

www.Prozheha.ir



با ظاهر جدید و امکانات جدید

# پروژه ها

بزرگترین مرجع دانلود کتاب، جزوه و منابع دانشگاهی رایگان

# فصل ۱

## مفاهیم مبنا و مقدمه‌ای بر تحلیل الگوریتم‌ها

مطالعه الگوریتم‌ها زمینه‌های متعددی را در برمی‌گیرد. در زیر به چند نمونه اشاره می‌کنیم که می‌توان آن‌ها را چرخه حیات یک الگوریتم نامید.

**(الف) طراحی الگوریتم‌ها:** روش‌های مختلفی برای طراحی الگوریتم‌ها وجود دارند که اتفاقاً جزء سیلابس این درس هم هستند و عبارت‌اند از: روش‌های تقسیم و غلبه، روش‌های حریصانه، روش‌های برنامه‌نویسی پویا، روش‌های پسگرد و روش‌های انشعاب و تحدید. **(ب) معتبرسازی یا اثبات درستی الگوریتم‌ها:** بعد از طراحی باید اثبات شود که الگوریتم مزبور درست است. الگوریتمی درست است که به ازای هر ورودی مناسب خروجی صحیحی بدهد. اثبات درستی الگوریتم‌ها به اثبات قضایا در ریاضی می‌ماند و مرحله بسیار مهمی در زمینه مطالعه الگوریتم‌ها است، اما در این درس با این قسمت زیاد سروکار نخواهیم داشت.

**(ج) تحلیل الگوریتم‌ها (تحلیل مقدم، ارزیابی کارایی الگوریتم‌ها):** یک الگوریتم در زمان اجراء از CPU کامپیوتر برای اجرای دستورالعمل‌ها و از حافظه برای ذخیره‌سازی برنامه و داده‌ها استفاده می‌کند. منظور از تحلیل یک الگوریتم فرآیندی است که مشخص می‌کند یک الگوریتم در زمان اجراء چه مدت زمان از CPU برای اجرای دستورالعمل‌ها (پیچیدگی زمانی) و چه مقدار از حافظه (چه اصلی و چه جانبی) برای ذخیره‌سازی برنامه و داده‌ها (پیچیدگی فضایی) نیاز دارد.

**(د) پیاده‌سازی الگوریتم‌ها:** پیاده‌سازی یک الگوریتم نوشتن آن به زبان برنامه‌نویسی خاص است که معمولاً بعد از تحلیل مقدم آن صورت می‌گیرد و نام برنامه به آن اطلاق می‌شود.

**(هـ) تست برنامه:** تست یک برنامه شامل  $i$ : اشکال‌زدایی و  $ii$ : تحلیل موخر (اندازه‌گیری کارایی) است. اندازه‌گیری کارایی عبارت است از فرآیند اجرای الگوریتم صحیح بر روی داده‌های نمونه‌گیری شده برای به دست آوردن زمان و حافظه موردنیاز توسط کامپیوتر. زمان اجرای یک الگوریتم به پارامترهای مختلفی بستگی دارد که از جمله می‌توان به نوع دستورالعمل‌ها (دستورالعمل‌های جمع، ضرب، نوشتن، خواندن، شرطی و...)، کامپایلر مورد استفاده، زبان برنامه‌نویسی، سخت‌افزار به کار رفته و پارامتری مثل  $n$  که می‌تواند معرف تعداد ورودی‌ها یا خروجی‌ها و یا هر دو باشد، اشاره کرد؛ با این وجود در تحلیل مقدم یک الگوریتم، پارامترهای مربوط به کامپایلر، سخت‌افزار و... نادیده گرفته و زمان اجرای یک الگوریتم را به صورت زیر در نظر خواهیم گرفت:

$$t_{P(n)} = \sum_i t_i f_i$$

دستورالعمل  $i$

که در آن،  $n$  تعداد ورودی‌ها (یا خروجی‌ها و یا هر دو)،  $t_i$  زمان اجرای دستور  $i$  و  $f_i$  تعداد دفعات اجرای دستورالعمل  $i$  است. حال اگر برای سهولت بیشتر فرض کنیم که زمان اجراء، برای همه دستورالعمل‌ها برابر واحد زمان  $\text{cpu}$  است، در این صورت داریم:

$$t_{P(n)} = \sum_i f_i$$

دستورالعمل  $i$

با این وجود، همان‌گونه که در مثال‌های زیر مشاهده خواهیم کرد، شاید مفهوم دستورالعمل در یک الگوریتم خیلی روشن نباشد.

**مثال ۱:**  $t_{P(n)}$  برای قطعه برنامه زیر کدام است؟

1) for  $i=1$  to  $n$  do

2)  $z=x+y$

صرف‌نظر از این‌که این قطعه برنامه را برای مشخص کردن دستورالعمل‌ها زیر ذره‌بین ببریم (مثلاً سطر ۲ را معادل ۲ دستورالعمل در نظر بگیریم یا یک دستورالعمل) چیزی که مسلم است  $t_{P(n)}$  به صورت  $a n + b$  برای  $a, b > 0$  صحیح خواهد بود. به عبارت دیگر، زمان اجرای این قطعه برنامه یک تابع خطی از  $n$  خواهد بود.

**مثال ۲:**  $t_{P(n)}$  را برای قطعه برنامه زیر به دست آورید:

for  $i = 1$  to  $n$   
 for  $j = 1$  to  $n$  do  
 $z = x + y$

مشابه توضیحی که برای مثال ۱ داده شد،  $t_{P(n)}$  برای این قطعه برنامه به صورت  $a n^2 + b n + c$  برای  $a > 0$ ، خواهد بود (چند جمله‌ای درجه ۲ بر حسب  $n$ )  
 این دو مثال ساده ما را به نمادهای مجانبی زیر رهنمون می‌کنند:

## نمادهای مجانبی $O, \theta, \Omega$

**تعریف  $O$**  (بررسی بدترین حالت زمان اجرای یک الگوریتم)

اگر  $f(n)$  زمان اجرای یک الگوریتم باشد، گوییم که  $f(n) = O(g(n))$  اگر  $\exists c_1 > 0$  و  $n$  طبیعی، به گونه‌ای که

$$\forall n \geq n_o \Rightarrow f(n) \leq c_1 g(n)$$

**مثال ۳:** نشان دهید که  $2n^2 + 7n + 5 = O(n^2)$

اگر  $c_1 = 3$  باشد داریم:

$$2n^2 + 7n + 5 \leq 3n^2 \Leftrightarrow 7n + 5 \leq n^2$$

بنابراین کافی است  $n_o = 8$

**تعریف  $\Omega$**  (بررسی بهترین حالت زمان اجرای یک الگوریتم)

$f(n) = \Omega(g(n))$   $\Leftrightarrow \exists c_2 > 0, n_o$  به گونه‌ای که

$$\forall n \geq n_o \Rightarrow f(n) \geq c_2 g(n)$$

**مثال ۴:** نشان دهید که  $2n^2 + 7n + 5 = \Omega(n^2)$

کافی است  $c_2 = 1$  باشد، زیرا رابطه  $2n^2 + 7n + 5 \geq n^2$  همواره برای هر  $n \geq n_o = 1$  برقرار است.

تعریف  $\theta$  (بررسی حالت میانگین زمان اجرای یک الگوریتم)

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

$$\exists c_1, c_2 > 0 \text{ و } n_0 \text{ طبیعی} \Rightarrow \forall n \geq n_0 \quad c_1 g(n) \leq f(n) \leq c_2 g(n)$$

مثال ۵: نشان دهید که  $2n^2 + 7n + 5 = \theta(n^2)$

حل : از مثال‌های ۴ و ۵ نتیجه می‌شود.

لم ۱: از تعریف  $\theta$  نتیجه می‌شود که  $f(n) = \Omega(g(n)) \Leftrightarrow f(n) = O(g(n))$  و  $f(n) = \theta(g(n))$

هر چند از تعریف  $O$ ، می‌توان برای مثال بالا رابطه زیر را داشت:

$$n_0 = 8 \Rightarrow 2n^2 + 7n + 5 \leq 3n^2 \leq 3n^2 \leq 3n^4 \leq \dots$$

بنابراین هر چند  $2n^2 + 7n + 5 = O(n^4)$  یا  $2n^2 + 7n + 5 = O(n^3)$  با وجود این همان  $O(n^2)$  را خواهیم پذیرفت. به عبارت دیگر، کوچک‌ترین کرانه بالایی زمان اجراء مورد نظر خواهد بود. عین همین مطلب در مورد  $\Omega$  هم صادق خواهد بود، یعنی هر چند طبق تعریف می‌توانیم داشته باشیم  $2n^2 + 7n + 5 = \Omega(n)$ ، با این وجود، همان  $2n^2 + 7n + 5 = \Omega(n^2)$  پذیرفته خواهد شد، به عبارت دیگر بزرگ‌ترین کرانه بالایی مدنظر خواهد بود.

قضیه : اگر زمان اجرای الگوریتمی به صورت  $A(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n + a$

$$A(n) = \theta(n^m)$$

در این صورت

لم: اگر برنامه‌ای داشته باشیم که از  $k$  قطعه برنامه مستقل از هم با مرتبه‌های بزرگی  $O(n^{m_1}), O(n^{m_2}), \dots, O(n^{m_k})$  تشکیل شده باشد، در این صورت مرتبه بزرگی خود برنامه برابر  $O(n^{\max\{m_1, m_2, \dots, m_k\}})$  خواهد بود.

## تحلیل الگوریتم مرتب‌سازی حبابی

ورودی: آرایه نامرتب  $A[1..n]$

خروجی: این آرایه مرتب شده به صورت صعودی.

Algorithm Bubblesort (A, n)

```

1 for i=1 to n-1 do
2   for j=n downto i+1 do
3     if A[j-1] > A[j] then
4       {
5         temp = A[j-1]
6         A[j-1] = A[j]
7         A[j] = temp
8       }
9   }
10 }
```

بررسی بدترین حالت: بدترین حالت زمانی اتفاق می‌افتد که دستور ۳ همواره درست باشد. این مسأله عملاً زمانی اتفاق می‌افتد که خود آرایه  $A[1..n]$  به صورت نزولی مرتب شده باشد، در این صورت داریم:

تعداد دفعات اجراء	شماره دستورالعمل
$n-1$	1
$(n-1) + (n-2) + \dots + 2 + 1$	2
$(n-1) + (n-2) + \dots + 2 + 1$	3
$(n-1) + (n-2) + \dots + 2 + 1$	4
$(n-1) + (n-2) + \dots + 2 + 1$	5
$(n-1) + (n-2) + \dots + 2 + 1$	6
جمع $\Leftarrow t(n) = n-1 + 5 \frac{n(n-1)}{2}$	

$$t(n) = O(n^2) \quad \text{بنابراین}$$

**بررسی بهترین حالت:** بهترین حالت زمانی اتفاق می افتد که دستور ۳ هیچ وقت اجراء نشود و این زمانی اتفاق می افتد که خود آرایه به صورت صعودی مرتب شده باشد، در این حالت داریم:

تعداد دفعات اجراء	شماره دستورالعمل
$n-1$	1
$(n-1) + (n-2) + \dots + 2 + 1$	2
$(n-1) + (n-2) + \dots + 2 + 1$	3
0	4
0	5
0	6
جمع $\Rightarrow t(n) = n-1 + 2 \frac{n(n-1)}{2} = \Omega(n^2)$	

لذا از لم ۱ نتیجه می شود که  $t(n) = \Omega(n^2)$ .

## تحلیل الگوریتم مرتب سازی درجی

**ورودی:** آرایه نامرتب  $A[1..n]$

**خروجی:** این آرایه مرتب شده به صورت صعودی.

Algorithm Insertion\_Sort (A,n)

$A[0] = -\infty$  / برای سهولت /

for j=2 to n do

{

item = A[j] , i=j-1

while  $\boxed{\text{item} < A[i]}$  \* do

{

A[i+1]=A[i] , i=i-1

}

A[i+1]=item

}

برای سهولت در تحلیل این الگوریتم، فقط تعداد دفعات اجرای دستورالعمل مقایسه‌ای \* را حساب می‌کنیم.

**بررسی بدترین حالت:** بدترین حالت زمانی اتفاق می‌افتد که دستور شرطی همواره درست باشد و این مسأله عملاً زمانی اتفاق

می‌افتد که آرایه  $A[1..n]$  به صورت نزولی مرتب شده باشد.

تعداد دفعات اجرای دستورالعمل	مقدار پارامتر $j$
۲	۱
۳	۲
$\vdots$	$\vdots$
$n$	$n$

$$\text{جمع } t(n) = 2 + 3 + \dots + n = \frac{n(n+1)}{2} - 1 = O(n^2)$$

**بررسی بهترین حالت:** بهترین حالت زمانی اتفاق می‌افتد که حلقه داخلی هیچ وقت اجرا نشود. (آرایه به صورت صعودی مرتب

شده باشد)

تعداد دفعات اجرای دستورالعمل	مقدار پارامتر $j$
۱	۲
۱	۳
$\vdots$	$\vdots$
$n$	$n$

$$\text{جمع } t(n) = 1 + 1 + \dots + 1 = n - 1 = \boxed{\Omega(n)}$$

## مرتبه‌های بزرگی متداول برای الگوریتم‌ها

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < \dots < O(2^n)$$

با توجه به مطلب فوق نتیجه می‌شود که میانگین زمان اجرای الگوریتم مرتب‌سازی درجی  $O(n)$ ،  $O(n \log n)$  یا  $O(n^2)$

است که اثبات می‌شود برابر  $\boxed{\theta(n^2)}$  است.

یکی از مهم‌ترین روش‌هایی که برای تحلیل الگوریتم‌ها به کار می‌رود، روابط بازگشتی است. در ذیل، نخست به چند مثال از روابط بازگشتی پرداخته و سپس روش‌های حل آن‌ها را بررسی خواهیم کرد.

### مثال ۱: تعداد مقایسه‌ها در الگوریتم مرتب‌سازی حبابی

تعداد مقایسه‌های انجام شده در الگوریتم مرتب‌سازی حبابی را می‌توان به صورت رابطه بازگشتی زیر بیان کرد. اگر فرض کنیم که  $a_n$  تعداد مقایسه‌های لازم توسط این الگوریتم برای مرتب‌سازی یک آرایه  $n$  عنصری باشد، در این صورت برای قرار دادن کوچک‌ترین عنصر در مکان اول،  $(n-1)$  مقایسه انجام می‌شود. اکنون آرایه  $n-1$  عنصری  $A[2..n]$  باید به همان صورت قبل، مرتب شود که بنا به فرض تعداد مقایسه‌های لازم، برابر  $a_{n-1}$  خواهد بود، لذا:

$$\begin{cases} a_n = a_{n-1} + n - 1 & n \geq 1 \\ a_0 = 0 \end{cases}$$

$a_0 = 0$  را شرط مرزی رابطه بازگشتی گوئیم.

### مثال ۲: مسأله تعداد جابه‌جایی‌ها در برج هانوی

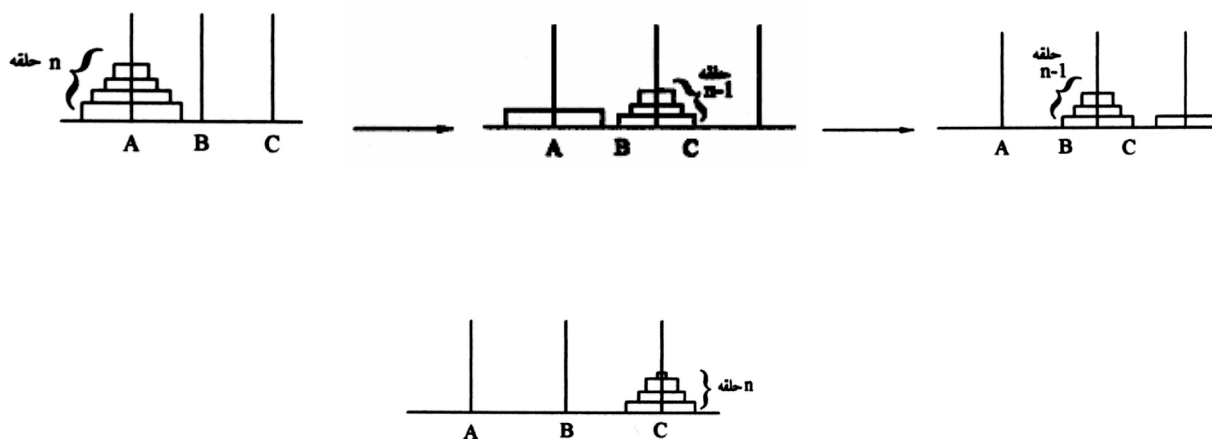
تعداد دفعات اجرای دستور \* در الگوریتم زیر را، که برای جابه‌جایی‌های  $n$  حلقه واقع در ستون  $A$  به ستون  $C$  نوشته شده، به دست آورید. فرض کنید که  $a_n$  تعداد دفعات اجراء باشد مسلم است که  $a_n$  برابر تعداد جابه‌جایی‌های به کار رفته در مسأله معروف برج هانوی و همچنین مرتبه بزرگی الگوریتم هانوی نیز هست.

Algorithm Hanoi ( $n, A, B, C$ ) /  $n$  تعداد حلقه‌ها و  $A, B, C$  از نوع کاراکتر هستند.

```
if  $n \geq 1$  then
{
  Hanoi ( $n-1, A, C, B$ )
  * Write Ln ( $A, " \rightarrow " , C$ )
  Hanoi ( $n-1, B, A, C$ )
}
```

ایده‌ای که برای نوشتن الگوریتم مزبور به کار رفته، استقرای ریاضی است. اگر  $n=1$  باشد، بدیهی است که با یک جابه‌جایی می‌توان حلقه واقع در ستون  $A$  را به ستون  $C$  برد (پایه استقراء). حال فرض کنید (فرض استقراء) که  $n-1$  حلقه بالایی را توسط ستون کمکی  $C$  به ستون  $B$  برده باشیم (تعداد این جابه‌جایی‌ها بنا به تعریف برابر  $a_{n-1}$  است)، اکنون حلقه زیرین را با یک انتقال به ستون  $C$  می‌بریم. سرانجام  $n-1$  حلقه واقع در ستون  $B$  را این دفعه با ستون کمکی  $A$  به  $C$  می‌بریم ( $a_{n-1}$  جابه‌جایی دیگر). بنابراین مجموع جابه‌جایی‌های انجام شده عبارت است از:

$$\begin{cases} a_n = 2a_{n-1} + 1 & n \geq 1 \\ a_1 = 1 \end{cases}$$



### مثال ۳: دنباله اعداد فیبوناچی

می‌دانیم که این دنباله به صورت زیر است:

$n$	۰	۱	۲	۳	۴	۵	۶	۷	۸	...
$F_n$	۰	۱	۱	۲	۳	۵	۸	۱۳	۲۱	...

رابطه بازگشتی برای این دنباله به صورت زیر است:

$$\begin{cases} F_n = F_{n-1} + F_{n-2} & n \geq 2 \\ F_1 = 0 & F_0 = 1 \end{cases}$$

## حل روابط بازگشتی

یک رابطه بازگشتی را می‌توان به سه صورت زیر حل کرد:

الف: جایگذاری با تکرار

ب: استفاده از سری مولد

ج: استفاده از معادله مشخصه

در ادامه برای هر کدام از سه روش چند مثال می‌آوریم.

### الف) حل روابط بازگشتی با استفاده از روش جایگذاری با تکرار

**مثال :** تعداد مقایسه‌ها در الگوریتم مرتب‌سازی حبابی برای مرتب‌سازی یک آرایه  $n$  عنصری

دیدیم که این تعداد مقایسه به صورت رابطه بازگشتی زیر بیان می‌شود:

$$\begin{cases} a_n = a_{n-1} + n - 1 & n \geq 1 \\ a_1 = 0 \end{cases}$$

یا

$$a_n = n - 1 + a_{n-1}$$

حال در رابطه بالا به جای  $a_{n-1}$  می‌توان  $n - 2 + a_{n-2}$  را قرار داد:

$$a_n = (n - 1) + (n - 2) + a_{n-2}$$

و به جای  $a_{n-2}$  هم می‌توان  $n - 3 + a_{n-3}$  را قرار داد:

$$a_n = (n - 1) + (n - 2) + (n - 3) + a_{n-3}$$

با ادامه این روند

$$a_n = (n - 1) + (n - 2) + (n - 3) + \dots + 2 + 1 + 0 + a_1$$

لذا

$$a_n = \frac{n(n-1)}{2}$$

البته قبلاً هم دیده بودیم که زمان اجرای الگوریتم مرتب‌سازی حبابی  $\theta(n^2)$  است.

**مثال :** تعداد جابه‌جایی‌ها در مسالهٔ برج هانوی برای جابه‌جایی  $n$  حلقه واقع در یک ستون به ستون دیگر

دیدیم که:

$$\begin{cases} a_n = 2a_{n-1} + 1 & n \geq 1 \\ a_1 = 0 \end{cases}$$



اگر مشابه مثال قبل عمل کنیم خواهیم داشت:

$$\begin{aligned} a_n &= 1 + 2a_{n-1} \\ a_n &= 1 + 2(1 + 2a_{n-2}) = 1 + 2 + 2^2 a_{n-2} \\ &= 1 + 2 + 2^2 (1 + 2a_{n-3}) = 1 + 2 + 2^2 + 2^3 a_{n-3} \end{aligned}$$

با ادامه این روند

$$a_n = 1 + 2 + 2^2 + 2^3 + \dots + 2^{n-1} + 2^n a_1$$

یا

$$a_n = \frac{2^n - 1}{2 - 1} \Rightarrow \boxed{a_n = 2^n - 1}$$

بنابراین مرتبه بزرگی الگوریتم هانوی برابر  $\boxed{\theta(2^n)}$  است.

**تمرین:** تعداد فراخوانی‌های الگوریتم هانوی را به دست آورید.

روش جایگذاری با تکرار معمولاً برای حل روابط بازگشتی مرتبه اول به کار می‌رود. در حالت عمومی یک رابطه بازگشتی مرتبه k به صورت زیر تعریف می‌شود:

$$c_n a_n + c_{n-1} a_{n-1} + \dots + c_{n-k} a_{n-k} = f(n)$$

اگر  $f(n)=0$ ، رابطه بازگشتی، همگن (متجانس) نامیده می‌شود.

**ب) حل روابط بازگشتی با استفاده از سری‌های مولد**

سری مولد برای یک رابطه بازگشتی که جمله عمومی آن مثلاً با  $a_n$  مشخص شده باشد، به وسیله  $A(x) = \sum_{n=0}^{\infty} a_n x^n$  تعریف می‌شود.

**مثال:** رابطه بازگشتی زیر را با استفاده از سری مولد حل کنید: (رابطه بازگشتی مربوط به الگوریتم مرتب‌سازی حبابی)

$$\begin{cases} a_n = a_{n-1} + n - 1 & n \geq 1 \\ a_0 = 0 \end{cases}$$

روش عمومی به این صورت است که اول سری مولد را تشکیل می‌دهیم:

$$A(x) = \sum_{n=0}^{\infty} a_n x^n \quad (1)$$

شرایط مرزی را جدا کرده و از روی تعریف رابطه بازگشتی مقدار  $a_n$  را قرار می‌دهیم:

$$A(x) = a_0 + \sum_{n=1}^{\infty} a_n x^n = 0 + \sum_{n=1}^{\infty} (a_{n-1} + n - 1) x^n$$

سپس جمع مزبور را تفکیک می‌کنیم:

$$\begin{aligned} A(x) &= x \left( \sum_{n=1}^{\infty} a_{n-1} x^{n-1} \right) + \sum_{n=1}^{\infty} (n-1) x^n \\ &= x A(x) + x^2 \sum_{n=1}^{\infty} n x^{n-1} \end{aligned}$$

اما

$$\sum_{n=1}^{\infty} nx^{n-1} = 1 + 2x + 3x^2 + 4x^3 + \dots = \frac{d}{dx}(1 + x + x^2 + x^3 + \dots)$$

$$= \frac{d}{dx}\left(\frac{1}{1-x}\right) = \frac{1}{(1-x)^2}$$

لذا

$$A(x) = xA(x) + \frac{x^2}{(1-x)^2} \Rightarrow A(x) = \frac{x^2}{(1-x)^2}$$

اما از روی فرمول  $\frac{1}{(1-x)^m} = \sum_{n=0}^{\infty} C_{n+m-1}^n x^n$  با قرار دادن  $m=3$  داریم:

$$\frac{1}{(1-x)^2} = \sum_{n=0}^{\infty} C_{n+2}^n x^n$$

لذا

$$A(x) = x^2 \sum_{n=0}^{\infty} a_n x^n \Rightarrow$$

$$A(x) = \sum_{n=0}^{\infty} C_{n+2}^n x^{n+2} \quad (2)$$

از مقایسه (۱) و (۲) نتیجه می‌شود که

$$a_n = C_n^{n-2} = \frac{n(n-1)}{2}$$

این همان نتیجه‌ای است که در روش جایگذاری با تکرار به آن رسیده بودیم.

**مثال:** رابطه بازگشتی را با استفاده از سری مولد حل کنید. (رابطه بازگشتی مربوط به مساله برج هانوی)

$$\begin{cases} a_n = 2a_{n-1} + 1 & n \geq 1 \\ a_0 = 0 \end{cases}$$

هانوی)

$$A(x) = \sum_{n=0}^{\infty} a_n x^n \quad *$$

اگر مشابه مثال قبل عمل کنیم خواهیم داشت:

$$A(x) = \sum_{n=0}^{\infty} a_n x^n = a_0 + \sum_{n=1}^{\infty} (2a_{n-1} + 1) x^n =$$

$$2xA(x) + \sum_{n=1}^{\infty} x^n = 2xA(x) + \frac{x}{1-x} \Rightarrow$$

$$A(x) = \frac{x}{(1-2x)(1-x)}$$

حال کسر مزبور را به صورت کسرهای ساده تر تجزیه می کنیم:

$$\frac{x}{(1-2x)(1-x)} = \frac{a}{1-2x} + \frac{b}{1-x} = \frac{a(1-x) + b(1-2x)}{(1-2x)(1-x)}$$

با مساوی قرار دادن صورت کسرهای اول و آخر داریم:

$$x = a(1-x) + b(1-2x)$$

رابطه مزبور باید به ازای هر مقدار از  $x$  برقرار باشد. از جمله با قرار دادن  $x = \frac{1}{2}$  و  $x=1$  به ترتیب خواهیم داشت  $b = -1, a = 1$ : لذا:

$$A(x) = \frac{1}{1-2x} - \frac{1}{1-x} = \sum_{n=0}^{\infty} 2^n x^n - \sum_{n=0}^{\infty} x^n \Rightarrow$$

$$A(x) = \sum_{n=0}^{\infty} (2^n - 1) x^n \quad **$$

از \* و \*\* نتیجه می شود که

$$a_n = 2^n - 1$$

باز هم همان نتیجه ای است که از روش جایگذاری با تکرار به آن رسیده بودیم.

مثال: رابطه بازگشتی  $\begin{cases} F_n = F_{n-1} + F_{n-2} & n \geq 2 \\ F_0 = 0, F_1 = 1 \end{cases}$  را با استفاده از سری مولد حل کنید.

$$F(x) = \sum_{n=0}^{\infty} F_n x^n$$

داریم:

$$F(x) = \sum_{n=0}^{\infty} F_n x^n = F_0 + F_1 x + \sum_{n=2}^{\infty} F_n x^n$$

$$F(x) = x + \sum_{n=2}^{\infty} (F_{n-1} + F_{n-2}) x^n$$

$$= x + x \left( \sum_{n=2}^{\infty} F_{n-1} x^{n-1} \right) + x^2 \left( \sum_{n=2}^{\infty} F_{n-2} x^{n-2} \right)$$

$$= x + x [F(x) - F_0] + x^2 F(x)$$

$$= x + xF(x) + x^2 F(x) \Rightarrow$$

$$F(x) = \frac{x}{1-x-x^2}$$

حال اگر تعریف کنیم  $\bar{\varphi} = \frac{1-\sqrt{5}}{2}$ ,  $\varphi = \frac{1+\sqrt{5}}{2}$ ، کسر مزبور را می توان به صورت زیر تجزیه کرد:

$$\frac{x}{1-x-x^2} = \frac{a}{1-\varphi x} + \frac{b}{1-\bar{\varphi} x} = \frac{a(1-\bar{\varphi} x) + b(1-\varphi x)}{(1-\varphi x)(1-\bar{\varphi} x)}$$

با مساوی قرار دادن صورت های کسرهای اول و آخر خواهیم داشت:

$$x = a(1-\bar{\varphi} x) + b(1-\varphi x)$$

با قرار دادن  $x = 0, x = \frac{1}{\varphi}$  خواهیم داشت:

$$b = \frac{-1}{\sqrt{5}}, a = \frac{1}{\sqrt{5}}$$

بنابراین

$$F(x) = \frac{1}{\sqrt{5}} \left[ \sum_{n=0}^{\infty} \varphi^n x^n - \sum_{n=0}^{\infty} \bar{\varphi}^n x^n \right]$$

$$F(x) = \frac{1}{\sqrt{5}} \sum_{n=0}^{\infty} (\varphi^n - \bar{\varphi}^n) x^n$$

در نتیجه

$$F_n = \frac{1}{\sqrt{5}} (\varphi^n - \bar{\varphi}^n)$$

یا

$$F_n = \frac{1}{\sqrt{5}} \left[ \left( \frac{1+\sqrt{5}}{2} \right)^n - \left( \frac{1-\sqrt{5}}{2} \right)^n \right]$$

اما  $\frac{1+\sqrt{5}}{2} \cong 1.61$  ,  $\frac{1-\sqrt{5}}{2} \cong -0.61$  , بنابراین با توجه به این که  $F_n$  ها اعداد طبیعی هستند، لذا می‌توان از جمله

در عبارت بالا صرف‌نظر کرده و  $F_n$  را به صورت زیر به دست آورد:

$$F_n = \text{Round} \left( \frac{1}{\sqrt{5}} \left( \frac{1+\sqrt{5}}{2} \right)^n \right)$$

بنابراین مرتبه رشد دنباله فیبوناچی به صورت  $(1.61)^n$  است.

**مثال :** مرتبه بزرگی الگوریتم بازگشتی زیر را، که برای محاسبه جمله  $n$ ام دنباله فیبوناچی نوشته شده است، به دست آورید.

Algorithm Fib(n)

if  $n \leq 1$  then Return(n)

else Return (Fib(n-1) + Fib(n-2))

کافی است تعداد فراخوانی‌های Fib(n) را به دست آوریم. فرض  $b_n$  برابر با تعداد فراخوانی‌های Fib(n) باشد، بدیهی است که:

$$\begin{cases} b_n = 1 + b_{n-1} + b_{n-2} & n \geq 2 \\ b_2 = 1, b_1 = 1 \end{cases}$$

حال اگر تعریف کنیم  $a_n = b_n + 1$  خواهیم داشت:

$$\begin{cases} a_n = a_{n-1} + a_{n-2} & n \geq 2 \\ a_2 = 2, a_1 = 2 \end{cases}$$

که مشابه رابطه بازگشتی برای دنباله فیبوناچی است (یک رابطه بازگشتی همگن مرتبه ۲). از حل آن نتیجه می‌شود:

$$b_n = a_n - 1 \Rightarrow$$

$$b_n = 2F_{n+1} - 1$$

اما دیدیم که  $F_n \cong (1.61)^n$  لذا مرتبه بزرگی الگوریتم مزبور به صورت زیر است:

$$b_n = \theta(1.61)^n$$

مثال : یک الگوریتم از مرتبه  $\theta(n)$  برای محاسبه جمله  $n$  دنباله فیبوناچی ارائه دهید:  
حل:

```
Algorithm LFib(n)
x=0
y=1
For k=2 to n do
{
    z=x+y
    x=y
    y=z
}
Return (z)
```

(ج) حل روابط بازگشتی با استفاده از معادله مشخصه

دیدیم که یک رابطه بازگشتی مرتبه  $k$  به صورت زیر تعریف می شود.

$$\begin{cases} c_n a_n + c_{n-1} a_{n-1} + \dots + c_{n-k} a_{n-k} = f(n) \\ \text{به همراه } k \text{ شرط مرزی} \end{cases}$$

که در آن باید  $c_n \neq 0$  ,  $c_{n-k} \neq 0$ .

معادله مشخصه برای رابطه بازگشتی بالا به صورت زیر تعریف می شود:

$$c_n r^k + c_{n-1} r^{k-1} + c_{n-2} r^{k-2} + \dots + c_{n-k} = 0 \quad (1)$$

در این جا فقط رابطه بازگشتی همگن مرتبه ۲ را بررسی می کنیم. در رابطه (۱)، اگر  $k=2$  باشد خواهیم داشت:

$$c_n r^2 + c_{n-1} r + c_{n-2} = 0 \quad \text{معادله مشخصه یک رابطه بازگشتی همگن مرتبه ۲}$$

که یک چند جمله ای از درجه ۲ است.

فرض کنید  $r_1, r_2$  ریشه های این چند جمله ای باشند. سه حالت امکان دارد:

الف) اگر  $r_1 \neq r_2$  و حقیقی باشند، در این صورت جواب رابطه بازگشتی مزبور به صورت زیر است:

$$a_n = c_1 r_1^n + c_2 r_2^n$$

ب) اگر  $r_1 = r_2 = r$  (ریشه مضاعف)، در این صورت:

$$a_n = (c_1 + n c_2) r^n$$

ج) اگر  $r_1 \neq r_2$  و مختلط باشند، در این حالت:

$$a_n = c_1 r_1^n + c_2 r_2^n$$

برای هر کدام از این حالت‌ها، مثالی می‌آوریم.

**مثال :** رابطه بازگشتی زیر را با استفاده از معادله مشخصه حل کنید:

$$\begin{cases} F_n = F_{n-1} + F_{n-2} & n \geq 2 \\ F_1 = 1, F_0 = 0 \end{cases}$$

معادله مشخصه آن به صورت  $r^2 - r - 1 = 0$  و ریشه‌های آن عبارت‌اند از:

$$r_1, r_2 = \frac{1 \pm \sqrt{5}}{2}$$

لذا

$$F_n = c_1 \left( \frac{1 + \sqrt{5}}{2} \right)^n + c_2 \left( \frac{1 - \sqrt{5}}{2} \right)^n$$

که با اعمال شرایط مرزی  $F_1 = 1, F_0 = 0$  خواهیم داشت:

$$\begin{cases} 0 = c_1 + c_2 \\ 1 = \frac{\sqrt{5}}{2} (c_1 - c_2) \end{cases}$$

لذا

$$c_2 = -\frac{1}{\sqrt{5}}, c_1 = \frac{1}{\sqrt{5}}$$

در نتیجه:

$$F_n = \frac{1}{\sqrt{5}} \left[ \left( \frac{1 + \sqrt{5}}{2} \right)^n - \left( \frac{1 - \sqrt{5}}{2} \right)^n \right]$$

که همان نتیجه حاصل از روش سری مولد است.

**مثال :** رابطه بازگشتی  $\begin{cases} a_n = 4a_{n-1} - 4a_{n-2} & n \geq 2 \\ a_1 = 1, a_0 = 0 \end{cases}$  را با استفاده از معادله مشخصه حل کنید.

**حل:**

معادله مشخصه به صورت  $r^2 - 4r + 4 = 0$  است که ریشه‌های آن عبارت‌اند از  $r_1 = r_2 = 2$  بنابراین:

$$a_n = (c_1 + nc_2) 2^n$$

با اعمال شرایط مرزی داریم:

$$1 = \frac{c_2}{2}, \quad \boxed{0 = c_1}$$

$$\boxed{a_n = n 2^{n-1}} \quad \text{لذا} \quad \boxed{c_2 = \frac{1}{2}}$$

مثال : رابطه بازگشتی  $\begin{cases} a_n = 2a_{n-1} - 2a_{n-2} & n \geq 2 \\ a_1 = 1 & a_0 = 0 \end{cases}$  را با استفاده از معادله مشخصه حل کنید.

حل:

$r^2 - 2r + 2 = 0$  و ریشه‌های آن عبارت‌اند از  $r_1, r_2 = 1 \pm i$  لذا

$$a_n = c_1 (1+i)^n + c_2 (1-i)^n$$

هر چند ظاهر این عبارت نشان می‌دهد که  $a_n$  یک عدد مختلط است، ولی با توجه به این که شرایط مرزی و نیز ضرایب رابطه بازگشتی اعداد صحیحی هستند، مسلم است که همه  $a_n$  ها اعداد صحیحی هستند. حال، حداقل برای این که ظاهر  $a_n$  به صورت مختلط دیده نشود، از اعداد مختلط کمک می‌گیریم. می‌دانیم که عدد مختلط  $z = x + iy$  را می‌توان به صورت  $z = r(\cos \theta + i \sin \theta)$  بیان کرد که در آن  $r = \sqrt{x^2 + y^2}$  و  $\theta = \arctan \left( \frac{y}{x} \right)$  از طرفی طبق قانون معروف

دموار داریم:

$$(\cos \theta + i \sin \theta)^n = \cos n\theta + i \sin n\theta$$

با اعمال این نتایج بر روی  $(1+i)^n, (1-i)^n$  خواهیم داشت:

$$\begin{cases} 1+i = \sqrt{2} \left( \cos \frac{\pi}{4} + i \sin \frac{\pi}{4} \right) \\ 1-i = \sqrt{2} \left( \cos \frac{\pi}{4} - i \sin \frac{\pi}{4} \right) \end{cases}$$

$$(1+i)^n = (\sqrt{2})^n \left[ \cos \frac{n\pi}{4} + i \sin \frac{n\pi}{4} \right]$$

$$(1-i)^n = (\sqrt{2})^n \left[ \cos \frac{n\pi}{4} - i \sin \frac{n\pi}{4} \right]$$

بنابراین

$$a_n = (\sqrt{2})^n \left\{ c_1 \cos \frac{n\pi}{4} + i c_1 \sin \frac{n\pi}{4} + c_2 \cos \frac{n\pi}{4} - i c_2 \sin \frac{n\pi}{4} \right\}$$

با فرض  $k_1 = c_1 + c_2, k_2 = i(c_1 - c_2)$  خواهیم داشت:

$$a_n = (\sqrt{2})^n \left\{ k_1 \cos \frac{n\pi}{4} + k_2 \sin \frac{n\pi}{4} \right\}$$

حال به جای به دست آوردن ضرایب  $c_1, c_2$ ، ضرایب  $k_1, k_2$  را از رابطه بالا به دست می‌آوریم. با اعمال شرایط مرزی داریم:

$$a_0 = 0 \Rightarrow \boxed{k_1 = 0}$$

$$a_1 = 1 \Rightarrow 1 = (\sqrt{2}) k_2 \sin \frac{\pi}{4} \Rightarrow \boxed{k_2 = 1}$$

لذا

$$\boxed{a_n = (\sqrt{2})^n \sin \frac{n\pi}{4}}$$

همان‌گونه که مشاهده می‌شود، در نتیجه به دست آمده، اثری از ظاهر شدن اعداد مختلط نیست. سری‌های مولد ابزار بسیار سودمندی هستند که می‌توانند برای حل مسائل مشکل‌تر هم، به کار روند. در زیر به چند مثال در این مورد می‌پردازیم:

**مثال :** برای محاسبه  $C_n^m$  از روی رابطه بازگشتی  $\begin{cases} C_n^m = C_{n-1}^{m-1} + C_{n-1}^m & n \geq m \\ C_n^0 = C_n^n = 1 \end{cases}$ ، الگوریتم بازگشتی زیر ارائه می‌شود. تعداد

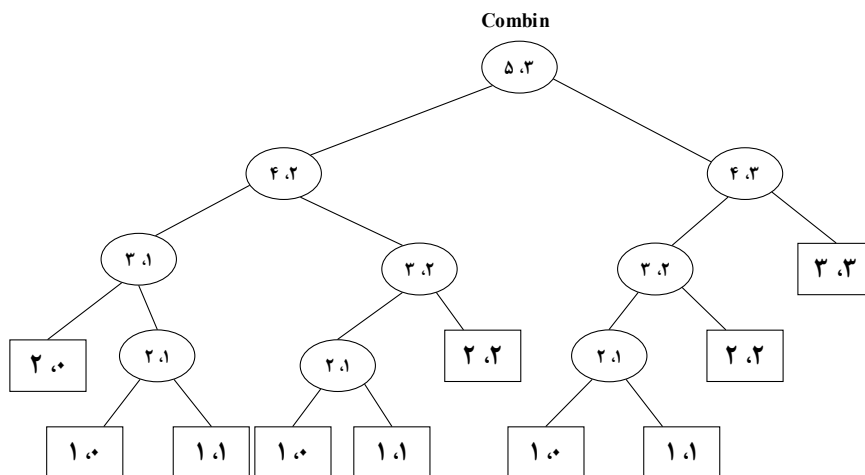
فراخوانی‌های  $\text{Combin}(n, m)$  و در نتیجه مرتبه بزرگی آن را به دست آورید:

Algorithm  $\text{Combin}(n, m)$

if  $(n=m)$  or  $(m=0)$  then Return (1)

else Return  $(\text{Combin}(n-1, m-1) + \text{Combin}(n-1, m))$

مراحل مختلف اجرای این الگوریتم برای  $n=5, m=3$  توسط درخت بازگشت زیر به تصویر کشیده شده است:



که در این حالت تعداد فراخوانی‌ها برابر ۱۹ است.

در حالت عمومی اگر فرض کنیم که  $t(n, m)$  تعداد فراخوانی‌های  $\text{Combin}(n, m)$  باشد،

در این صورت رابطه بازگشتی زیر را داریم:

$$\begin{cases} t(n, m) = t(n-1, m-1) + t(n-1, m) + 1 & n \geq m \\ t(n, n) = 1, \quad t(n, 0) = 1 \end{cases}$$

به طرفین رابطه بالا یک واحد اضافه کرده و  $a(n, m)$  را به صورت زیر تعریف کنیم:

$$a(n, m) = t(n, m) + 1$$

حال سری مولد  $G_n(x)$  را به صورت زیر تعریف می‌کنیم:

$$G_n(x) = \sum_{m=0}^{\infty} a(n, m) x^m \quad (1)$$

$$G_n(x) = a(n, 0) + \sum_{m=1}^{\infty} a(n, m) x^m$$

$$G_n(x) = 2 + \sum_{m=1}^{\infty} [a(n-1, m-1) + a(n-1, m)] x^m$$

$$G_n(x) = 2 + x \left( \sum_{m=1}^{\infty} a(n-1, m-1) x^{m-1} \right) + \sum_{m=1}^{\infty} a(n-1, m) x^m$$



$$G_n(x) = xG_{n-1}(x) + 2 + \sum_{m=1}^{\infty} a(n-1, m)x^m$$

$$G_n(x) = xG_{n-1}(x) + G_{n-1}(x)$$

$$G_n(x) = (1+x)G_{n-1}(x)$$

اگر از روش جایگذاری با تکرار استفاده کنیم:

$$G_n(x) = (1+x)^r G_{n-r}(x)$$

$$= (1+x)^r G_{n-r}(x)$$

⋮

$$= (1+x)^n G_0(x)$$

اما  $G_0(x) = 2$  لذا  $G_n(x) = 2(1+x)^n$ . اما اگر فرض کنیم که برای  $n < m$ ،  $C_n^m = 0$ ، لذا:

$$G_n(x) = 2 \sum_{m=0}^{\infty} C_n^m x^m \quad (2)$$

با مقایسه روابط (1) و (2) نتیجه می شود که:

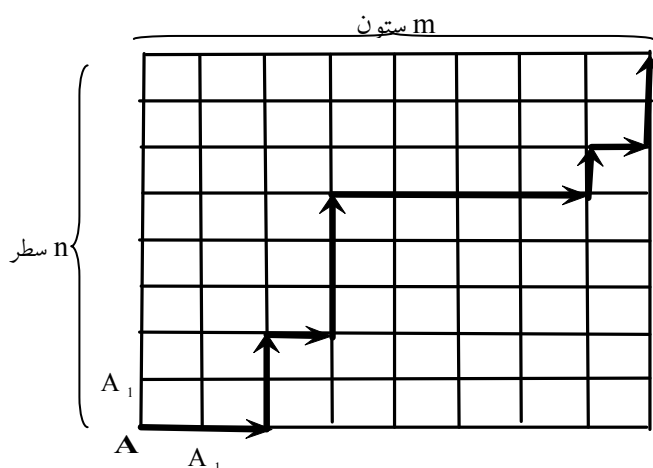
$$a(n, m) = 2C_n^m$$

در نتیجه

$$t(n, m) = 2C_n^m - 1$$

ثابت می شود که در حالت  $m = \frac{n}{2}$ ، این تعداد فراخوانی ها از مرتبه  $\Omega(2^n)$  است.

**مثال :** صفحه شطرنجی زیر مفروض است. می خواهیم از نقطه A به نقطه B برویم. جهت حرکت های مجاز فقط به سمت راست یا بالا هستند. تعداد مسیرهای ممکن از A به B را به دست آورید. یک نمونه از مسیرهای ممکن در شکل مزبور به نمایش در آمده است.



فرض کنیم  $t(n, m)$  تعداد مسیرهایی باشد که در حالت عمومی در یک صفحه شطرنجی  $n \times m$  می توان با توجه به شرایط خواسته شده از نقطه A به نقطه B رفت. برای شروع از نقطه A دو حالت امکان دارد. یا می توان به  $A_1$  رفت که در این صورت تعداد مسیرهای ممکن از  $A_1$  به B از رابطه بازگشتی  $t(n, m-1)$  حاصل خواهد شد. اما اگر از A به  $A_1$  رفتیم، در این صورت تعداد مسیرهای

ممکن از  $A_r$  به  $B$  برابر  $t(n-1, m)$  خواهد بود، لذا رابطه بازگشتی زیر را داریم:

$$\begin{cases} t(n, m) = t(n-1, m) + t(n, m-1) \\ t(n, \cdot) = t(\cdot, m) = 1 \end{cases}$$

حال سری مولد  $G_n(x)$  را به صورت زیر تعریف می‌کنیم:

$$G_n(x) = \sum_{m=\cdot}^{\infty} t(n, m)x^m \quad (1)$$

$$\begin{aligned} G_n(x) &= \sum_{m=\cdot}^{\infty} t(n, m)x^m = t(n, \cdot) + \sum_{m=1}^{\infty} t(n, m)x^m \\ &= 1 + \sum_{m=1}^{\infty} (t(n-1, m) + t(n, m-1))x^m \\ &= 1 + \sum_{m=1}^{\infty} t(n-1, m)x^m + x \sum_{m=1}^{\infty} t(n, m-1)x^{m-1} \\ &= G_{n-1}(x) + xG_n(x) \Rightarrow \end{aligned}$$

$$G_n(x) = \frac{G_{n-1}(x)}{1-x}$$

اگر از روش جایگذاری با تکرار برای حل این رابطه بازگشتی استفاده کنیم، خواهیم داشت:

$$G_n(x) = \frac{G_{n-r}(x)}{(1-x)^r} = \frac{G_{n-r}(x)}{(1-x)^r}$$

با ادامه این روند

$$G_n(x) = \frac{G_o(x)}{(1-x)^n}$$

اما

$$G_{\cdot}(x) = \sum_{m=\cdot}^{\infty} x^m = \frac{1}{(1-x)} \quad \text{یا} \quad G_{\cdot}(x) = \sum_{m=\cdot}^{\infty} t(\cdot, m)x^m$$

در نتیجه

$$G_n(x) = \frac{1}{(1-x)^{n+1}} \Rightarrow$$

$$G_n(x) = \sum_{m=\cdot}^{\infty} C_{n+m}^m x^m \quad (2)$$

از مقایسه (۱) و (۲) خواهیم داشت:

$$t(n, m) = C_{n+m}^m$$

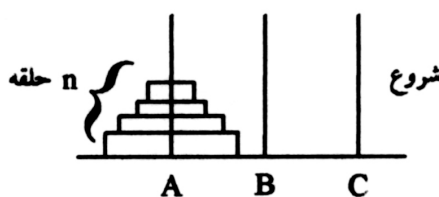
که برای حالت  $n = m = 8$  خواهیم داشت:

$$t(8, 8) = C_{16}^8$$

**مثال :** مسألهٔ برج هانوی را با سه میله  $A$ ,  $B$ ,  $C$  در نظر بگیرید. در این جا هیچ حلقه‌ای را نمی‌توان مستقیماً از  $A$  به  $B$  یا از  $B$  به  $A$  منتقل کرد. یعنی چنین انتقال‌هایی تنها به کمک میلهٔ  $C$  انجام می‌پذیرند. اگر در شروع کار،  $n$  حلقه در میلهٔ  $A$  داشته باشیم و  $t(n)$  حداقل تعداد جابه‌جایی‌ها برای انتقال  $n$  حلقه از  $A$  به  $B$  باشد، اول یک رابطهٔ بازگشتی برای  $t(n)$  ارائه دهید و سپس این رابطهٔ بازگشتی را حل کنید. در این جا نیز مثل برج هانوی هیچ حلقه‌ای را نمی‌توان روی حلقهٔ کوچک‌تر از خودش قرار داد و در هر مرحله هم فقط یک حلقه جابه‌جا می‌شود.

**حل :**

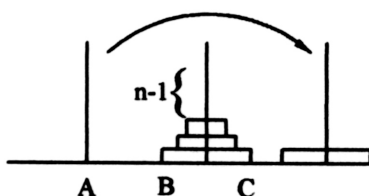
بدیهی است که  $t(1)=2$  و برای  $n \geq 2$  مراحل مختلف انجام کار (با استفاده از استقراء) در زیر به تصویر کشیده شده‌اند:



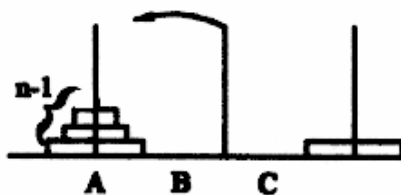
(الف) فرض (استقراء)  $n-1$  حلقهٔ بالایی را با  $t(n-1)$  جابه‌جایی از  $A$  به  $B$  برده باشیم.



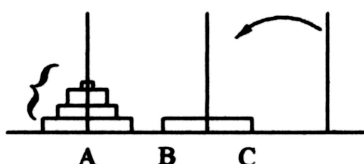
(ب) با یک جابه‌جایی حلقهٔ زیرین را به میله  $C$  می‌بریم.



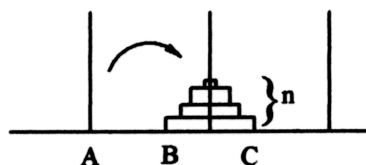
(ج) با  $t(n-1)$  جابه‌جایی  $n-1$  حلقهٔ را از  $B$  به  $A$  می‌بریم:



(د) با یک جابه‌جایی تک حلقهٔ واقع در ستون  $C$  را به ستون  $B$  می‌بریم:



ه) با  $t(n-1)$  جابه‌جایی دیگر،  $n-1$  حلقه واقع در ستون A را به ستون B می‌بریم:



بنابراین تعداد کل جابه‌جایی‌ها از رابطه بازگشتی زیر حاصل می‌شود:

$$t(n) = 3t(n-1) + 2$$

رابطه بازگشتی بالا را می‌توان با استفاده از روش جایگذاری با تکرار حل کرد:

$$\begin{aligned} t(n) &= 2 + 3t(n-1) = 2 + 3(2 + 3t(n-2)) \\ &= 2 + 2 \times 3 + 3^2 t(n-2) \\ &= 2 + 2 \times 3 + 3^2 (2 + 3t(n-3)) \\ &= 2 + 2 \times 3 + 2 \times 3^2 + 3^3 t(n-3) \\ t(n) &= 2 + 2 \times 3 + 2 \times 3^2 + 2 \times 3^3 + \dots + 2 \times 3^{n-2} + 3^{n-1} t(1) \\ t(n) &= 2 + 2 \times 3 + 2 \times 3^2 + 2 \times 3^3 + \dots + 2 \times 3^{n-1} \\ t(n) &= 2 \frac{3^n - 1}{3 - 1} \Rightarrow t(n) = 3^n - 1 \end{aligned}$$

یکی دیگر از ابزارهای بسیار مهم که برای تحلیل الگوریتم‌های تقسیم و غلبه (فصل بعد) به کار می‌روند، روابط تقسیم و غلبه هستند. صورت عمومی یک رابطه تقسیم و غلبه به صورت زیر است:

$$t(n) = at\left(\frac{n}{b}\right) + f(n) \quad (1)$$

که در آن  $a > 0$ ,  $b > 1$  و  $f(n)$  یک تابع بر حسب  $n$  است.

برای مثال عنصر ماکزیمم آرایه  $A[1..n]$  را می‌توان به صورت زیر به دست آورد (روش تقسیم و غلبه). برای سهولت فرض می‌کنیم که  $n = 2^m$ . اگر آرایه یک عضو داشته باشد، همین عنصر، عنصر ماکزیمم است (پایه استقراء). فرض کنید (فرض استقراء) که عناصر ماکزیمم را برای زیر آرایه‌های  $A\left[1..\frac{n}{2}\right]$ ،  $A\left[\frac{n}{2}+1..n\right]$  پیدا کرده باشیم. اگر  $lmax$  و  $rmax$  این عناصر باشند، بدیهی است که با یک مقایسه بین این دو می‌توان عنصر ماکزیمم کل آرایه را به دست آورد. بنابراین، اگر  $t(n)$  تعداد مقایسه‌های لازم برای پیدا کردن عنصر ماکزیمم با این روش باشد، بدیهی است که برای  $n = 2^m$

$$\begin{cases} t(n) = 2t\left(\frac{n}{2}\right) + 1 \\ t(1) = 0 \end{cases} \quad (2)$$

**مثال :** یکی از الگوریتم‌های مهم برای مرتب کردن عناصر آرایه  $A[1..n]$ ، الگوریتم مرتب‌سازی ادغامی است که یک روش تقسیم و غلبه دیگر است و در فصل بعد به آن خواهیم پرداخت. در اینجا فقط به ایده الگوریتم مزبور اشاره می‌کنیم که برای رسیدن به رابطه تقسیم و غلبه، لازم است. در اینجا این نکته مهم را یادآوری می‌کنیم که حل مسائل به روش تقسیم و غلبه ارتباط تنگاتنگی

با اثبات مسائل به طریقه استقرای ریاضی دارد. یک نمونه را در مثال قبل دیدیم. حال به یک نمونه دیگر اشاره می‌کنیم. اگر آرایه، یک عنصر داشته باشد (پایه استقرا)، مسلم است که نیازی به هیچ جابه‌جایی عناصر آرایه نیست. در این‌جا نیز برای سهولت فرض می‌کنیم که  $n = 2^m$ . حال فرض کنید (فرض استقرا) که ما نیم آرایه‌های  $A\left[\frac{n}{2}+1..n\right]$ ،  $A\left[1..\frac{n}{2}\right]$  را مرتب کرده باشیم. کافی است که برای مرتب کردن کل عناصر آرایه  $A[1..n]$ ، دو نیم آرایه مرتب شده مزبور را با هم ادغام کنیم. حال اگر فرض کنیم که  $t(n)$  برابر با تعداد جابه‌جایی‌های عناصر آرایه  $A[1..n]$  توسط الگوریتم مرتب‌سازی ادغامی برای مرتب کردن آن باشد، بدیهی است که رابطه تقسیم و غلبه زیر را خواهیم داشت:

$$\begin{cases} t(n) = 2t\left(\frac{n}{2}\right) + n \\ t(1) = 0 \end{cases} \quad (3)$$

که در آن  $n$  تعداد جابه‌جایی‌های لازم برای ادغام دو نیم آرایه مرتب شده است. همان‌گونه که مشاهده می‌شود روابط تقسیم و غلبه (۲) و (۳)، حالت‌های خاصی از رابطه عمومی تقسیم و غلبه ارائه شده در (۱) هستند. برای رسیدن به جواب می‌توان هر کدام از این‌ها را با استفاده از روش جایگذاری با تکرار حل کرد. اگر این عمل را برای رابطه تقسیم و غلبه عمومی ارائه شده در (۲) انجام دهیم، دیگر نیازی به حل حالت‌های خاص روابط تقسیم و غلبه نخواهد بود. قضیه زیر که به قضیه اصلی معروف است، جواب رابطه تقسیم و غلبه عمومی (۲) را به دست می‌دهد.

#### قضیه اصلی:

جواب رابطه تقسیم و غلبه  $t(n) = at\left(\frac{n}{b}\right) + cn^k$  که در آن  $n$  یک عدد طبیعی و  $b > 1$  به صورت زیر بیان می‌شود.

$$t(n) = \begin{cases} \theta\left(n^{\log_b a}\right) & a > b^k \\ \theta\left(n^k \log n\right) & a = b^k \\ \theta\left(n^k\right) & a < b^k \end{cases} \quad \begin{array}{l} \text{الف) اگر} \\ \text{ب) اگر} \\ \text{ج) اگر} \end{array}$$

**مثال:** جواب رابطه تقسیم و غلبه زیر را به دست آورید که مربوط به تعداد مقایسه‌های لازم در روش تقسیم و غلبه برای پیدا کردن عنصر ماکزیمم یک آرایه است.

$$t(n) = 2t\left(\frac{n}{2}\right) + 1$$

در این‌جا  $a=2, b=2, k=0$  بنابراین  $b^k=1, a>b^k$  لذا  $a=2$  و در نتیجه داریم:

$$t(n) = \theta(n)$$

اگر رابطه مزبور را با استفاده از روش جایگذاری با تکرار حل کنیم، با این فرض که  $t(1)=0$  در این صورت  $t(n) = n-1$  خواهد بود.

**مثال:** جواب رابطه تقسیم و غلبه زیر را، که مربوط به الگوریتم مرتب‌سازی ادغامی است، به دست آورید.

$$t(n) = 2t\left(\frac{n}{2}\right) + n$$

در این جا  $a=2, b=2, k=1$  لذا داریم  $a=b^k$  و بنابراین بنا به حالت (ب) قضیه اصلی داریم:

$$t(n) = \theta(n \log n)$$

در زیر به چند مثال دیگر در مورد استفاده از قضیه اصلی می‌پردازیم:

**مثال :** جواب رابطه تقسیم و غلبه زیر را به دست آورید.

$$t(n) = t\left(\frac{2n}{3}\right) + 1$$

در این جا  $a=1, b=\frac{2}{3}, k=0$  لذا  $a=b^k$  و در نتیجه:

$$t(n) = \theta(\log n)$$

**مثال :** جواب رابطه تقسیم و غلبه زیر را با استفاده از قضیه اصلی به دست آورید:

$$t(n) = 9t\left(\frac{n}{3}\right) + n^2$$

داریم  $a=9, b=\frac{1}{3}, k=2$  لذا  $a=b^k$  و در نتیجه:

$$t(n) = \theta(n^2 \log n)$$

**مثال :** جواب رابطه تقسیم و غلبه زیر را با استفاده از قضیه اصلی به دست آورید:

فرض  $n = 2^m$

$$t(n) = 2t(\sqrt{n}) + \log_2 n$$

**حل :**

$$s(m) = 2s\left(\frac{m}{2}\right) + m \quad \text{لذا} \quad s(m) = t(2^m) = t(n)$$

حال برای این رابطه تقسیم و غلبه داریم:

$$a=2, b=\frac{1}{2}, k=1$$

لذا

$$S(m) = \theta(m \log m) \Rightarrow$$

$$t(n) = \theta(\log_2 n \log \log_2 n)$$