

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 چرخهٔ حیات یک الگوریتم نامید.

(الف) طراحی الگوریتم‌ها: روش‌های مختلفی برای طراحی الگوریتم‌ها وجود دارند که اتفاقاً جزء سیلاس 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 مذبور درست است. الگوریتمی درست است که به ازای هر ورودی مناسب خروجی صحیحی بدهد. اثبات درستی الگوریتم‌ها به اثبات قضایا در ریاضی می‌ماند و مرحلهٔ بسیار مهمی در زمینهٔ مطالعهٔ الگوریتم‌ها است، اما در این درس با این قسمت زیاد سروکار نخواهیم داشت.

(ج) تحلیل الگوریتم‌ها (تحلیل مقدم، ارزیابی کارآیی الگوریتم‌ها): یک الگوریتم در زمان اجرا 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 نوشتمن آن به زبان برنامه‌نویسی خاص است که معمولاً بعد از تحلیل مقدم آن صورت می‌گیرد و نام برنامه به آن اطلاق می‌شود.

(ه) تست برنامه: تست یک برنامه شامل i : اشکال‌زدایی و ii : تحلیل موخر (اندازه‌گیری کارآیی) است. اندازه‌گیری کارآیی 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، نوشتمن، 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... را نادیده گرفته و زمان اجرای یک الگوریتم را به صورت زیر در نظر خواهیم گرفت:

$$t_{P(n)} = \sum_{\text{دستورالعمل } i} t_i f_i$$

که در آن، n تعداد ورودی‌ها (یا خروجی‌ها و یا هر دو)، 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 cpu است، در این صورت داریم:

$$t_{P(n)} = \sum_{\text{دستورالعمل } i} f_i$$

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 $t_{P(n)} = a n + b$ 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 $O(\theta)$

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_0 طبیعی، به گونه‌ای که

$$\forall n \geq n_0 \Rightarrow f(n) \leq c_1 g(n)$$

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

اگر $c_1 = 3$ باشد داریم:

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

و بنابراین کافی است $n_0 = 8$

تعريف Ω (بررسی بهترین حالت زمان اجرای یک الگوریتم)

$$\exists c_2 > 0, n_0 \Leftrightarrow f(n) = \Omega(g(n))$$

$$\forall n \geq n_0 \Rightarrow f(n) \geq c_2 g(n)$$

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

کافی است $c_2 = 1$ باشد، زیرا رابطه $2n^2 + 7n + 5 \geq n^2$ همواره برای هر $n \geq n_0 = 1$ برقرار است.

تعريف θ (بررسی حالت میانگین زمان اجرای یک الگوریتم)

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

$$\exists c_1, c_2 > 0 \quad \Rightarrow \quad \forall n \geq n_0 \quad c_2 g(n) \leq f(n) \leq c_1 g(n)$$

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

حل: از مثال‌های ۵, ۴ نتیجه می‌شود.

$$\boxed{f(n) = \Omega(g(n)) \text{ و } f(n) = O(g(n)) \Leftrightarrow f(n) = \theta(g(n))}$$

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

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

بنابراین هر چند می‌توان نوشت $2n^2 + 7n + 5 = O(n^4)$ یا $2n^2 + 7n + 5 = O(n^3)$ یا $2n^2 + 7n + 5 = O(n^2)$ ، لذا در نهایت همان $O(n^2)$ را خواهیم پذیرفت.

به عبارت دیگر، کوچک‌ترین کرانه بالایی زمان اجرا مورد نظر خواهد بود. عین همین مطلب در مورد Ω هم صادق خواهد بود، یعنی

هر چند طبق تعریف می‌توانیم داشته باشیم $2n^2 + 7n + 5 = \Omega(n^2)$ ، 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 $A(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n + a_0$ تشکیل شده

$$\boxed{A(n) = \theta(n^m)}$$

лем: اگر برنامه‌ای داشته باشیم که از k قطعه برنامه مستقل از هم با مرتبه‌های بزرگی $O(n^{m_k}), O(n^{m_{k-1}}), \dots, O(n^{m_1})$ تشکیل شده

باشد، در این صورت مرتبه بزرگی خود برنامه برابر $O(n^{\max\{m_1, m_2, \dots, m_k\}})$ خواهد بود.

تحليل 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.       temp=A[j-1]
5.       A[j-1]=A[j]
6.       A[j]=temp
    }
  }
}

```

بررسی بدترین حالت: بدترین حالت زمانی اتفاق می‌افتد که دستور 3 همواره درست باشد. این مسئله عملأً زمانی اتفاق می‌افتد که خود آرایه $A[1..n]$ به صورت نزولی مرتب شده باشد، در این صورت داریم:

شماره دستور العمل	تعداد دفعات اجرا
1	$n-1$
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	$(n-1)+(n-2)+\dots+2+1$

$$t(n) = n-1 + 5 \frac{n(n-1)}{2} \leftarrow \text{جمع}$$

$$t(n) = O(n^2)$$

بررسی بهترین حالت: بهترین حالت زمانی اتفاق می‌افتد که دستور 3 هیچ وقت اجرا نشود و این زمانی اتفاق می‌افتد که خود آرایه به صورت صعودی مرتب شده باشد، در این حالت داریم:

شماره دستور العمل	تعداد دفعات اجرا
1	$n-1$
2	$(n-1)+(n-2)+\dots+2+1$
3	$(n-1)+(n-2)+\dots+2+1$
4	0
5	0
6	0

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

$$\therefore t(n) = \Omega(n^2)$$

لذا از لم 1 نتیجه می‌شود که

تحلیل الگوریتم مرتب‌سازی 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  $[item < A[i]]^*$  do
    {
         $A[i+1] = A[i]$ ,  $i=i-1$ 
    }
     $A[i+1] = item$ 
}

```

برای سهولت در تحلیل این الگوریتم، فقط تعداد دفعات اجرای دستور العمل 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.

مقدار پارامتر j	تعداد دفعات اجرای دستور العمل
2	2
3	3
:	:
n	n

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

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

مقدار پارامتر j	تعداد دفعات اجرای دستور العمل
2	1
3	1
:	1
n	n

$$t(n) = 1 + 1 + \dots + 1 = n - 1 = \Omega(n)$$

مرتبه‌های بزرگی متدائل 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 $O(n^2)$ ، $O(n \log n)$ ، $O(n)$ یا $O(2^n)$ است که

اثبات می‌شود برابر $\Theta(n^2)$ است.

یکی از مهم‌ترین روش‌هایی که برای تحلیل الگوریتم‌ها به کار می‌رود، روابط بازگشتی است. در دلیل، 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 $[2..n]$ 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 گوییم.

مثال ۲: مسئله 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 مسئله معروف برج هانوی و هم‌چنین مرتبه بزرگی الگوریتم هانوی نیز هست.

Algorithm Hanoi (n, A, B, C)

تعداد حلقه‌ها و C, B, A از نوع کاراکتر هستند.

if $n \geq 1$ then

{

Hanoi ($n-1$, A, C, B)

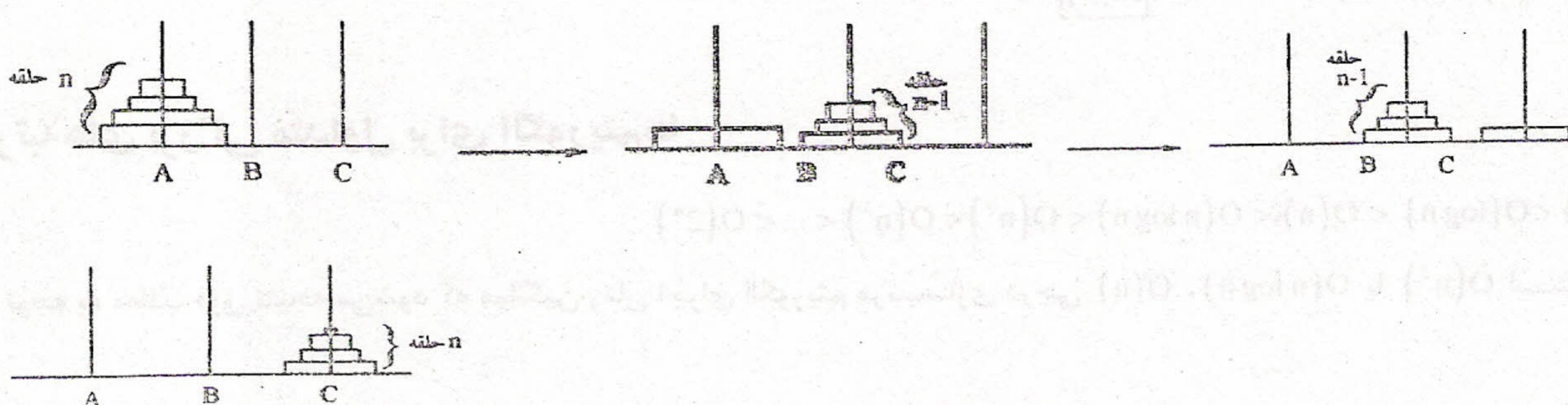
* Write Ln (A, " → ", C)

Hanoi ($n-1$, B, A, C)

}

ایده‌ای که برای نوشنی الگوریتم مذبور 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 برد 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 (آنچه جایی دیگر). بنابراین مجموع جابه‌جایی‌های انجام شده عبارت است از:

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



مثال ۳: دنباله اعداد فیبوناچی
می‌دانیم که این دنباله به صورت زیر است:

n	0	1	2	3	4	5	6	7	8	...
F_n	0	1	1	2	3	5	8	13	21	...

رابطه بازگشتی برای این دنباله به صورت زیر است:

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

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:

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

یا

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

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

$$a_n = (n-1) + (n-2) + (n-3) + \dots + 2 + 1 + 0 + a_0$$

و لذا

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

البته قبل "هم دیده بودیم که زمان اجرای الگوریتم مرتب‌سازی حبابی $\Theta(n^2)$ است.

مثال : تعداد جایه‌جایی‌ها در مساله برج هانوی برای جایه‌جایی n حلقه واقع در یک ستون به ستون دیگر دیدیم که:

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

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

$$a_n = 1 + 2 + 2^2 + 2^3 + \dots + 2^{n-1} + 2^n a_0$$

و یا

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

بنابراین مرتبه بزرگی الگوریتم هانوی برابر $\Theta(2^n)$ 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 سُری‌های 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 مذبور 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

$$A(x) = xA(x) + \frac{x^2}{(1-x)^2} \Rightarrow A(x) = \frac{x^2}{(1-x)^3}$$

اما از روی فرمول با قرار دادن $m=3$ داریم:

$$\frac{1}{(1-x)^3} = \sum_{n=0}^{\infty} C_{n+2}^n x^n$$

و 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 (1) و (2) نتیجه می‌شود که

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

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

مثال: رابطه بازگشتی $\begin{cases} a_n = 2a_{n-1} + 1 & n \geq 1 \\ a_0 = 0 & \end{cases}$ را با استفاده از سری مولد حل کنید. (رابطه بازگشتی مربوط به مسالة برج 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 مذبور را به صورت کسرهای ساده‌تر تجزیه می‌کنیم:

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

با مساوی قرار دادن صورت کسرهای اول و آخر داریم:

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

رابطه مذبور 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 $a = 1$, $b = -1$ و 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 می‌شود که

$$a_n = 2^n - 1$$

و باز هم همان نتیجه‌های است که از روش جایگذاری با تکرار به آن رسیده بودیم.

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

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

داریم:

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

$$\begin{aligned} F(x) &= x + \sum_{n=2}^{\infty} (F_{n-1} + F_{n-2}) x^n \\ &= x + x \left(\sum_{n=2}^{\infty} F_{n-1} x^{n-1} \right) + x^2 \left(\sum_{n=2}^{\infty} F_{n-2} x^{n-2} \right) \\ &= x + x [F(x) - F_0] + x^2 F(x) \\ &= x + x F(x) + x^2 F(x) \Rightarrow \end{aligned}$$

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

حال اگر تعریف کنیم $\bar{\varphi} = \frac{1-\sqrt{5}}{2}$, $\varphi = \frac{1+\sqrt{5}}{2}$, کسر مذبور را می‌توان 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 صورت‌های کسرهای اول و آخر خواهیم داشت:

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

با قرار دادن $x = 0$, $x = 1$ خواهیم داشت:

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

و بنابراین

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

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

و در نتیجه

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

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

اما $\left(\frac{1-\sqrt{5}}{2} \right)^n$ در $\frac{1-\sqrt{5}}{2} \approx -0.61$ ، $\frac{1+\sqrt{5}}{2} \approx 1.61$ ، بنابراین با توجه به این که F_n ها اعداد طبیعی هستند، لذا می‌توان از جمله

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 ≤ 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:

$$\begin{cases} b_n = 1 + b_{n-1} + b_{n-2} & n \geq 2 \\ b_0 = 1, b_1 = 1 & \end{cases}$$

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

$$\begin{cases} a_n = a_{n-1} + a_{n-2} & n \geq 2 \\ a_0 = 2, a_1 = 2 & \end{cases}$$

که مشابه رابطه بازگشتی برای دنباله فینوناچی است (یک رابطه بازگشتی همگن مرتبه 2). از حل آن نتیجه می‌شود که:

$$b_n = a_n - 1 \Rightarrow$$

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

اما دیدیم که $F_n \approx (1.61)^n$ و 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 ام دنباله فیبوناچی ارایه 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 می‌شود:

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

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

معادله مشخصه برای رابطه بازگشتی بالا به صورت زیر تعریف می‌شود:

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

در اینجا فقط رابطه بازگشتی همگن مرتبه 2 را بررسی می‌کنیم. در رابطه (1)، اگر $k=2$ باشد خواهیم داشت:

$$c_n r^2 + c_{n-1} r + c_{n-2} = 0$$

که یک چند جمله‌ای از درجه 2 است.

فرض کنید r_1, r_2 ریشه‌های این چند جمله‌ای 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 $r_1 = r_2$ (ریشه مضاعف)، در این صورت:

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

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:

$$\left\{ \begin{array}{l} F_n = F_{n-1} + F_{n-2} \quad n \geq 2 \\ F_0 = 0 \quad F_1 = 1 \end{array} \right.$$

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

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

و 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:

$$\left\{ \begin{array}{l} 0 = c_1 + c_2 \\ 1 = \frac{\sqrt{5}}{2} (c_1 - c_2) \end{array} \right.$$

و لذا

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

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 $\begin{cases} a_n = 4a_{n-1} - 4a_{n-2} & n \geq 2 \\ a_0 = 0 \quad a_1 = 1 & \end{cases}$

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:

$$\cancel{\times 1 - \frac{c_2}{2}}, \quad 0 = c_1$$

$$1 = 2c_2 \Rightarrow c_2 = \frac{1}{2}$$

$$a_n = n2^{n-1} \quad \text{و لذا} \quad c_2 = \frac{1}{2}$$

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

حل:

$r^2 - 2r + 2 = 0$ و ریشه‌های آن عبارت‌اند از $r_1, r_2 = 1 \pm i$ و 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 این‌که شرایط مرزی و نیز ضرایب رابطه بازگشتی اعداد صحیحی هستند، مسلم است که همه a_n اعداد صحیحی هستند. حال، حداقل برای این‌که ظاهر a_n به صورت مختلط دیده نشود، از اعداد مختلط کمک می‌گیریم. می‌دانیم که عدد مختلط $z = x+iy$ را می‌توان به صورت $(r(\cos\theta + i\sin\theta))$ بیان کرد که

در آن $\theta = \arctan g \left(\frac{y}{x} \right)$ و $r = \sqrt{x^2 + y^2}$ 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:

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

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

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

و بنابراین

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

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

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

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

$$a_0 = 0 \Rightarrow k_1 = 0$$

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

ولذا

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

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 مسایل مشکل‌تر هم، به کار روند. در زیر به چند مثال در این مورد می‌پردازیم:

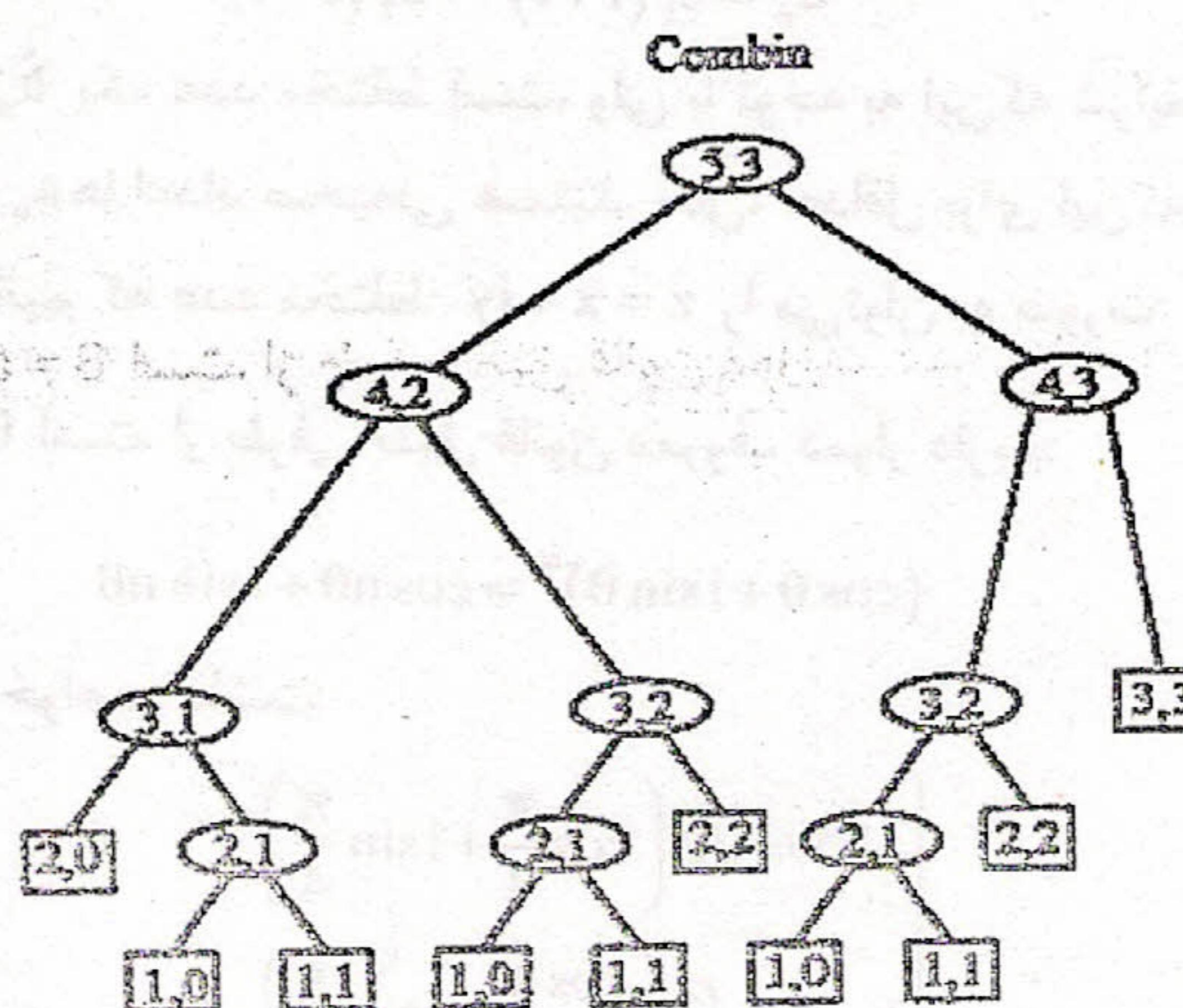
مثال: برای محاسبه C_n^m از روی رابطه بازگشتی زیر ارایه می‌شود. تعداد

فراخوانی‌های Combin(n, m) و در نتیجه مرتبه بزرگی آن را به دست آورید:

Algorithm Combin(n, m)

```
if (n=m) or (m=0) then Return (1)
else Return (Combin (n-1, m-1) + Combin (n-1, m))
```

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



که در این حالت تعداد فراخوانی‌ها برابر 19 است.

در حالت عمومی اگر فرض کنیم که $t(n, m)$ تعداد فراخوانی‌های 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:

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

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

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

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

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

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

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

اگر از روش جایگذاری با تکرار استفاده کنیم:

$$G_n(x) = (1+x)^2 G_{n-2}(x)$$

$$= (1+x)^3 G_{n-3}(x)$$

⋮

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

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

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

با مقایسه روابط (1) و (2) نتیجه می‌شود که :

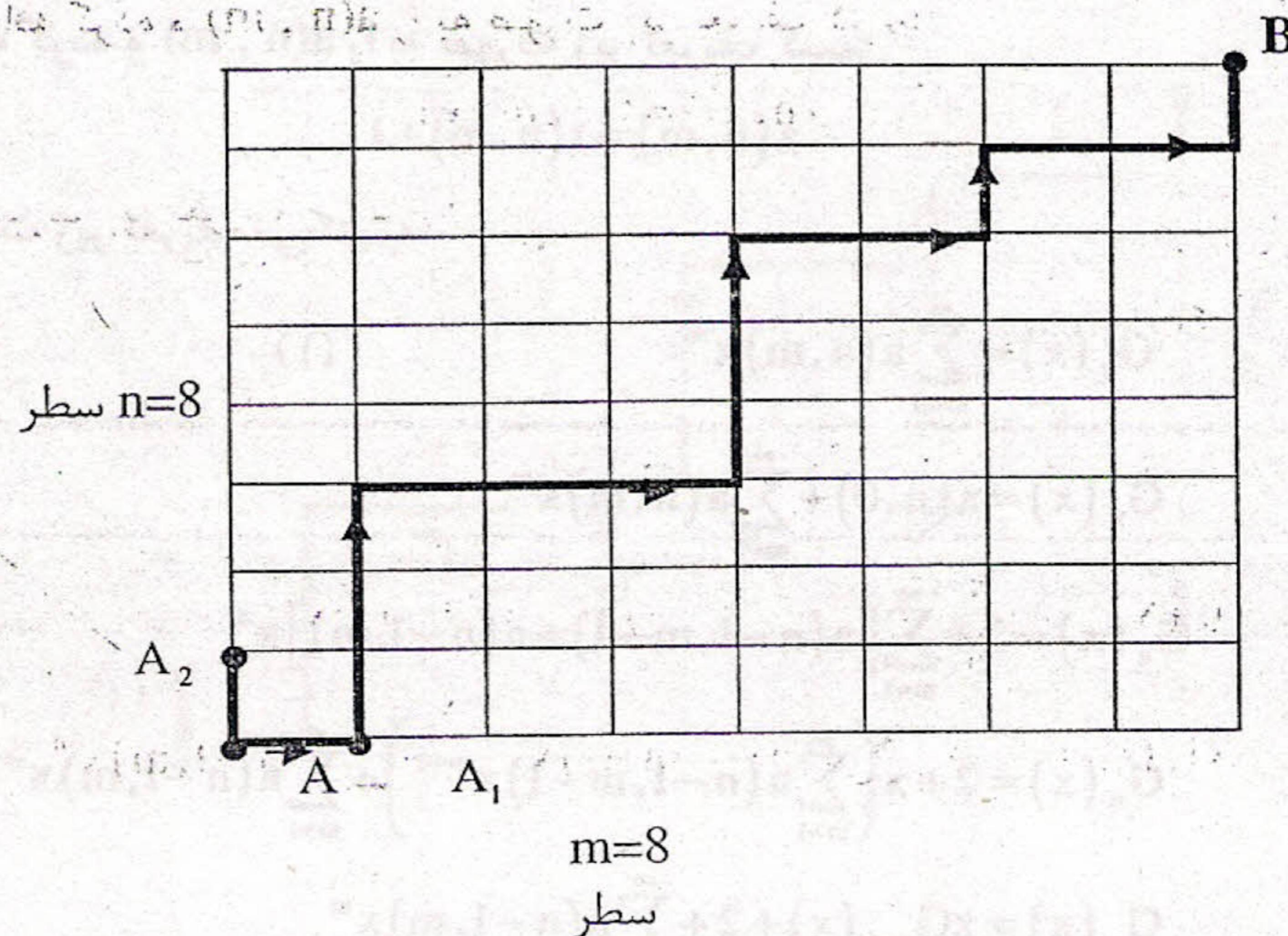
$$a(n, m) = 2C_n^m$$

در نتیجه

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

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

مثال : صفحه شترنجی 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 مذکور به نمایش در آمده است:



فرض کنیم $t(n,m)$ تعداد مسیرهایی باشد که در حالت عمومی در یک صفحه شترنجی $n \times m$ 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_2 رفته باشیم، در این صورت تعداد مسیرهای ممکن از A_2 به B برابر $t(n-1, m)$ خواهد بود، لذا رابطه بازگشتی زیر را داریم:

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

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

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

$$\begin{aligned} G_n(x) &= \sum_{m=0}^{\infty} t(n, m) x^m = t(n, 0) + \sum_{m=1}^{\infty} t(n, m) x^m \\ &= 1 + \sum_{m=1}^{\infty} (t(n-1, m) + t(n, m-1)) x^m \\ &= 1 + \sum_{m=1}^{\infty} t(n-1, m) x^m + x \sum_{m=1}^{\infty} t(n, m-1) x^{m-1} \\ &= G_{n-1}(x) + x G_n(x) \quad \Rightarrow \end{aligned}$$

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

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

$$G_n(x) = \frac{G_{n-2}(x)}{(1-x)^2} = \frac{G_{n-3}(x)}{(1-x)^3}$$

با ادامه این روند

$$G_n(x) = \frac{G_0(x)}{(1-x)^n}$$

$$G_0(x) = \sum_{m=0}^{\infty} x^m = \frac{1}{(1-x)}$$

$$G_0(x) = \sum_{m=0}^{\infty} t(0,m)x^m$$

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

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

(۲)

از مقایسه (۱) و (۲) خواهیم داشت:

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

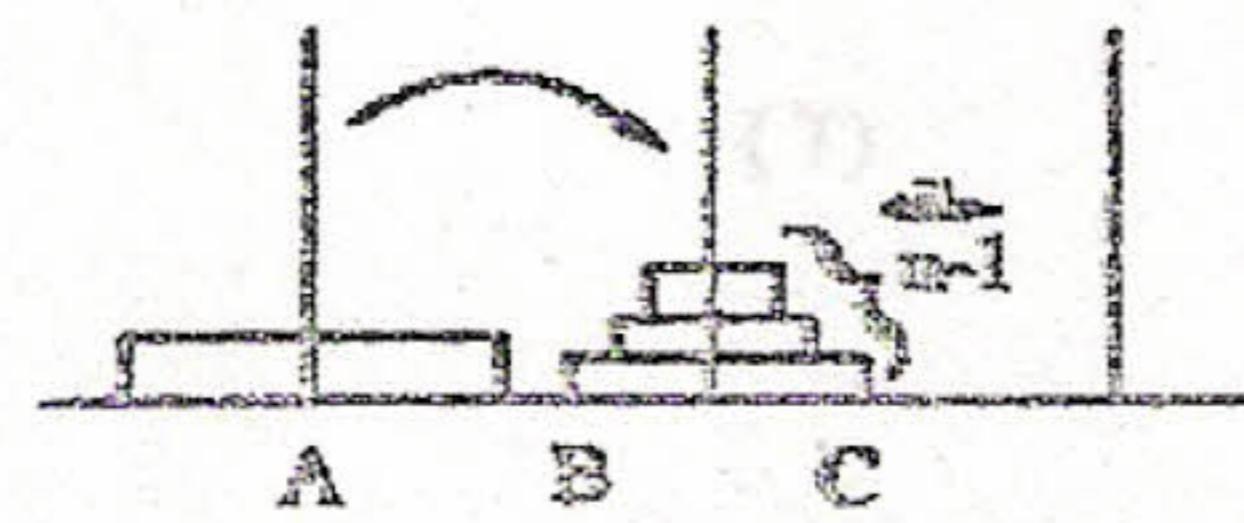
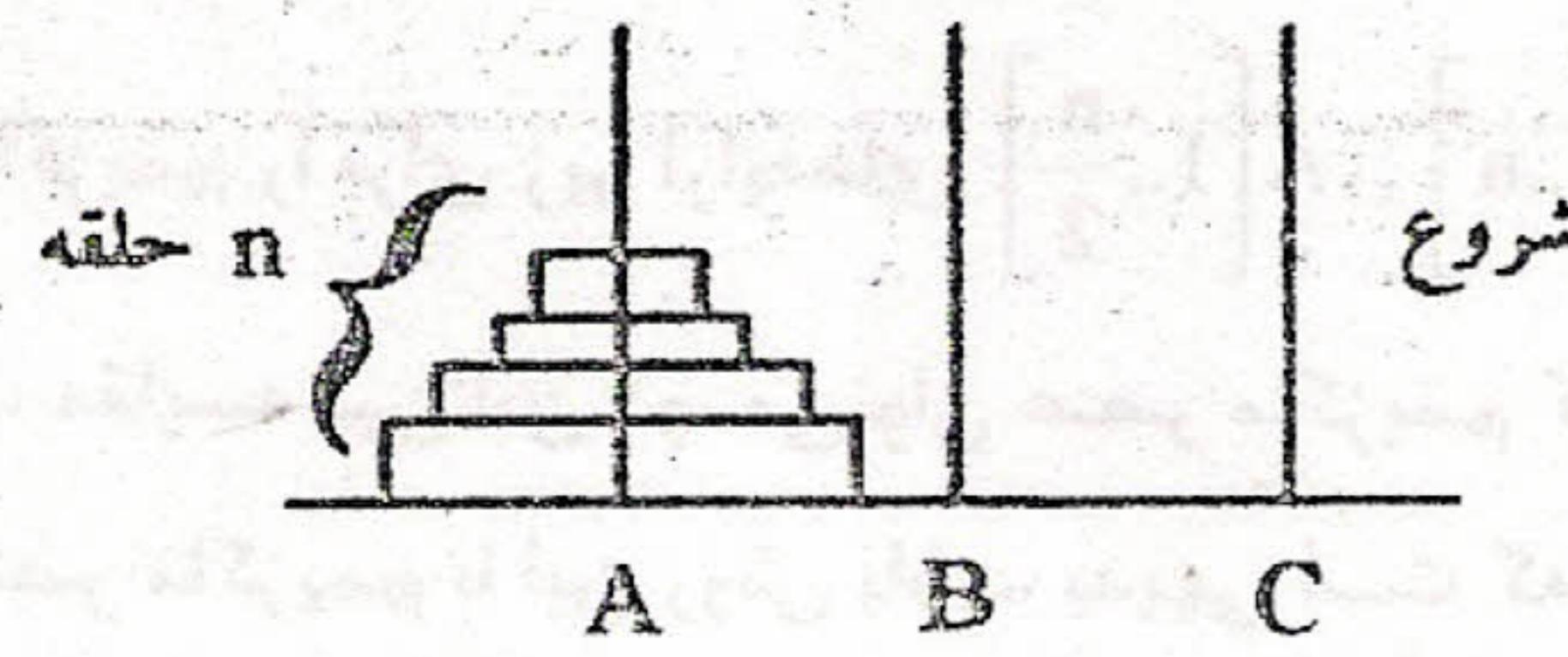
که برای حالت $n = m = 8$ خواهیم داشت:

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

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

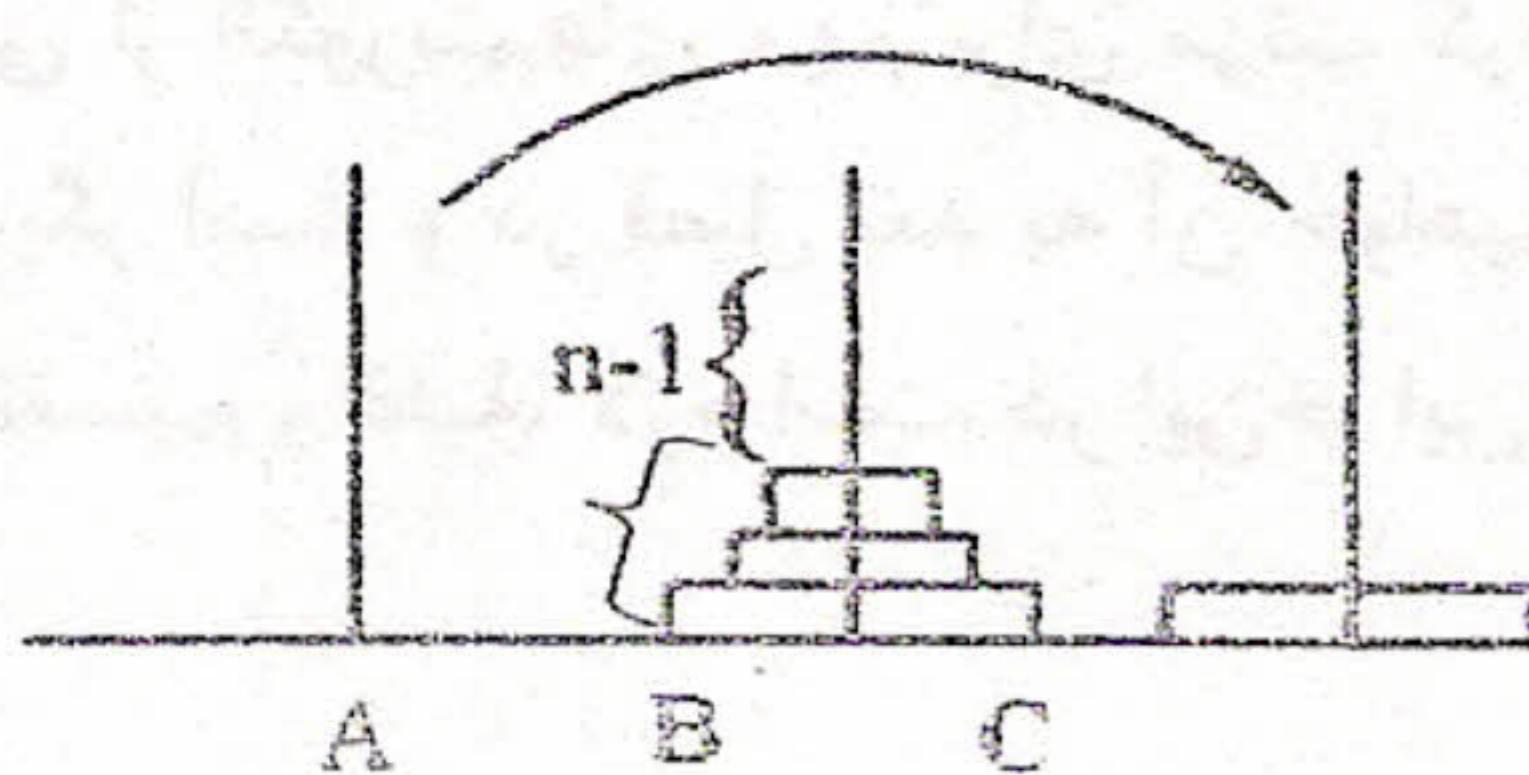
حل:

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:

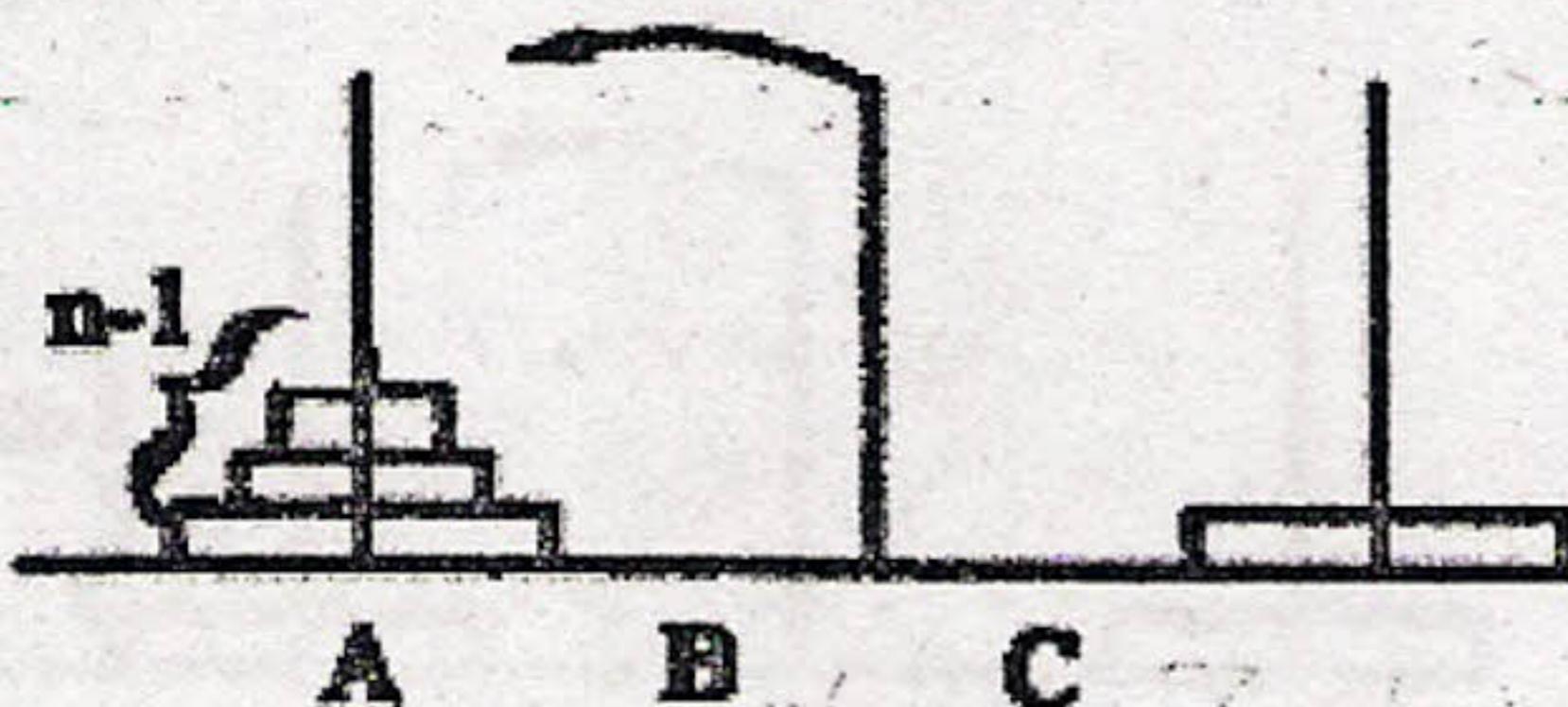


الف) فرض (استقراء) 1- n-1 حلقه بالای را با $t(n-1)$ جابه‌جایی از A به B برده باشیم.

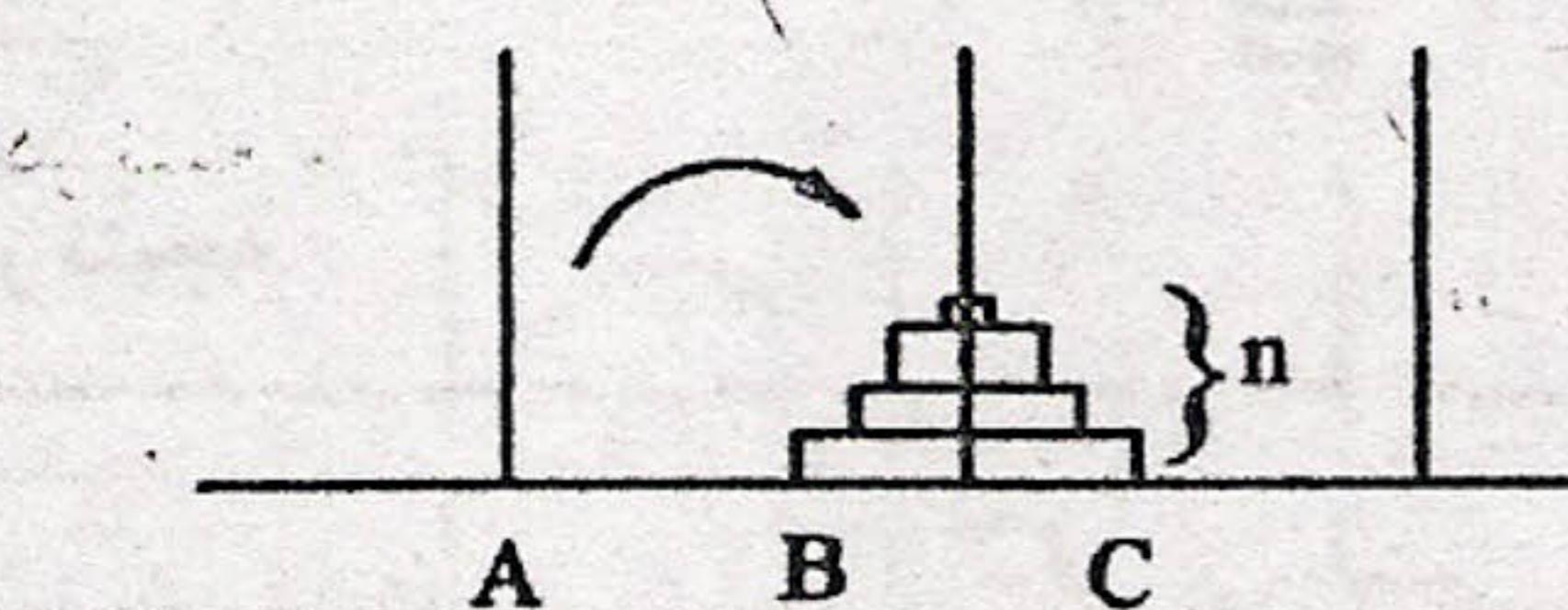
ب) با یک جابه‌جایی حلقة زیرین را به میله C می‌بریم.



ج) با $t(n-1)$ جایه‌جایی $n-1$ حلقه را از A به B می‌بریم.



د) با یک جایه‌جایی تک حلقه واقع در ستون C را به ستون B می‌بریم.



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

بنابراین تعداد کل جایه‌جایی‌ها 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:

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

$$= 2 + 2 \times 3 + 3^2 t(n-2)$$

$$= 2 + 2 \times 3 + 3^2 (2 + 3t(n-3))$$

$$= 2 + 2 \times 3 + 2 \times 3^2 + 3^3 t(n-3)$$

$$t(n) = 2 + 2 \times 3 + 2 \times 3^2 + 2 \times 3^3 + \dots + 2 \times 3^{n-2} + 3^{n-1} t(1)$$

$$t(n) = 2 + 2 \times 3 + 2 \times 3^2 + 2 \times 3^3 + \dots + 2 \times 3^{n-1}$$

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

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 $a > 1$, $b > 0$, $f(n)$ 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 مسلماً عنصر ماکزیمم نیز همین عنصر است (پایه استقراء). فرض کنید (فرض استقراء) که

عناصر ماکزیمم را برای زیر آرایه‌های $A\left[\frac{n}{2}+1..n\right]$, $A\left[1..\frac{n}{2}\right]$, I_{max} و r_{max} 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 مذبور اشاره می‌کنیم که برای رسیدن به رابطه تقسیم و غلبه، لازم است. در اینجا این نکته مهم را یادآوری می‌کنیم که حل مسایل به روش تقسیم و غلبه ارتباط تنگاتنگی

با اثبات مسایل به طریق استقرای ریاضی دارد. یک نمونه را در مثال قبل دیدیم. حال به یک نمونه دیگر اشاره می‌کنیم. اگر آرایه، یک عنصر داشته باشد (پایه استقرای)، 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 استقرای) که مانند 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 $[1..n]$, دو نیم آرایه مرتب شده مذبور را با هم ادغام کنیم. حال اگر فرض کنیم که $t(n)$ برابر با تعداد جابه‌جایی‌های عناصر آرایه $[1..n]$ A 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 (2), (3)، حالت‌های خاصی از رابطه عمومی تقسیم و غلبه ارایه شده در (1) 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 ارایه شده در (1) 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 (1) 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 $a > b$ به صورت زیر بیان می‌شود:

$$t(n) = \begin{cases} \theta(n^{\log_b a}) & \text{اگر } a > b^k \\ \theta(n^k \log n) & \text{اگر } a = b^k \\ \theta(n^k) & \text{اگر } a < b^k \end{cases}$$

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

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

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

$$t(n) = \theta(n^{\log_2 2})$$

اگر رابطه مذبور 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 اینجا $a = 2$, $b = 2$, $k = 1$, $n = 2^k$ و بنابراین با 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 اینجا $k=0$, $b=\frac{3}{2}$, $a=1$ و 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 $k=2$, $b=3$, $a=9$ و 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:

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

حال برای این رابطه تقسیم و غلبه داریم:

$$k=1, b=2, a=2$$

و لذا

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

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