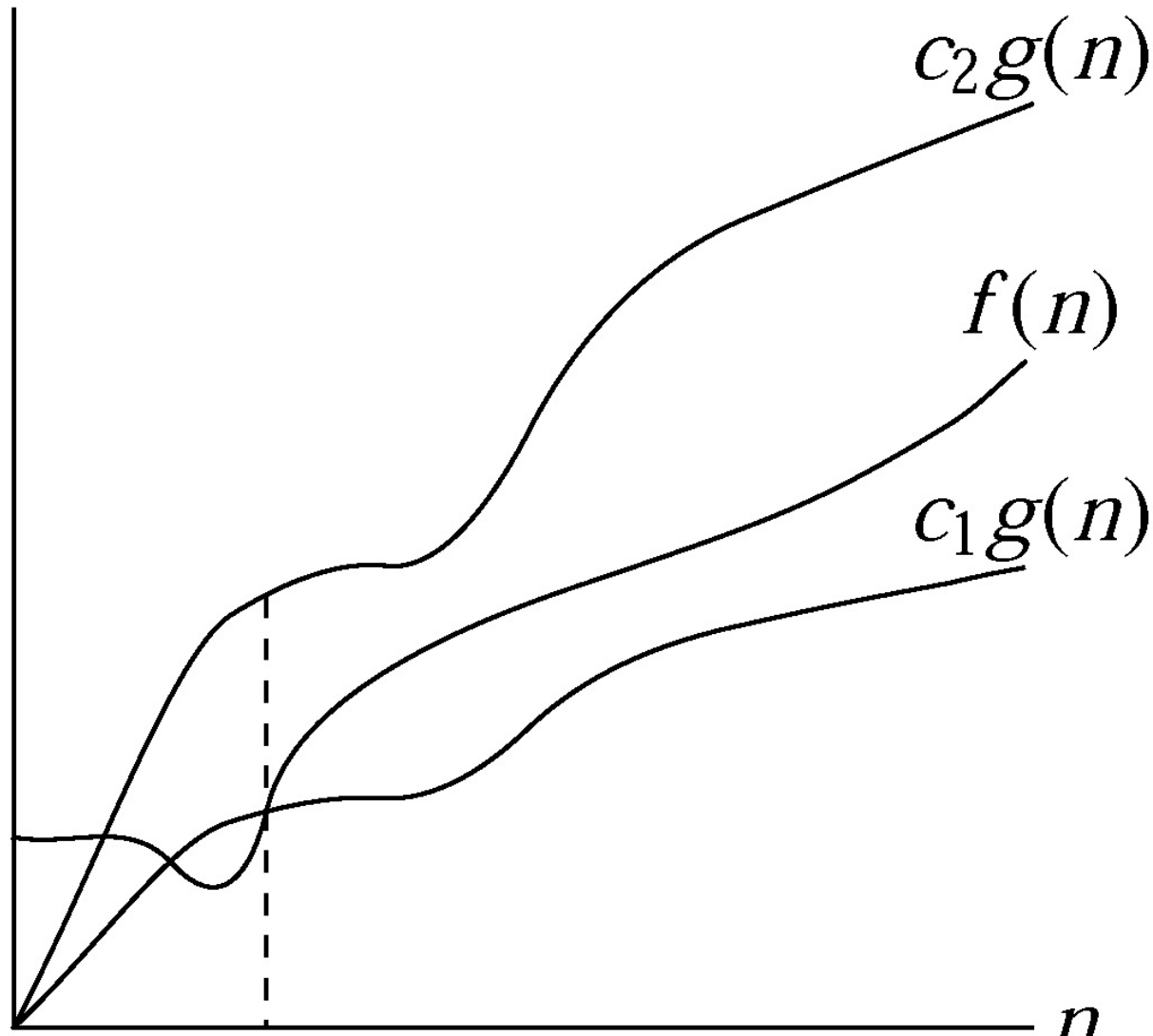
قضیه **۳-۱**: اگر $T(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0$ زمان اجرای یک الگوریتم بوده و $a_m > 0$ باشد آنگاه $T(n) \in \theta(n^m)$.

نمودار نمادهای $\mathcal{O}, \Omega, \Theta$



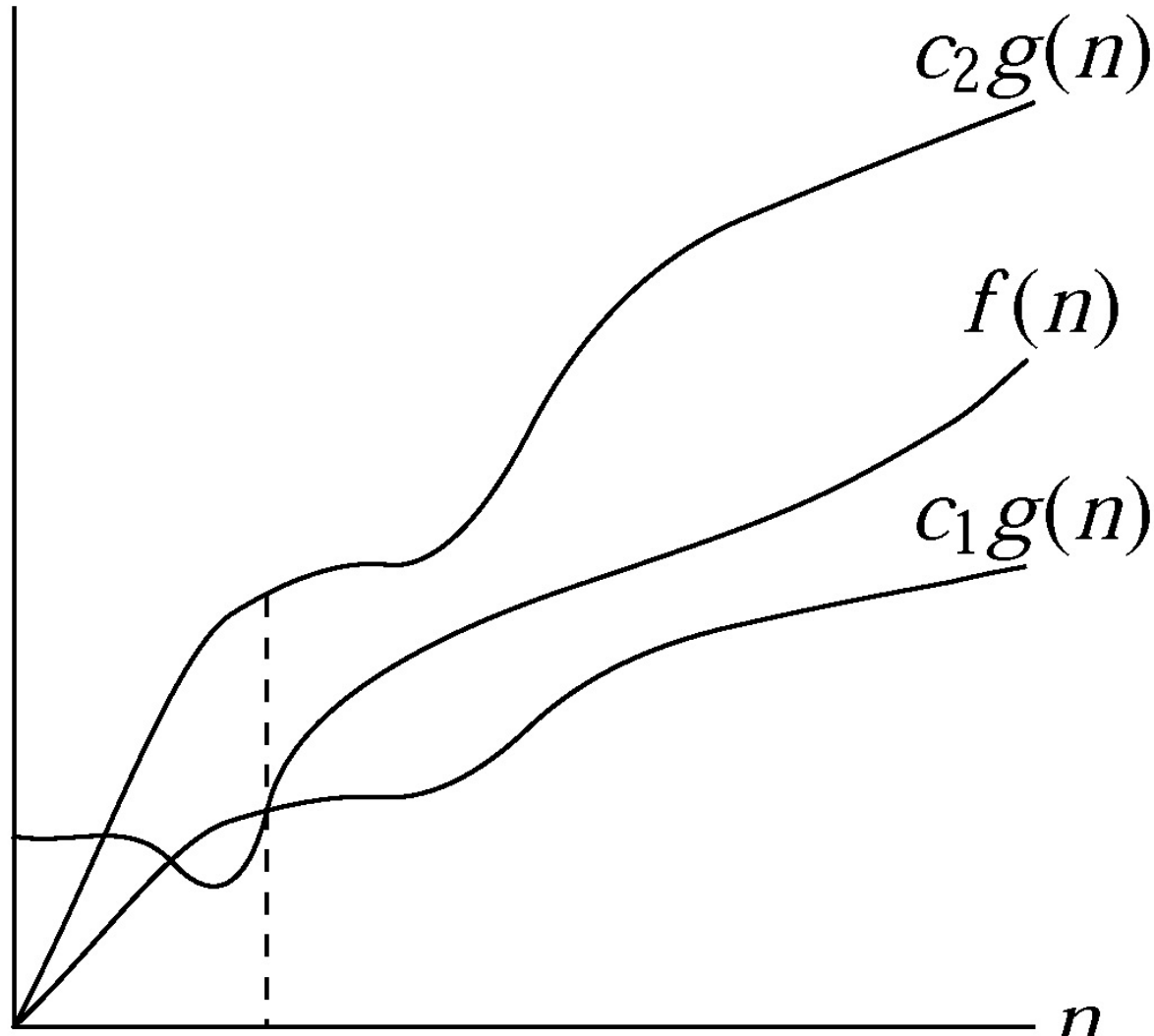
$$f(n) = O(g(n))$$

نمودار نمادهای $\mathcal{O}, \Omega, \Theta$



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

نمودار نمادهای $\mathcal{O}, \Omega, \Theta$



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

مرتبه رشد

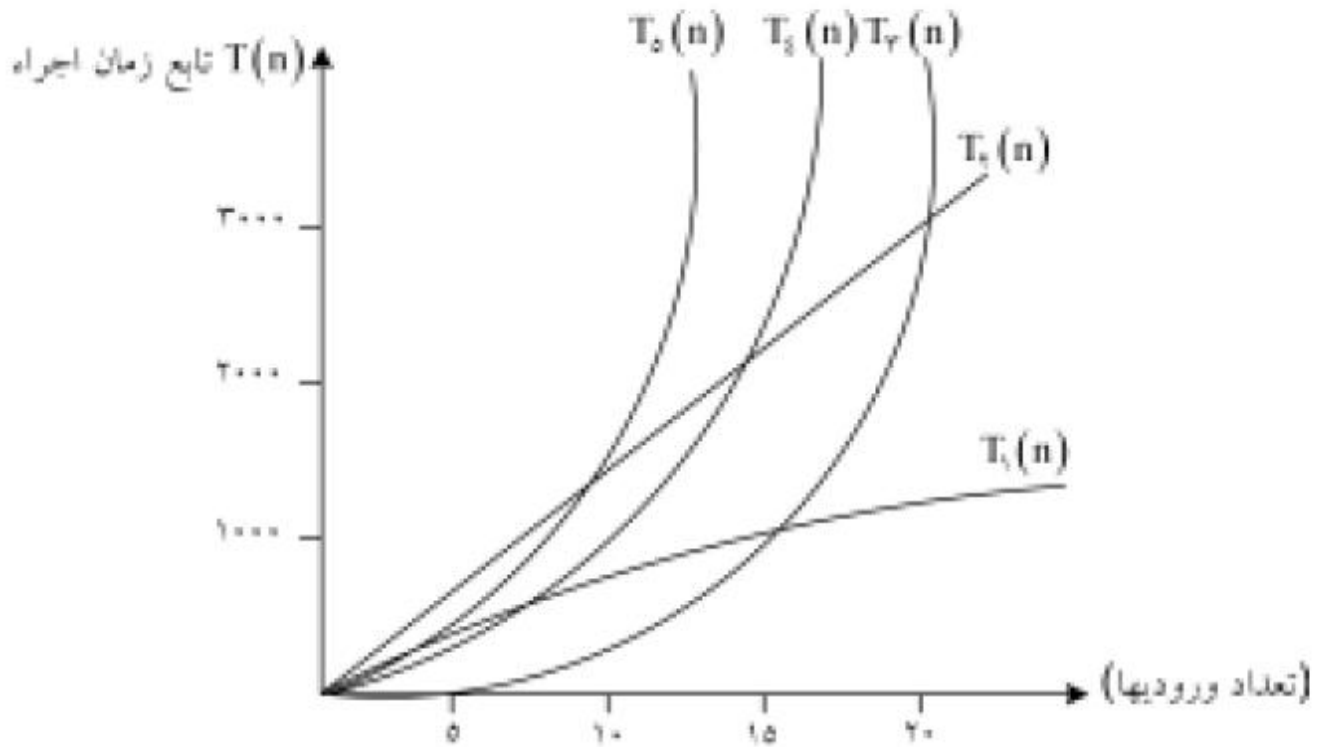
فرض کنید که توابع زمان اجرای برنامه‌ها به صورت زیر باشند:

$$T_1(n) = ۲\text{Log}_۲^n \quad ، T_۲(n) = ۱۰۰n$$

$$T_۳(n) = ۵n^۲ \quad ، T_۴(n) = \frac{۱}{۲} n^۳$$

$$T_۵(n) = ۲^n$$

حال نمودار توابع بالا را به ازای nهای مختلف تحلیل می‌کنیم:



شکل ۲-۱ زمان اجرای پنج برنامه

زمان اجراء $T(n)$	بیشترین اندازه مسئله برای 10^3 ثانیه	بیشترین اندازه مسئله برای 10^4 ثانیه	بیشترین اندازه مسئله
$100n$	۱۰	۱۰۰	۱۰۰۰
$5n^2$	۱٤	٤٥	۳۲۰
$n^3/2$	۱۲	۲۷	۲۳۰
2^n	۱۰	۱۳	۱۳۰

جدول ۱-۱: زمان اجرای ٤ برنامه

کوییز:

درستی یا نادرستی عبارات زیر را با ذکر دلیل تعیین کنید:

۱) $(n+a)^b \in \Theta(n^b)$ $\quad$ (a, b ثابت)

۲) $\max\{f(n), g(n)\} \in \mathcal{O}(f(n))$

۳) $\max\{f(n), g(n)\} \in \Theta(f(n)+g(n))$

روش‌های تحلیل الگوریتم‌ها

الگوریتم‌های ترتیبی(غیر بازگشتی)

۱. اعمال انتساب، عملگرهای محاسباتی، شرط‌های if (ساده) و غیره زمان ثابت دارند.

۲. اگر تعدادی دستور تکرار شوند زمان اجراء، حاصلضرب تعداد تکرار در زمان اجرای دستورات خواهد بود. که معمولاً این قسمت از برنامه‌ها توسط حلقه‌ها نمایش داده می‌شوند.

۳. اگر برنامه شامل ساختار if و else باشد. که هر کدام زمان‌های T_1 و T_2 داشته باشند، در اینصورت زمان اجرای این تکه برنامه برابر بیشترین مقدار T_1 و T_2 خواهد بود.

٤. زمان کل برنامه برابر حاصل جمع زمای تکه برنامه‌ها می‌باشد.

مثال :

مسئله: پیچیدگی زمانی مرتب‌سازی حبابی را تحلیل کنید.

ورودی: آرایه A از عناصر و n تعداد عناصر آرایه

خروجی: لیست مرتب از داده‌ها

```
void    bubble (elementtype  A[ ] , int    n)
{
        for (i=0  ;  i < n – 1  ;  i++)
            for (j= n – 1   ;  j >= i + 1  ;  j --)
                if (A[ j-1 ] > A[ j ])
                    exchange (A[ j ] , A[ j – 1 ])
}
```

مثال :

مسئله: پیچیدگی زمانی مرتب‌سازی حبابی را تحلیل کنید.

ورودی: آرایه A از عناصر و n تعداد عناصر آرایه

خروجی: لیست مرتب از داده‌ها

```
void    bubble (elementtype  A[ ] , int    n)
{
        for (i=0  ;  i < n – 1  ;  i++)
            for (j= n – 1   ;  j >= i + 1  ;  j --)
                if (A[ j-1 ] > A[ j ])
                    exchange (A[ j ] , A[ j – 1 ])
}
```

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d = d \sum_{i=1}^{n-1} (n-i)$$

$$= d\, n(n-1)/2 = d\left(\frac{n^2}{2} - \frac{n}{2}\right)$$

مثال :

مسئله: پیچیدگی زمانی مرتب‌سازی حبابی را تحلیل کنید.

ورودی: آرایه A از عناصر و n تعداد عناصر آرایه

خروجی: لیست مرتب از داده‌ها

```
void    bubble (elementtype  A[ ] , int    n)
{
        for (i=0  ;  i < n – 1  ;  i++)
            for (j= n – 1   ;  j >= i + 1  ;  j --)
                if (A[ j-1 ] > A[ j ])

                    exchange (A[ j ] , A[ j – 1 ])
}
```

$$T(n)=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d = d\sum_{i=1}^{n-1}(n-i)$$

$$= d\, n(n-1)/2 = d\left(\frac{n^2}{2}-\frac{n}{2}\right)$$

$$T(n)=\mathcal{O}(n^2)$$

$$T(n)=\Omega(n^2)$$

$$\Rightarrow T(n)=\Theta(n^2)$$

مثال :

- **الگوریتم ۲-۱ جستجوی ترتیبی**

مسئله: پیچیدگی زمانی الگوریتم جستجوی ترتیبی را تحلیل نمایید.

ورودی: A آرایه‌ای از عناصر، n تعداد عناصر، x عنصر مورد جستجو.

خروجی: اندیس عنصر مورد جستجو در صورت وجود.

```
int   Seq_Search (elementtype a[ ] ,  int    n, elementtype   x)
{
      int    i
      for (i=0 ;  i < n  ;  i++)
          if (a[ i ] == x)
              return (i)
       return (-1)
}
```

مثال :

- **الگوریتم ۲-۱ جستجوی ترتیبی**

مسئله: پیچیدگی زمانی الگوریتم جستجوی ترتیبی را تحلیل نمایید.

ورودی: A آرایه‌ای از عناصر، n تعداد عناصر، x عنصر مورد جستجو.

خروجی: اندیس عنصر مورد جستجو در صورت وجود.

```
int   Seq_Search (elementtype a[ ] , int   n, elementtype  x)
{
      int   i
      for (i=0 ;  i < n  ;  i++)
          if (a[ i ] == x)
              return (i)
       return (-1)
}
```

بهترین حالت الگوریتم زمانی اتفاق می‌افتد که عنصر مورد جستجو با اولین عنصر آرایه برابر باشد در اینصورت $T(n) \in O(1)$ می‌باشد.

مثال :

- **الگوریتم ۲-۱ جستجوی ترتیبی**

مسئله: پیچیدگی زمانی الگوریتم جستجوی ترتیبی را تحلیل نمایید.

ورودی: A آرایه‌ای از عناصر، n تعداد عناصر، x عنصر مورد جستجو.

خروجی: اندیس عنصر مورد جستجو در صورت وجود.

```
int   Seq_Search (elementtype a[ ] ,  int    n, elementtype   x)
{
      int    i
      for (i=0 ;  i < n  ;  i++)
          if (a[ i ] == x)
              return (i)
       return (-1)
}
```

بهترین حالت الگوریتم زمانی اتفاق می‌افتد که عنصر مورد جستجو با اولین عنصر آرایه برابر باشد در اینصورت $T(n) \in O(1)$ می‌باشد.

الگوریتم بالا در بدترین حالت، که عنصر مورد جستجو با عنصر nام برابر باشد دارای پیچیدگی زمانی $O(n)$ خواهد بود.

مثال :

- **الگوریتم ۵-۱ یافتن بیشترین مقدار**

مسئله: پیچیدگی زمانی پیدا کردن بیشترین مقدار در یک آرایه را تحلیل نمائید.

ورودی: آرایه A، n تعداد عناصر.

خروجی: بیشترین مقدار آرایه

```
elementtype  Maximum(elementtype  A[ ]  ,  int  n)
{
      Max = A[0] ;
      for(i = 1  ;  i< n  ;  i++)
          if(A[i] > Max)
                Max = A[i] ;
       return(Max) ;
}
```

مثال :

- الگوریتم ۵-۱ یافتن بیشترین مقدار

مسئله: پیچیدگی زمانی پیدا کردن بیشترین مقدار در یک آرایه را تحلیل نمائید.

ورودی: آرایه A، n تعداد عناصر.

خروجی: بیشترین مقدار آرایه

```
elementtype  Maximum(elementtype  A[ ]  ,  int  n)
{
      Max = A[0] ;
      for(i = 1  ;  i< n  ;  i++)
          if(A[i] > Max)
                Max = A[i] ;
       return(Max) ;
}
```

$$T(n) \in \mathcal{O}(n)$$

روش‌های تحلیل الگوریتم‌ها

الگوریتم‌های بازگشتی (Recursive Algorithms)

معمولاً در الگوریتم‌های بازگشتی، مسئله را به دو یا چند زیرمسئله کوچک‌تر تقسیم می‌کنیم. عمل تقسیم مسئله به زیرمسئله‌ها را تا زمانی که اندازه زیرمسئله‌ها به اندازه کافی کوچک شوند ادامه می‌دهیم. بعد از تقسیم به اندازه کافی، برای حل زیرمسئله‌ها از خود الگوریتم استفاده می‌کنیم. سپس حاصل زیرمسئله‌ها را با هم ترکیب می‌کنیم تا راه‌حل مسئله بزرگ‌تر حاصل شود. اعمال ترکیب حاصل زیرمسئله‌ها را تا زمانی‌که مسئله اصلی حل نشده باشد ادامه می‌دهیم.

برای محاسبه زمان اجرای الگوریتم‌های بازگشتی به صورت زیر عمل می‌کنیم:

۱. زمان حل زیرمسئله‌ها را محاسبه می‌کنیم (که معمولاً مقدار ثابتی است)

۲. زمان لازم برای شکستن مسئله به زیرمسئله‌ها

۳. زمان لازم برای ادغام جوابهای زیرمسئله‌ها.

اگر مجموع سه زمان بالا را محاسبه کنیم، زمان اجرای الگوریتم به‌دست خواهد آمد.

روش‌های تحلیل الگوریتم‌ها

محاسبه تابع زمانی الگوریتم‌های بازگشتی

روش‌های تحلیل الگوریتم‌ها

محاسبه تابع زمانی الگوریتم‌های بازگشتی

- **الگوریتم ٦-١ محاسبه فاکتوریل**

مسئله: الگوریتم بازگشتی برای محاسبه فاکتوریل یک عدد نوشته و زمان اجرای الگوریتم را تحلیل کنید.

ورودی: عدد صحیح n

خروجی: محاسبه فاکتوریل عدد صحیح n

روش‌های تحلیل الگوریتم‌ها

محاسبه تابع زمانی الگوریتم‌های بازگشتی

- **الگوریتم ۶-۱ محاسبه فاکتوریل**

مسئله: الگوریتم بازگشتی برای محاسبه فاکتوریل یک عدد نوشته و زمان اجرای الگوریتم را تحلیل کنید.

ورودی: عدد صحیح n

خروجی: محاسبه فاکتوریل عدد صحیح n

$$(۱) \qquad n! = \begin{cases} ۱ & \text{if } n = ۰ \\ ۱ \times ۲ \times ۳ \times \ldots \times (n-۱) \times n & \text{if } n > ۰ \end{cases}$$

$$(۲) \qquad n! = \begin{cases} ۱ & \text{if } n = ۰ \\ n \times (n-۱)! & \text{if } n > ۰ \end{cases}$$

روش‌های تحلیل الگوریتم‌ها

محاسبه تابع زمانی الگوریتم‌های بازگشتی

- **الگوریتم ٦-١ محاسبه فاکتوریل**

مسئله: الگوریتم بازگشتی برای محاسبه فاکتوریل یک عدد نوشته و زمان اجرای الگوریتم را تحلیل کنید.

ورودی: عدد صحیح n

خروجی: محاسبه فاکتوریل عدد صحیح n

```
int       fact (int   n)
{

          if (n == 0)
                return (1) ;
          else
                return (n  *  fact (n – 1))  ;
}
```

```
int       fact (int   n)
{

          if (n == 0)
                return (1) ;
          else
                return (n  *  fact (n – 1))  ;
}
```

$T(n)$ را زمان اجرای تابع fact(n) در نظر می‌گیریم. زمان اجرای دستور if برابر $O(۱)$ می‌باشد و زمان اجرای else دستور if برابر $O(۱)+T(n-۱)$ که در آن $O(۱)$ زمان مربوط به عمل ضرب و فراخوانی تابع می‌باشد. بنابراین:

$$T(n)=\begin{cases} O(۱) & \text{if } n=۰ \\ O(۱)+T(n-۱) & \text{if } n>۰ \end{cases}$$

$T(n)$ را به صورت زیر نیز می‌توان نوشت:

$$T(n)=\begin{cases} d & \text{if } n=۰ \\ T(n-۱)+C & \text{if } n>۰ \end{cases}$$

روش‌های حل رابطه‌های بازگشتی

۱) حدس و استقرا: حدس خوب و استقرا

۲) تکرار با جای‌گذاری: بازکردن فرمول و به‌دست آوردن جواب به‌طور صریح

۳) قضیه‌ی اصلی

۴) روش‌های حل رابطه‌های بازگشتی همگن و ناهمگن

۵) روش‌های دیگر مانند تابع مولد

حدس و استقرا: مثال

$$T(n) = T(\lfloor \frac{n}{۲} \rfloor) + n$$
$$T(۲) = T(۱) = \mathcal{O}(۱)$$

حدس و استقرا:مثال

$$T(n) = T(\lfloor \frac{n}{۲} \rfloor) + n$$
$$T(۲) = T(۱) = \mathcal{O}(۱)$$

حدس: $T(n) \leq Cn$ **(چه‌گونه حدس می‌زنیم؟)**

برای عدد مثبت C **(برای آن‌که اثبات کنیم** $T(n) = \mathcal{O}(n)$**)**

حدس و استقرا: مثال

$$T(n) = T(\lfloor \frac{n}{۲} \rfloor) + n$$
$$T(۲) = T(۱) = \mathcal{O}(۱)$$

پایه‌ی استقرا:

$$n = ۲ \Rightarrow T(۲) \leq ۲C \Rightarrow ۲C \leq \mathcal{O}(۱) = r$$

فرض استقرا: برای $k < n$ **فرض می‌کنیم** $T(k) \leq Ck$

حکم استقرا: باید ثابت کنیم: $T(n) \leq Cn$.

$$\begin{aligned} T(n) &\leq C\lfloor \tfrac{n}{۲} \rfloor + n \\ &\leq Cn/۲ + n \end{aligned}$$

برای آن‌که $Cn/۲ + n \leq Cn$ **شود، باید داشته باشیم** $C \geq ۲$.

پس برای $۲ \leq C \leq r/۲$ **گام استقرا اثبات می‌شود. اگر** $C \geq ۱$ **حکم اثبات می‌شود.**

مثال: مرتب‌سازی ادغامی

$$T(۲) = a$$
$$T(n) = ۲T(\frac{n}{۲}) + bn$$

مثال: مرتب‌سازی ادغامی

$$T(۱) = T(۲) = a$$
$$T(n) = ۲T(\frac{n}{۲}) + bn$$

۱) حدس: $T(n) \leq cn \lg n \Rightarrow T(n) = \mathcal{O}(n \lg n)$

۲) پایه: $T(۲) = a \leq ۲c$

۳) فرض استقرا: برای $\frac{n}{۲}$ داریم، $T(n/۲) <= c\frac{n}{۲}\lg\frac{n}{۲}$

$$\begin{aligned} T(n) &\leq ۲c\frac{n}{۲}\lg\frac{n}{۲} + bn \\ &= cn\lg n + (b-c)n \leq cn\lg n \\ &\leq cn\lg n, \;\; if \;\; b-c \leq ۰ \;\; and \;\; c \geq b, c \geq a/۲ \end{aligned}$$

برای این که اثبات کنیم $T(n) = \Theta(n\lg n)$ باید اثبات کنیم که $T(n) = \Omega(n\lg n)$ هم هست.

برای این کار باید اثبات کنیم که یک c هست که برای n های بزرگ داریم، $T(n) \geq cn\lg n$

پایه: $T(۲) = ۲ \geq ۲c$

فرض و حکم:

$$\begin{aligned} T(n) &\geq ۲c\frac{n}{۲}\lg\frac{n}{۲} + bn \\ &= cn\lg n + (b-c)n \leq cn\lg n \\ &\geq cn\lg n, \;\; if \;\; b-c \geq ۰ \;\; and \;\; c \leq a/۲ \end{aligned}$$

مثال

$$T(n) = T(\lfloor \frac{n}{۲} \rfloor) + T(\lceil \frac{n}{۲} \rceil) + ۱$$

مثال

$$T(n) = T(\lfloor \frac{n}{۲} \rfloor) + T(\lceil \frac{n}{۲} \rceil) + ۱$$

حدس: $T(n) = \mathcal{O}(n)$. **باید ثابت کنیم** $T(n) \leq Cn$.

مثال

$$T(n) = T(\lfloor \frac{n}{۲} \rfloor) + T(\lceil \frac{n}{۲} \rceil) + ۱$$

حدس: $T(n) = \mathcal{O}(n)$. **باید ثابت کنیم** $T(n) \leq Cn$.

$$T(n) \leq C\lfloor \frac{n}{۲} \rfloor + C\lceil \frac{n}{۲} \rceil + ۱ = Cn + ۱ \geq Cn$$

اشتباه!

حدس دیگر: $T(n) \leq Cn + B$.

$$T(n) \leq C\lfloor\frac{n}{۲}\rfloor + C\lceil\frac{n}{۲}\rceil + ۲B + ۱ = Cn + ۲B + ۱ \leq Cn + B \Rightarrow B < ۰$$

مثال دیگر از حدس اشتباه

$$T(n) = ۲T(\lfloor \frac{n}{۲} \rfloor) + n$$

مثال دیگر از حدس اشتباه

$$T(n) = ۲T(\lfloor \frac{n}{۲} \rfloor) + n$$

حدس: $T(n) \leq Cn$

مثال دیگر

$$T(n) = ۲T(\lfloor \frac{n}{۲} \rfloor) + n$$

حدس: $T(n) \leq Cn$

$$T(n) \leq ۲C\lfloor \frac{n}{۲} \rfloor + n \leq Cn + n = (C + ۱)n \geq Cn$$

اشتباه!

حدس جدید: $T(n) \leq Cn \lg n + B$

درست است؟

می‌توان نشان داد که اگر $n = ۲^k$ فرض کنیم، حل رابطه‌ی بازگشتی با این فرض از نظر $\mathcal{O}$ همان جواب است. چرا؟

برای هر $۲^k \leq n < ۲^{k+۱}$ می‌توان نشان داد که مرتبه‌ی $T(n)$ همان مرتبه‌ی $T(۲^k)$ است.

مثال

$$T(n) = ۲T(\sqrt{n}) + \lg n$$

مثال

$$T(n) = ۲T(\sqrt{n}) + \lg n$$

متغیر جدید: $m = \lg n \Leftarrow T(۲^m) = ۲T(۲^{\frac{m}{۲}}) + m$.

$$S(m) = ۲S(\frac{m}{۲}) + m$$

مثال

$$T(n) = ۲T(\sqrt{n}) + \lg n$$

$$S(m) = \mathcal{O}(m \lg m) \Rightarrow T(n) = \mathcal{O}(\lg n \lg \lg n)$$

حل روابط بازگشتی

حل روابط بازگشتی

روش تکرار با جای‌گذاری

حل روابط بازگشتی

روش تکرار با جای‌گذاری

این روش با استفاده از جای‌گذاری های متوالی می‌تواند، جواب مناسب را تولید کند. در این روش با توجه به خاصیت رابطه بازگشتی به ازای nهای مختلف (که در نهایت به یک مقدار ثابت می‌رسد) و جای‌گذاری آنها در هم جواب مسئله حاصل می‌شود.

حل روابط بازگشتی روش تکرار با جای‌گذاری

مثال ۱۳-۱: رابطه بازگشتی زیر را در نظر بگیرید:

$$T(n)=\begin{cases}C & \text{if} \quad n=۲\\ T(n-۲)+d & \text{if} \quad n>۲\end{cases}$$

رابطه بالا را به روش تکرار با جایگذاری حل کنید.

طرف راست رابطه بالا را بسط می‌دهیم. بنابراین خواهیم داشت:

$$\begin{aligned}T(n)&=T(n-۲)+d\\&=\dots\\&=T(n-۲i)+i\times d\end{aligned}$$

رابطه بالا تا زمانیکه به $T(۲)$ نرسیدیم ادامه می‌دهیم. بنابراین اگر $n-۲i$ به عدد ۲ برسد آنگاه $T(۲)$ حاصل می‌شود:

$$n-۲i=۲ \quad \Rightarrow \quad i=(n-۲)/۲$$

با جایگذاری در رابطه بالا خواهیم داشت:

$$\begin{aligned}T(n)&=T(۲)+\frac{(n-۲)}{۲}\times d\\&=C+\frac{(n-۲)}{۲}\times d\end{aligned}$$

بنابراین $T(n)\in O(n)$.

حل روابط بازگشتی روش تکرار با جای‌گذاری

مثال ۱۴-۱: رابطه بازگشتی زیر را در نظر بگیرید:

$$T(n) = ۲T\left(\left\lfloor n/۲ \right\rfloor\right) + d \qquad (۱-۱)$$

که در آن d یک ثابت زمانی می‌باشد. روش تکرار با جای‌گذاری را برای رابطه بازگشتی (۱-۱) به‌صورت زیر به‌کار می‌بریم:

$$\begin{aligned} T(n) &= ۲T\left(\left\lfloor n/۲ \right\rfloor\right) + d \\ &= ۴T\left(\frac{\left\lfloor n/۲ \right\rfloor}{۲}\right) + ۳d \\ &\le ۴T\left(n/۴\right) + ۳d \\ &= \dots \\ &\le ۲^i T\left(n/۲^i\right) + \left(۲^i - ۱\right)d \end{aligned} \qquad (۱-۲)$$

حل روابط بازگشتی روش تکرار با جای‌گذاری

مثال ۱۴-۱: رابطه بازگشتی زیر را در نظر بگیرید:

$$T(n) = ۲T\left(\left\lfloor n/۲ \right\rfloor\right) + d \qquad (۱-۱)$$

که در آن d یک ثابت زمانی می‌باشد. روش تکرار با جای‌گذاری را برای رابطه بازگشتی (۱-۱) به‌صورت زیر به‌کار می‌بریم:

$$\begin{aligned} T(n) &= ۲T\left(\left\lfloor n/۲ \right\rfloor\right) + d \\ &= ۴T\left(\frac{\left\lfloor n/۲ \right\rfloor}{۲}\right) + ۳d \\ &\le ۴T\left(n/۴\right) + ۳d \\ &= \ldots \\ &\le ۲^i T\left(n/۲^i\right) + \left(۲^i - ۱\right)d \end{aligned} \qquad (۱-۲)$$

$$\frac{n}{۲^i} = ۱ \Rightarrow i = \log_۲ n$$

حل روابط بازگشتی روش تکرار با جای‌گذاری

مثال ۱۴-۱: رابطه بازگشتی زیر را در نظر بگیرید:

$$T(n) = ۲T\left(\left\lfloor n/۲ \right\rfloor\right) + d \qquad (۱-۱)$$

که در آن d یک ثابت زمانی می‌باشد. روش تکرار با جای‌گذاری را برای رابطه بازگشتی (۱-۱) به‌صورت زیر به‌کار می‌بریم:

$$\begin{aligned}
T(n) &= ۲T\left(\left\lfloor n/۲ \right\rfloor\right) + d \\
&= ۴T\left(\frac{\left\lfloor n/۲ \right\rfloor}{۲}\right) + ۳d \\
&\le ۴T\left(n/۴\right) + ۳d \\
&= \ldots \\
&\le ۲^i T\left(n/۲^i\right) + \left(۲^i - ۱\right)d
\end{aligned}$$

$$\frac{n}{۲^i} = ۱ \Rightarrow i = \log_۲ n$$

$$T(n) \le ۲^{\log_۲ n} T(۱) + \left(۲^{\log_۲ n} - ۱\right) d$$

(۱-۲)

حل روابط بازگشتی روش تکرار با جای‌گذاری

مثال ۱۴-۱: رابطه بازگشتی زیر را در نظر بگیرید:

$$T(n) = ۲T\left(\left\lfloor n/۲ \right\rfloor\right) + d \qquad (۱-۱)$$

که در آن d یک ثابت زمانی می‌باشد. روش تکرار با جای‌گذاری را برای رابطه بازگشتی (۱-۱) به‌صورت زیر به‌کار می‌بریم:

$$\begin{aligned} T(n) &= ۲T\left(\left\lfloor n/۲ \right\rfloor\right) + d \\ &= ۴T\left(\frac{\left\lfloor n/۲ \right\rfloor}{۲}\right) + ۳d \\ &\le ۴T\left(n/۴\right) + ۳d \\ &= \ldots \\ &\le ۲^{i}T\left(n/۲^{i}\right) + \left(۲^{i} - ۱\right)d \end{aligned}$$

$$\frac{n}{۲^{i}} = ۱ \Rightarrow i = \log_{۲} n$$

$$T(n) \le ۲^{\log_{۲} n} T(۱) + \left(۲^{\log_{۲}^{n}} - ۱\right) d$$

$$T(n) \in \mathcal{O}(n) \qquad (۱-۲)$$

حل روابط بازگشتی روش تکرار با جای‌گذاری

مثال ۱۵-۱: رابطه بازگشتی زیر را در نظر بگیرید:

$$T(n) \le \begin{cases} C_1 & \text{if} \quad n = 1 \\ 2T\left(n/2\right) + C_2 n & \text{if} \quad n > 1 \end{cases} \qquad (1-3)$$

حل روابط بازگشتی روش تکرار با جای‌گذاری

مثال ۱۵-۱: رابطه بازگشتی زیر را در نظر بگیرید:

$$T(n) \le \begin{cases} C_1 & \text{if} \quad n = 1 \\ 2T(n/2) + C_2 n & \text{if} \quad n > 1 \end{cases} \qquad (1-3)$$

$$\begin{aligned} T(n) &\le 2T(n/2) + C_2 n \\ &\le 2\left(2T(n/4) + C_2 n/2\right) + C_2 n \\ &\le 4T(n/4) + 2C_2 n \\ &\vdots \\ &\le 2^i T(n/2^i) + iC_2 n \end{aligned}$$

حل روابط بازگشتی روش تکرار با جای‌گذاری

مثال ۱۵-۱: رابطه بازگشتی زیر را در نظر بگیرید:

$$T(n) \le \begin{cases} C_1 & \text{if} \quad n = 1 \\ 2T(n/2) + C_2 n & \text{if} \quad n > 1 \end{cases} \qquad (1\text{-}3)$$

$$\begin{aligned} T(n) &\le 2T(n/2) + C_2 n \\ &\le 2\left(2T(n/4) + C_2 n/2\right) + C_2 n \\ &\le 4T(n/4) + 2C_2 n \\ &\vdots \\ &\le 2^i T(n/2^i) + iC_2 n \end{aligned}$$

$$\frac{n}{2^i} = 1 \Rightarrow i = \log_2 n$$

$$T(n) \le 2^{\log n} T(1) + C_2 n \log n \Rightarrow T(n) \in \mathcal{O}(n \log n)$$

پایان فصل اول