

ساختمان داده‌ها

دکتر امین گلزاری اسکوئی

a.golzari@azaruniv.ac.ir

a.golzari@tabrizu.ac.ir

<https://github.com/Amin-Golzari-Oskouei>

دانشگاه شهید مدنی آذربایجان

پاییز ۱۴۰۲



فصل ۲

توابع بازگشتی

مطالب این فصل

تعریف توابع بازگشتی

نوشتن رابطه بازگشتی و محاسبه زمان اجرا

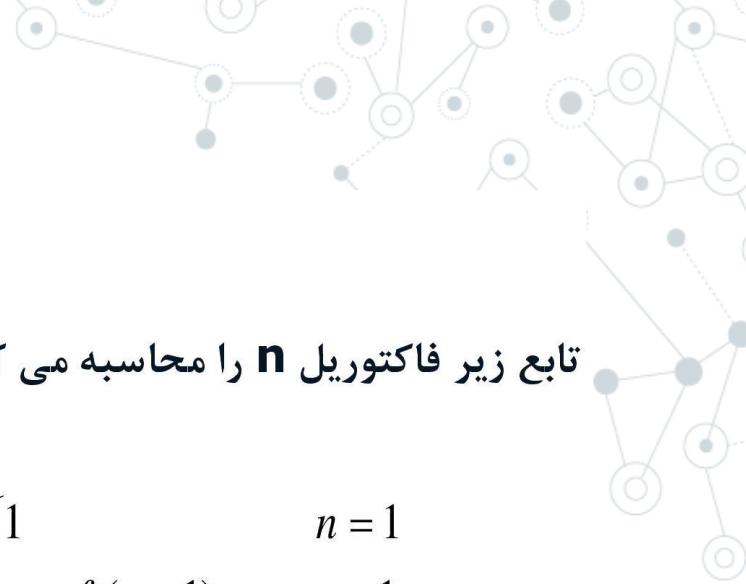
تعريف

تابع بازگشتی (recursive) :

تابعی است که حاوی حداقل یک دستور باشد که خود تابع را صدا بزند. این تابع به تعداد مراحل محدودی اجرا می‌شود و پس از آن متوقف می‌شود.

مثال‌هایی از توابع بازگشتی :

- ۱- فاکتوریل
- ۲- مجموع اعداد ۱ تا n
- ۳- توان
- ۴- ترکیب
- ۵- خارج قسمت تقسیم صحیح
- ۶- آکرمان
- ۷- هانوی
- ۸- فیبوناچی
- ۹- زاد و ولد خرگوش‌ها



تابع فاکتوریل

تابع زیر فاکتوریل n را محاسبه می کند:

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

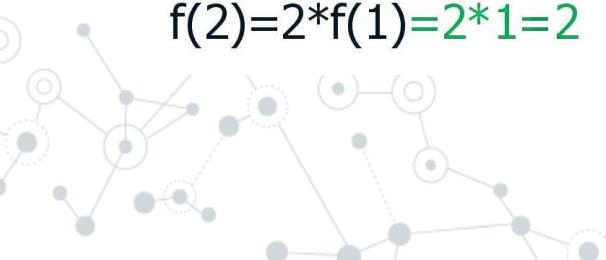
$$f(n) = \begin{cases} 1 & n = 1 \\ n \times f(n-1) & n > 1 \end{cases}$$

مثال: فراخوانی تابع با مقدار ۴ :

$$f(4) = 4 * f(3) = 4 * 6 = 24$$

$$f(3) = 3 * f(2) = 3 * 2 = 6$$

$$f(2) = 2 * f(1) = 2 * 1 = 2$$



مجموع اعداد ۱ تا n

تابع زیر مجموع اعداد ۱ تا n را محاسبه می کند:

```
sum( n )  
{  
    if (n==1)  
        return 1;  
    else  
        return n+ sum(n-1) ;  
}
```

$$sum(n) = \begin{cases} 1 & n = 1 \\ n + sum(n - 1) & n > 1 \end{cases}$$

خروجی فراخوانی تابع به ازای n=3 برابر 6 است:

$$\begin{aligned} sum(3) &= 3 + sum(2) = 3 + 3 = 6 \\ sum(2) &= 2 + sum(1) = 2 + 1 = 3 \end{aligned}$$

تابع توان

تابع زیر مقدار n^m را محاسبه می کند : (ورودی ها صحیح و مثبت هستند)

```
f(n , m){  
    if (m==1)  
        return n;  
    else  
        return n * f(n,m-1);  
}
```

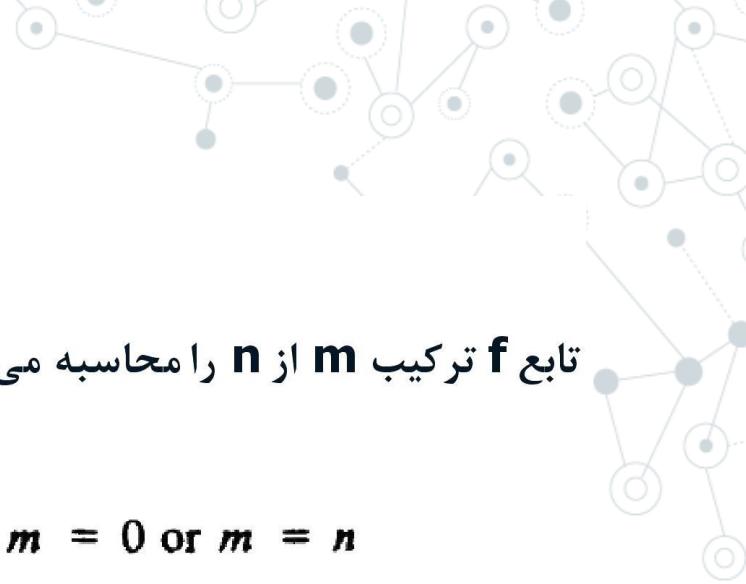
$$f(n, m) = \begin{cases} n & m = 1 \\ n \times f(n, m - 1) & m > 1 \end{cases}$$

به طور نمونه تابع را به صورت $f(3,4)$ فراخوانی می کنیم:

$$f(3,4) = 3 * f(3,3) = 3*3*3*3$$

$$f(3,3) = 3 * f(3,2) = 3*3*3$$

$$f(3,2) = 3 * f(3,1) = 3*3$$



تابع ترکیب

تابع f ترکیب m از n را محاسبه می کند:

```
f ( n , m )  
{  
    if ( (n==m) || (m==0) )  
        return 1;  
    else  
        return f(n-1 , m) + f(n-1 , m-1);  
}
```

$$\binom{n}{m} = 1 \quad \text{if } m = 0 \text{ or } m = n$$

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1} \quad \text{if } 0 < m < n$$

: فراخوانی $f(4,2)$

$$f(4,2) = f(3,2) + f(3,1) = 3+3=6$$

$$f(3,2) = f(2,2) + f(2,1) = 1+2=3$$

$$f(3,1) = f(2,1) + f(2,0) = 2+1=3$$

$$f(2,1) = f(1,1) + f(1,0) = 1+1=2$$

تابع **div**

(محاسبه خارج قسمت تقسیم صحیح)

محاسبه خارج قسمت تقسیم صحیح **a** بر **b** :

$$f(a,b) = \begin{cases} 0 & a < b \\ f(a-b,b) + 1 & a \geq b \end{cases}$$

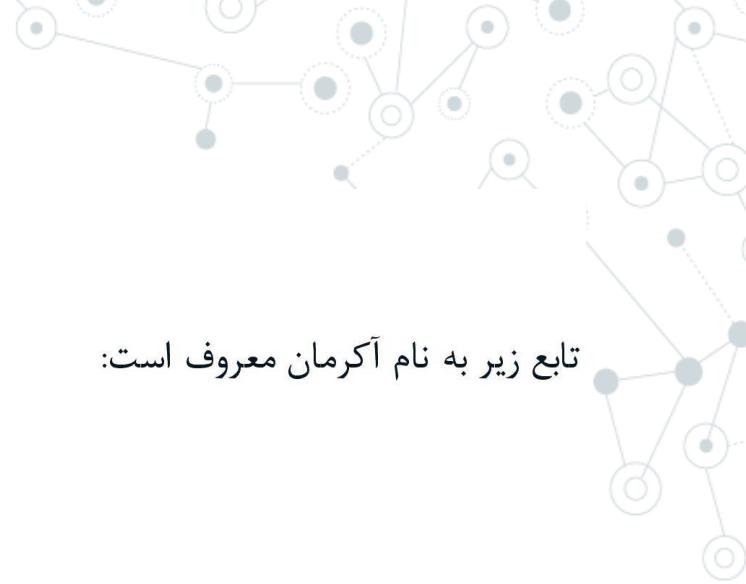
فراخوانی (**f(11,3)**) :

$$f(11,3) = f(8,3) + 1 = 2+1=3$$

$$f(8,3) = f(5,3) + 1 = 1+1=2$$

$$f(5,3) = f(2,3) + 1 = 0+1=1$$

تمرین: با تغییری در این تابع، **mod** را پیاده سازی کنید.



تابع آکرمان

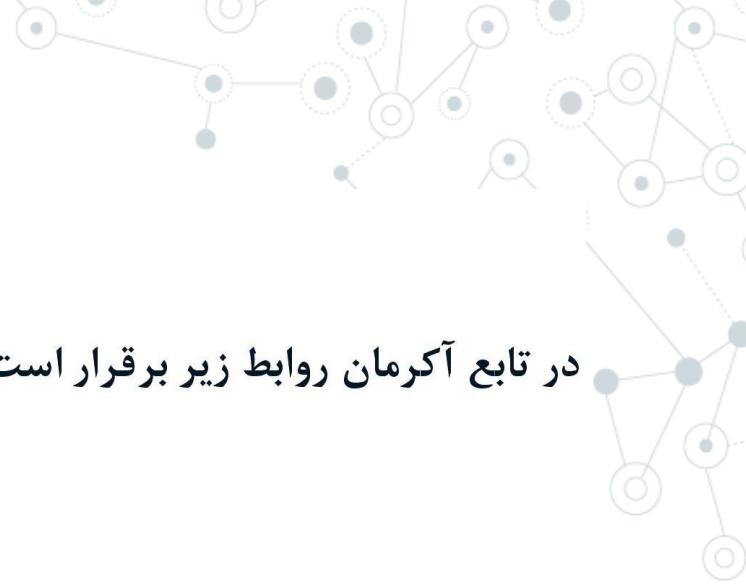
تابع زیر به نام آکرمان معروف است:

$$f(a, b) = \begin{cases} b + 1 & a = 0 \\ f(a - 1, 1) & b = 0 \\ f(a - 1, f(a, b - 1)) & a > 0, b > 0 \end{cases}$$

: $f(1,1)$

$$\begin{aligned} & f(1, 1) \\ &= f(0, f(1, 0)) \\ &= f(0, f(0, 1)) \\ &= f(0, 2) \\ &= 3 \end{aligned}$$





روابطی در تابع آکرمان

$$f(1, n) = 2 + (n + 3) - 3$$

$$f(2, n) = 2(n + 3) - 3$$

$$f(3, n) = 2^{n+3} - 3$$

$$f(4, n) = \underbrace{2^{2^{\dots^2}}}_{n+3} - 3$$

در تابع آکرمان روابط زیر برقرار است:

به طور نمونه :

$$f(1, 7) = 2 + (7 + 3) - 3 = 9$$

$$f(2, 7) = 2(7 + 3) - 3 = 17$$

$$f(3, 7) = 2^{(7+3)} - 3 = 1021$$

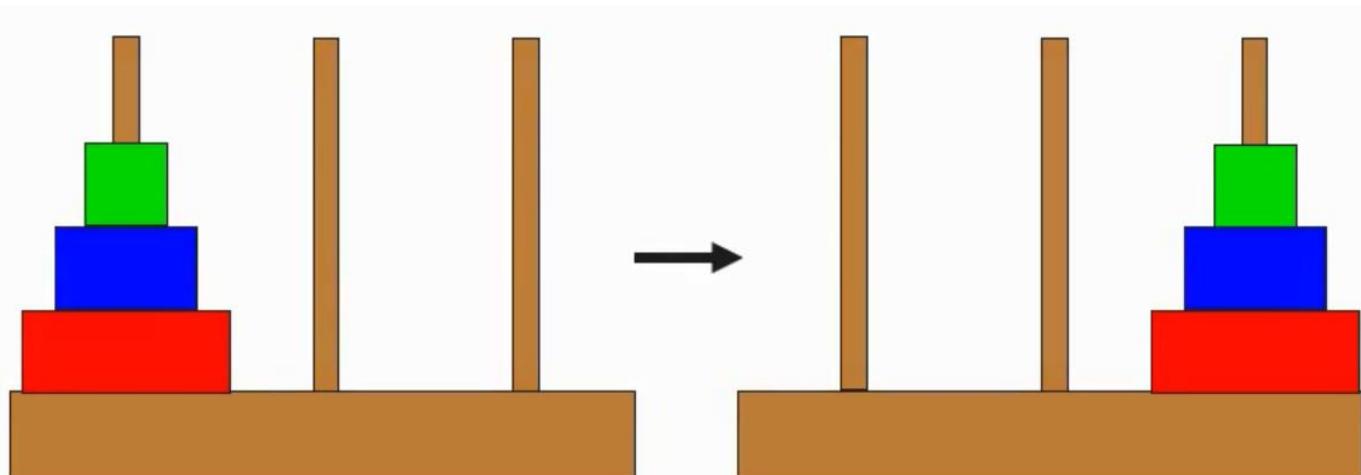
$$f(4, 1) = 2^{2^2} - 3$$

برج هانوی

سه میله A,B,C وجود دارد، که n مهره بر روی میله A قرار دارد. هدف انتقال n مهره به میله C است که برای این کار از میله B کمک گرفته می شود.

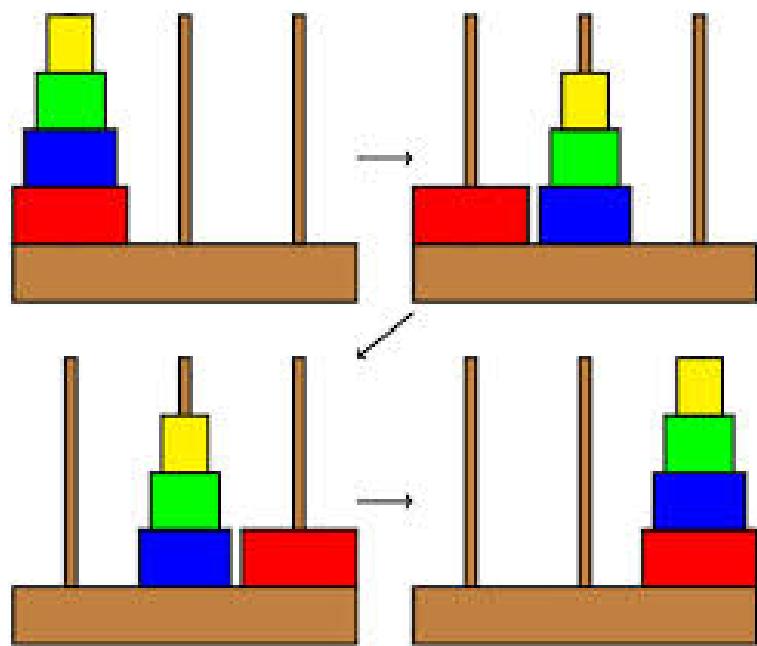
اندازه مهره ها در میله A از پایین به بالا کاهش می یابد.

در انتقال مهره ها هرگز مهره بزرگتر بر روی مهره کوچکتر نباید قرار بگیرد. در هر بار فقط امکان انتقال یک مهره وجود دارد که از بالای میله انتخاب می شود.



برج هانوی(ادامه)

در صورت وجود بیش از یک مهره ، مسئله را به صورت زیر حل می کنیم:



- ۱) انتقال $n-1$ مهره از میله A به میله B.
- ۲) انتقال ۱ مهره از میله A به میله C.
- ۳) انتقال $n-1$ مهره از میله B به میله C.



برج هانوی (ادامه)

الگوریتم انتقال n مهره از **A** به کمک **B**

tower (n , A, B, C)

{

if ($n == 1$)

 A **to** C;

else {

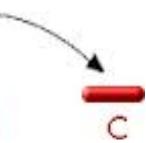
tower ($n-1$, A, C, B);

 A **to** C;

tower ($n-1$, B, A, C);

 }

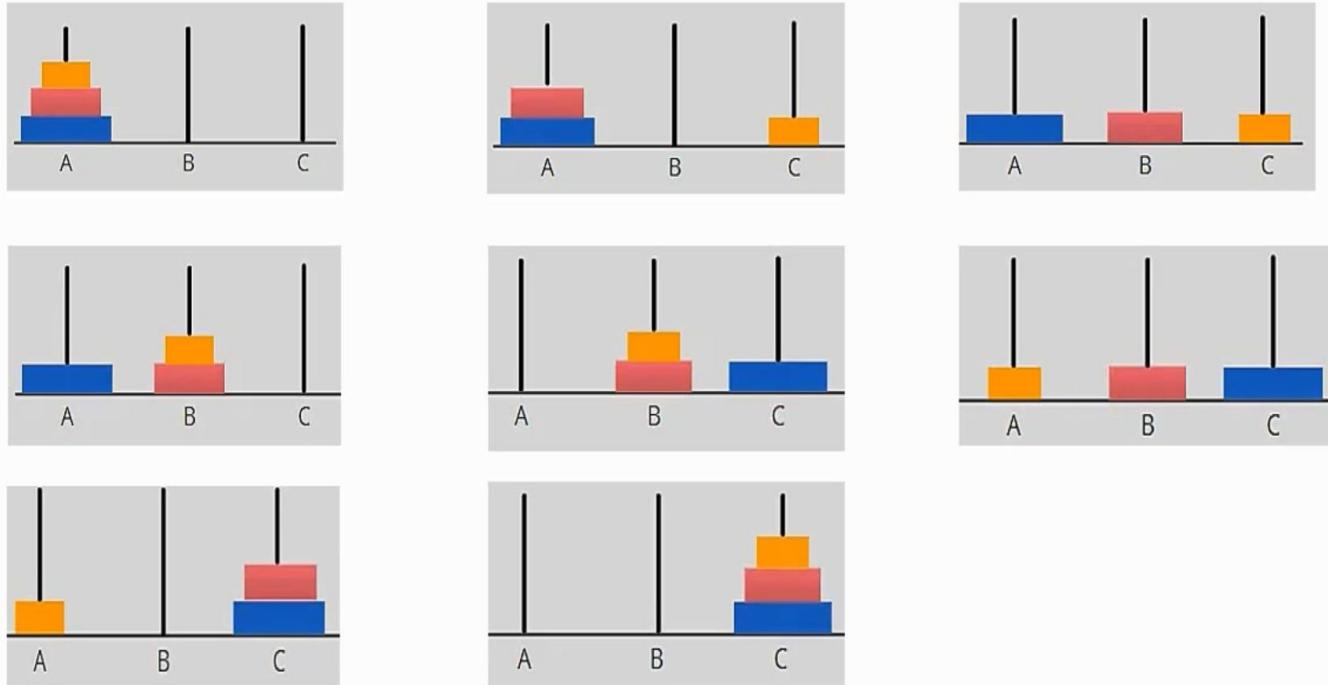
}



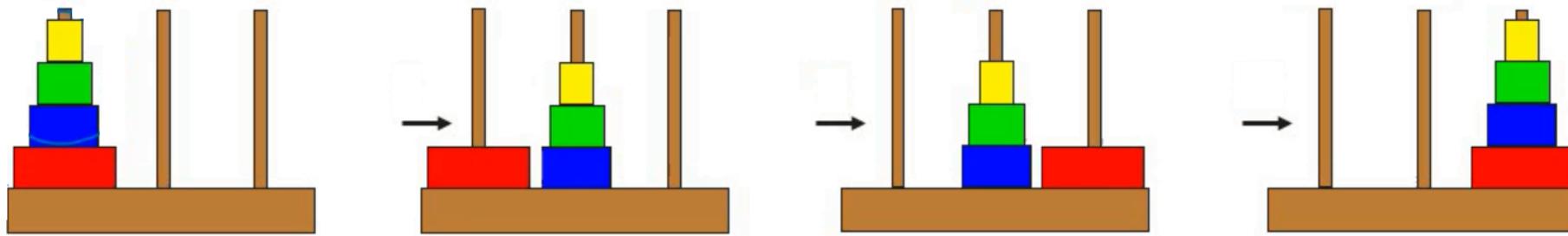
برج هانوی(مثال)

$$\left\{ \begin{array}{l} T(1, A, B, C) = A \rightarrow C \quad (1) \\ T(2, A, C, B) = \begin{cases} A \rightarrow B & (2) \\ T(1, C, A, B) = C \rightarrow B & (3) \end{cases} \\ A \rightarrow C \quad (4) \\ T(1, B, C, A) = B \rightarrow A \quad (5) \\ T(2, B, A, C) = \begin{cases} B \rightarrow C & (6) \\ T(1, A, B, C) = A \rightarrow C & (7) \end{cases} \end{array} \right.$$

انتقال 3 مهره از میله A به میله C به کمک میله B



رابطه بازگشتی برج هانوی



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

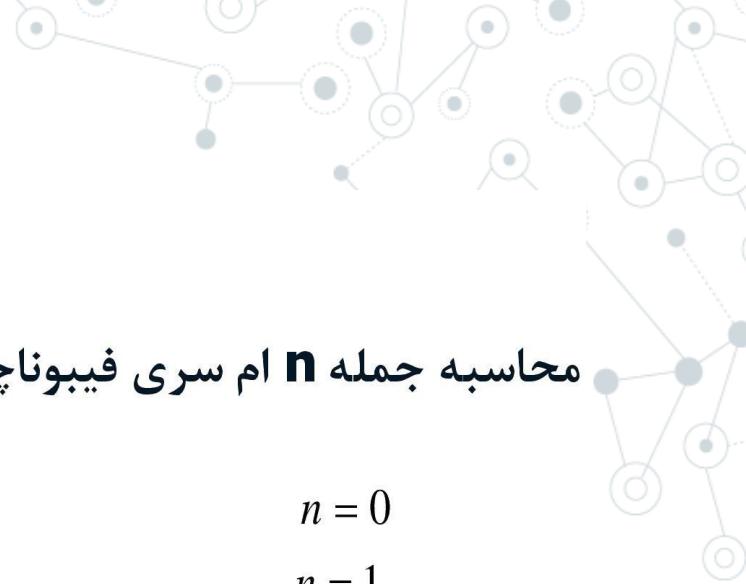
$$\Rightarrow a_n = 2^n - 1$$

تمرین

مسئله برج هانوی را با سه میله A، B و C در نظر بگیرید که در ابتدا n مهره به ترتیب در میله A قرار دارند. مهره کوچک تر بر روی مهره بزرگ تر قرار دارد.

می خواهیم با رعایت مقررات این مسئله، همه مهره ها را به میله B ببریم با این شرط که **نمی توان** هیچ مهره ای را یک راست از میله A به B یا از B به A برد.

اگر $T(n)$ کمینه تعداد حرکت مهره ها برای این مسئله باشد، رابطه بازگشتی برای این مسئله را بنویسید.



تابع فیبوناچی

محاسبه جمله n ام سری فیبوناچی : (0,1,1,2,3,5,8,13,21,34,...)

```
f(n){  
    if ((n==0) || (n==1) )  
        return n;  
    else  
        return f(n-1) + f(n-2);  
}
```

$$f(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f(n - 1) + f(n - 2) & n \geq 2 \end{cases}$$

محاسبه جمله چهارم سری فیبوناچی :

$$f(4)=f(3) + f(2) = 2+1=3$$

$$f(3)=f(2) + f(1) = 1+1=2$$

$$f(2)=f(1) + f(0) = 1+0=1$$

زاد و ولد خرگوش‌ها

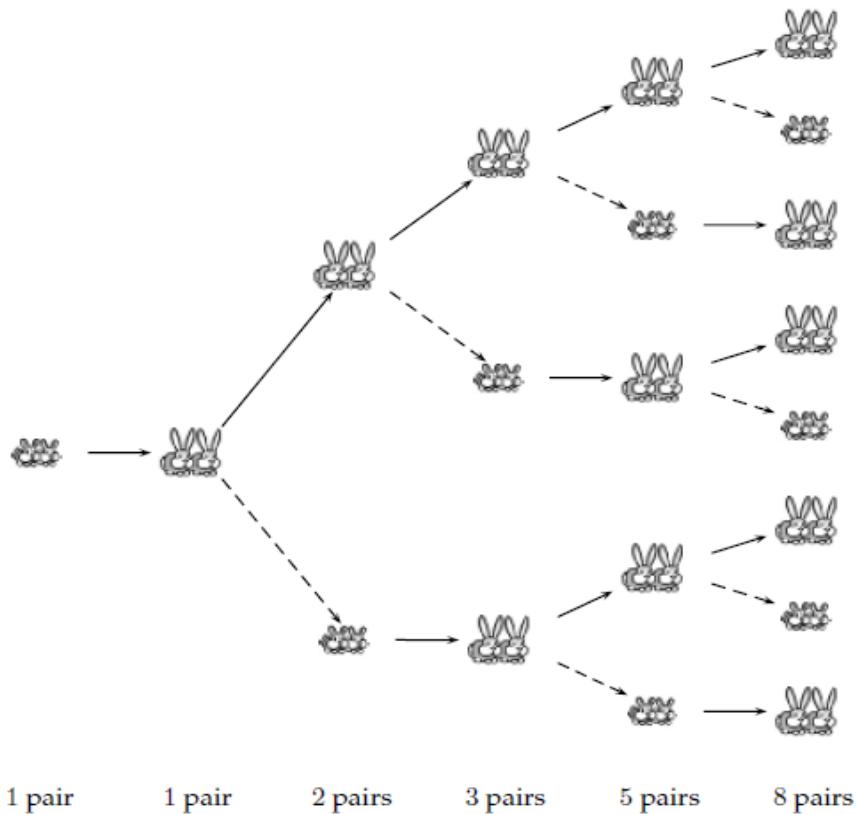
در یک جزیره یک جفت خرگوش نر و ماده نوزاد وجود دارد و مدل رشد جمعیت خرگوش‌ها به صورت زیر است:

- (۱) خرگوش‌ها یک ماه پس از تولد به سن بلوغ می‌رسند.
- (۲) دوران بارداری یک ماه است.
- (۳) هنگامی که خرگوش ماده به سن بلوغ می‌رسد حتماً باردار می‌شود.
- (۴) خرگوش ماده در هر بارداری، یک خرگوش نر و یک خرگوش ماده به دنیا می‌آورد.
- (۵) خرگوش‌ها هرگز نمی‌میرند.

رابطه بازگشتی بنویسید که تعداد خرگوش‌ها را در شروع ماه n ام نشان دهد؟



زاد و ولد خرگوش ها(ادامه)



تعداد جفت خرگوش ها در شروع ماه n ام برابر است با مجموع

دو مقدار زیر:

الف- تعداد جفت خرگوش های ماه قبل

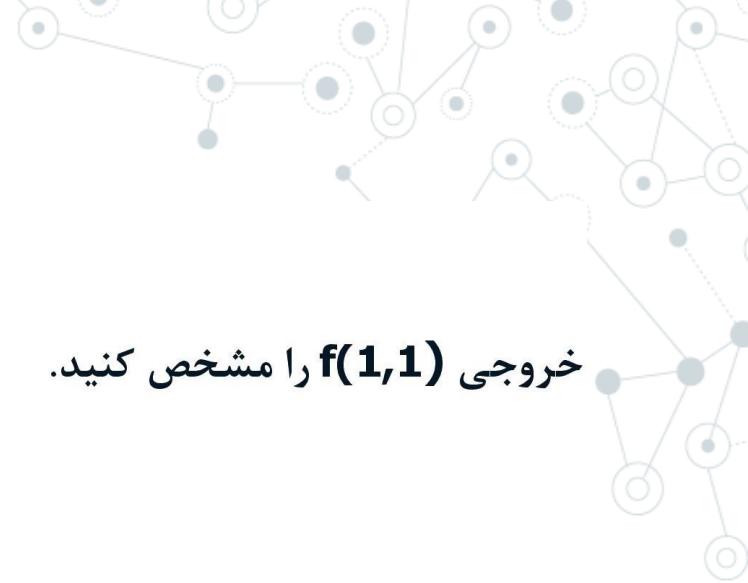
ب- تعداد جفت نوزادان (که برابر است با تعداد جفت خرگوش های بالغ ماه قبل که آن هم برابر است با کل خرگوش های دو ماه قبل) بنابراین داریم:

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

$$f(1) = 1, f(2) = 1$$

1,1,2,3,5,8,13,21,34,55,89,144,233

پاسخ:



تمرین

خروجی $f(1,1)$ را مشخص کنید.

$$f(x,0) = f(x+1,0) + f(x+1,1) \quad , \text{ if } x < 3$$

$$f(x,1) = 2f(x+1,0) + f(x+1,1) \quad , \text{ if } x < 3$$

$$f(3,0) = 1$$

$$f(3,1) = 0$$

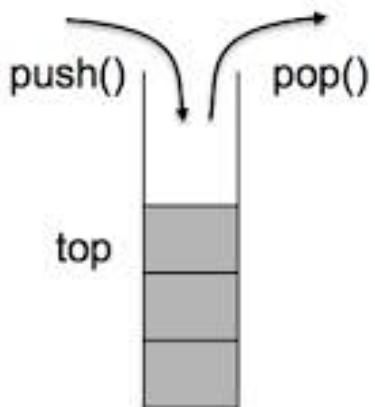


کاربرد پشته در زیر برنامه های بازگشتی

تعریف پشته: (stack)

پشته ساختمان داده ای است که حذف و اضافه از بالای آن انجام می شود.

پشته را **FILO** می نامند، چون آخرین عنصر وارد شده در آن، اولین عنصری است که از آن برداشته می شود.



عملیات در پشته و کاربردهای آن در فصل پشته به طور مفصل بررسی خواهند شد.

مثال

خروجی زیر برنامه زیر، به ازای فراخوانی $f(1)$ چیست؟

```
f(n){  
    if (n==3)  
        exit();  
    else{  
        n=n+1;  
        f(n);  
        cout<<n;  
    }  
}
```

$$f(1) \Rightarrow f(2) \Rightarrow f(3)$$



بنابراین خروجی برابر 32 خواهد بود.

مثال

خروجی $f(4)$ چه می باشد؟

```
f (n){  
    if (n >2)  
    {  
        f(n-1);  
        cout << 'a' ;  
        cout << 'b' ;  
    }  
}
```

$$f(4) \Rightarrow f(3) \Rightarrow f(2)$$



پاسخ: خروجی **abab** می باشد.

مثال

خروجی زیر برنامه زیر، به ازای فراخوانی $f(1,4)$ چیست؟

```
f(n,m){  
    if (n==3)  
        exit();  
    else {  
        n=n+1;  
        m=m+1;  
        f(n,m);  
        cout<<n;  
        cout<<m;  
    }  
}
```

$$f(1,4) \Rightarrow f(2,5) \Rightarrow f(3,6)$$

3
6
2
5

بنابراین خروجی برابر 3625 خواهد بود.

تمرین

خروجی $f(4)$ چیست؟

```
f(int x){  
    if (x)  
        g(x-1);  
    cout<<x;  
}
```

```
g(int y){  
    if (y)  
    {  
        cout<<y+1;  
        f(y-1);  
    }  
}
```

رابطه بازگشتی برای توابع بازگشتی

یک رابطه بازگشتی برای تعداد ضرب ها در تابع فاکتوریل بنویسید.

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

حل:

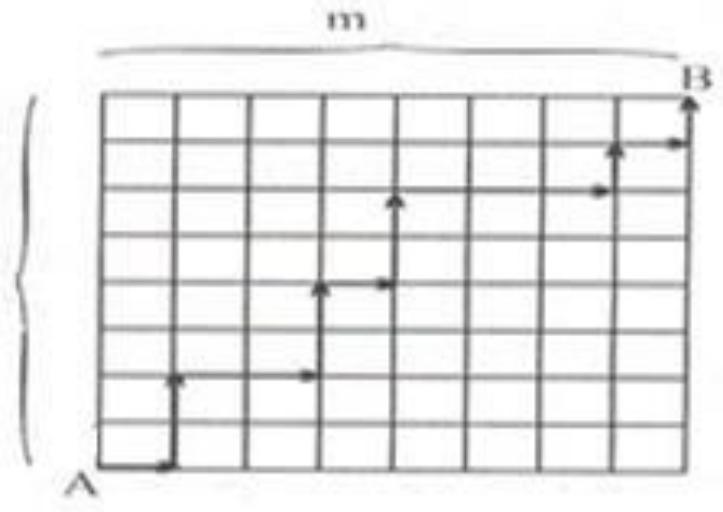
برای یک n معین تعداد ضرب هایی که انجام می شود برابر است با تعداد ضرب های انجام شده در فراخوانی $(n-1)$ به اضافه عمل ضرب n در $fact(n-1)$. بنابراین اگر تعداد ضرب های انجام شده برای یک مقدار معین n را با $T(n)$ نمایش دهیم، داریم:

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

$$T(0) = 0 \quad \text{وقتی } n=0 \text{ باشد، هیچ ضربی صورت نمی گیرد:}$$

حرکت در صفحه شطرنجی

در صفحه شطرنجی زیر می خواهیم از نقطه A به نقطه B برویم. حرکت های مجاز فقط به سمت راست یا بالا می باشد. یک رابطه بازگشتی برای $T(n,m)$ (تعداد مسیرهای ممکن از A به B) بنویسید.



$$T(n,m) = T(n, m - 1) + T(n - 1, m)$$

$$T(0,m) = 1$$

$$T(n,0) = 1$$

جواب این رابطه بازگشتی برابر است با:

$$\binom{m+n}{m}$$

مثال

اگر $T(n)$ تعداد ستاره های چاپ شده توسط $f(n)$ باشد، رابطه بازگشتی $T(n)$ را مشخص کنید.

```
f (n){  
    if (n>=2)  
    {  
        f (n-1);  
        f (n-1);  
        f (n-2);  
        cout<<"*";  
    }  
}
```

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

$$T(0) = 0$$

$$T(1) = 0$$

رابطه بازگشتی برای مسئله برج هانوی

اگر به تابع برج هانوی نگاه کنیم، مشاهده می شود که تابع دو بار خودش را فراخوانی می کند:

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

جواب این رابطه برابر است با:

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

تمرین

برای هر یک از مسئله های زیر یک رابطه بازگشتی بنویسید.

(۱) تعداد دنباله های ۰ و ۱ به طول n که هیچ دو صفر متوالی ندارند.

(۲) تعداد روش های فرش کردن یک صفحه شطرنجی با اندازه $n \times 2$ با موزاییک های 2×1

(۳) فرض کنید کدی سه تایی شامل ارقام ۰ و ۱ و ۲ با طول n زمانی معتبر است که تعداد صفرهای ظاهر شده در آن زوج باشد. رابطه بازگشتی بنویسید که تعداد کدهایی به طول n که معتبر هستند را محاسبه کند.

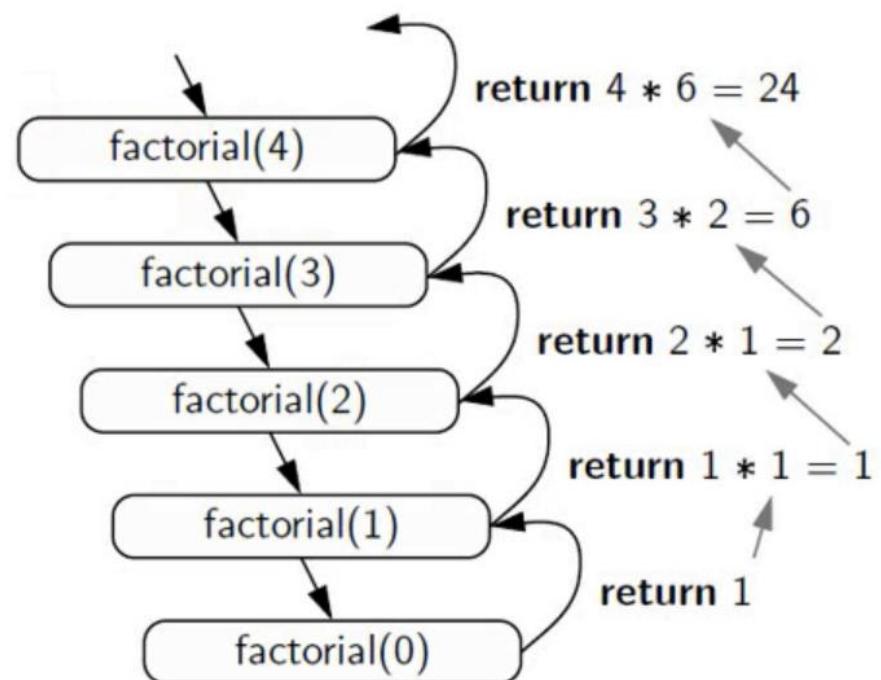


PYTHON TIME

تابع بازگشتی برای محاسبه فاکتوریل

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

$$T(n) = \begin{cases} 0 & n = 0 \\ 1 + T(n - 1) & n > 0 \end{cases}$$



جستجوی دودویی (غیر بازگشتی)

```
def binary_search_iterative(x, target):
    low = 0
    high = len(x) - 1
    while low <= high:
        mid = (low + high) // 2
        if target == x[mid]:
            return True
        elif target < x[mid]:
            high = mid - 1
        else:
            low = mid + 1
    return False
```



تابع برج هانوی

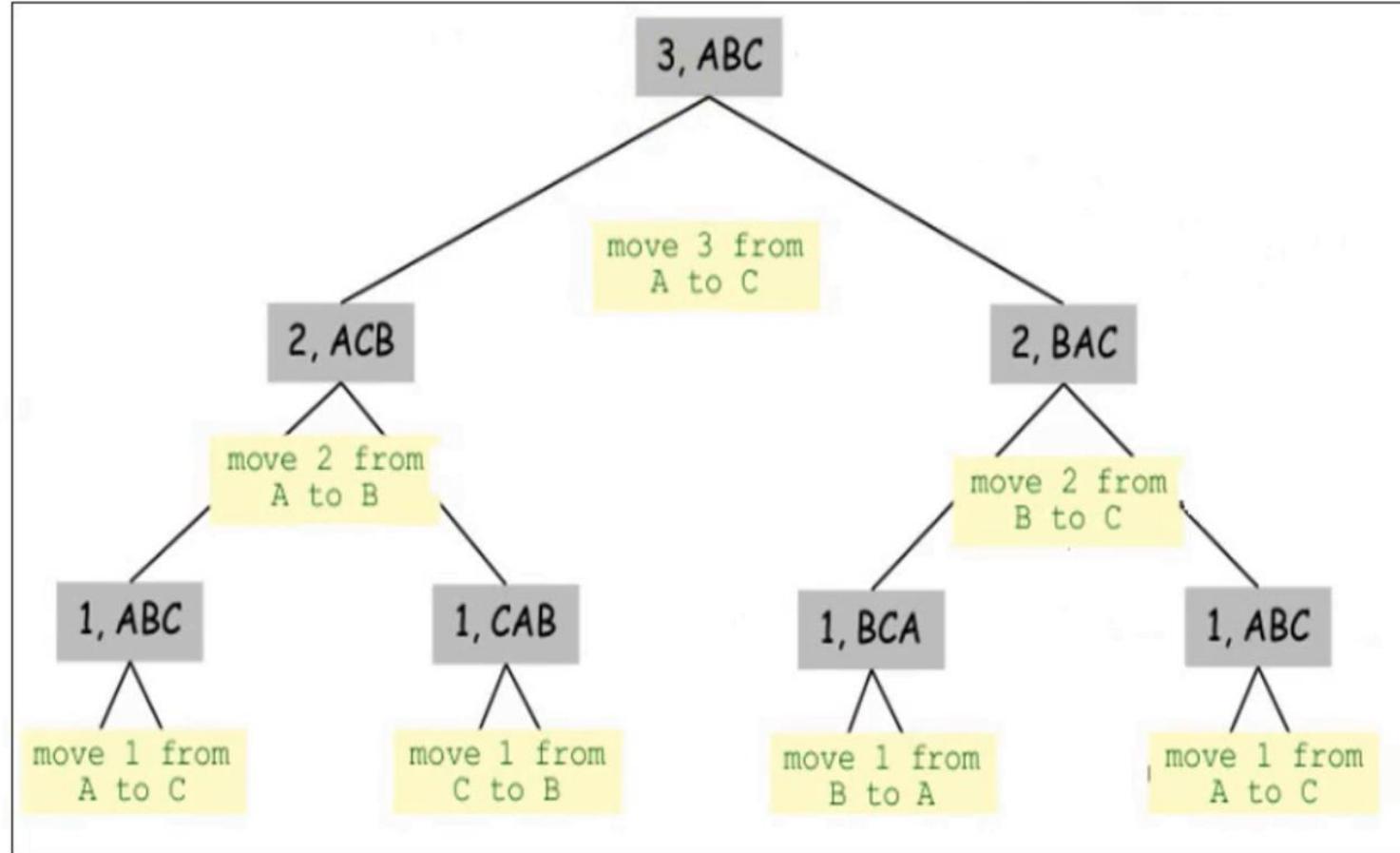
```
def hanoi(n, s, a, t):
    if(n == 1):
        print('Move 1 from', s, 'to', t)
        return
    hanoi(n-1, s, t, a)
    print('Move', n, 'from', s, 'to', t)
    hanoi(n-1, a, s, t)

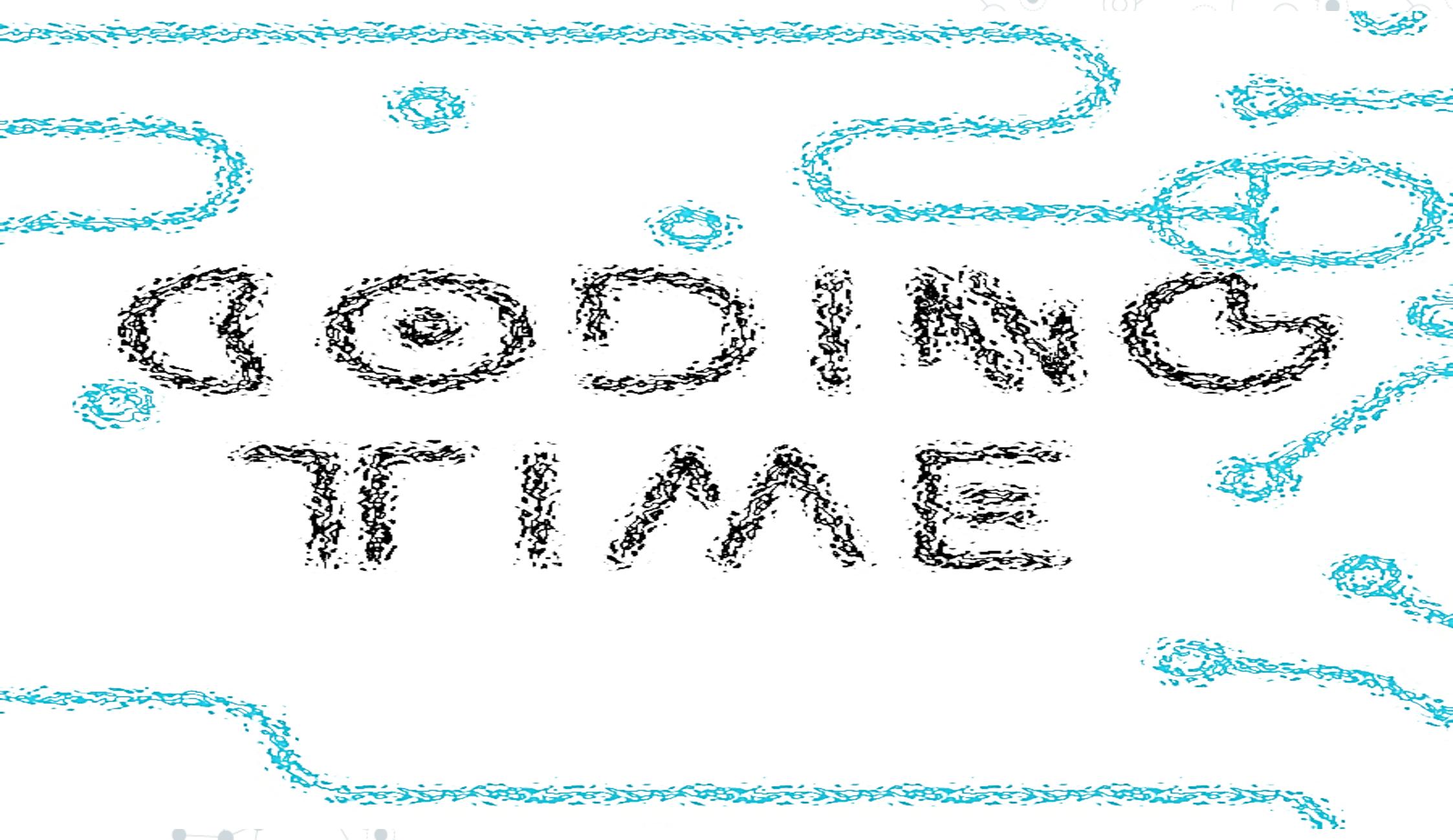
hanoi(3, 'A', 'B', 'C')
```



ردیابی برای سه مهره

```
Move 1 from A to C  
Move 2 from A to B  
Move 1 from C to B  
Move 3 from A to C  
Move 1 from B to A  
Move 2 from B to C  
Move 1 from A to C
```





روش‌های حل رابطه‌های بازگشتی

تکرار با جایگذاری

درخت بازگشت

قضیه اصلی

تکرار با جایگذاری

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

$$T(1) = 1$$

$$\begin{aligned}T(3) &= 3 + T(2) \\&= 3 + 2 + T(1) \\&= 3 + 2 + 1\end{aligned}$$

$$\begin{aligned}T(n) &= n + T(n-1) \\&= n + (n-1) + T(n-2)\end{aligned}$$

.

.

.

$$\begin{aligned}&= n + (n-1) + (n-2) + \dots + 2 + T(1) \\&= n(n+1)/2\end{aligned}$$

$$O(n^2)$$

مثال

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

$$T(1) = 1$$

$$\begin{aligned} T(3) &= 2T(2) + 1 \\ &= 2[2T(1) + 1] + 1 \\ &= 2^2 T(1) + 2 + 1 \\ &= 2^2 + 2 + 1 \\ &= 2^3 + 1 \end{aligned}$$

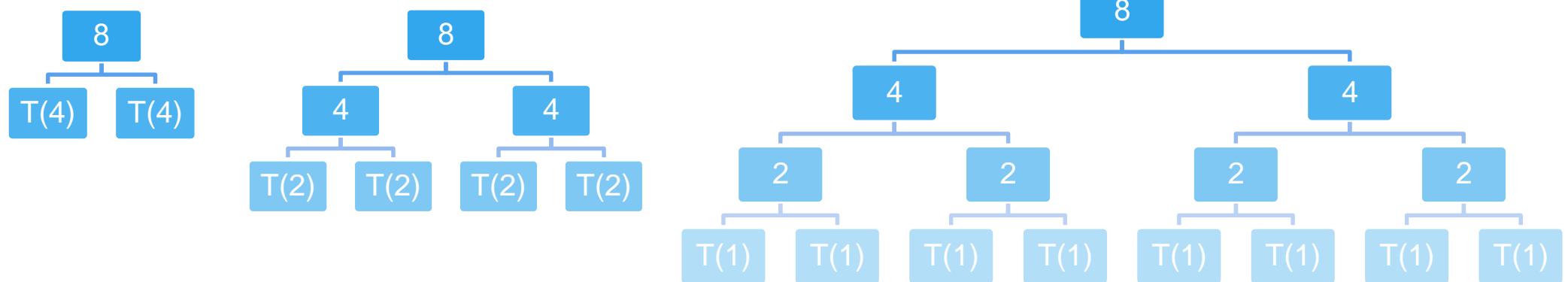
$$\begin{aligned} T(n) &= 2T(n - 1) + 1 \\ &= 2[2T(n - 2) + 1] + 1 = 2^2 T(n - 2) + 2 + 1 \\ &= 2^2 [2T(n - 3) + 1] + 2 + 1 \\ &= 2^3 T(n - 3) + 2^2 + 2 + 1 \\ &\quad \vdots \\ &= 2^{n-1} T(n - (n - 1)) + \cdots + 2^2 + 2 + 1 \\ &= 2^{n-1} + \cdots + 2^2 + 2 + 1 = 2^n - 1 = O(2^n) \end{aligned}$$

درخت بازگشت

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

$$T(1) = 1$$

$$\begin{aligned} T(8) &= 8 + 2T(4) \\ &= 4 + 2T(2) \\ &= 2 + 2T(1) \end{aligned}$$

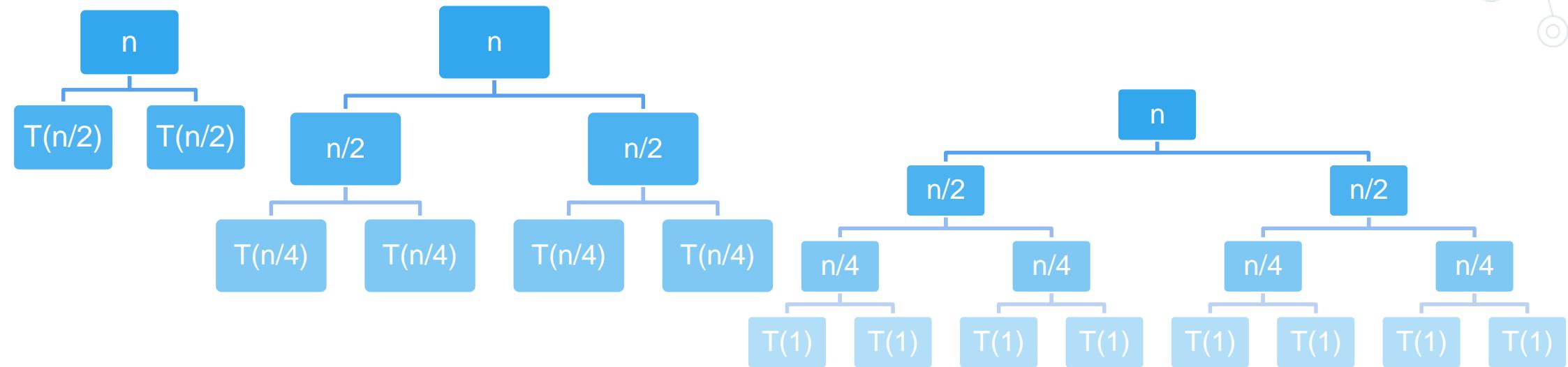


$$level = \log_2^n + 1$$

درخت بازگشت

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

$$T(1) = 1$$

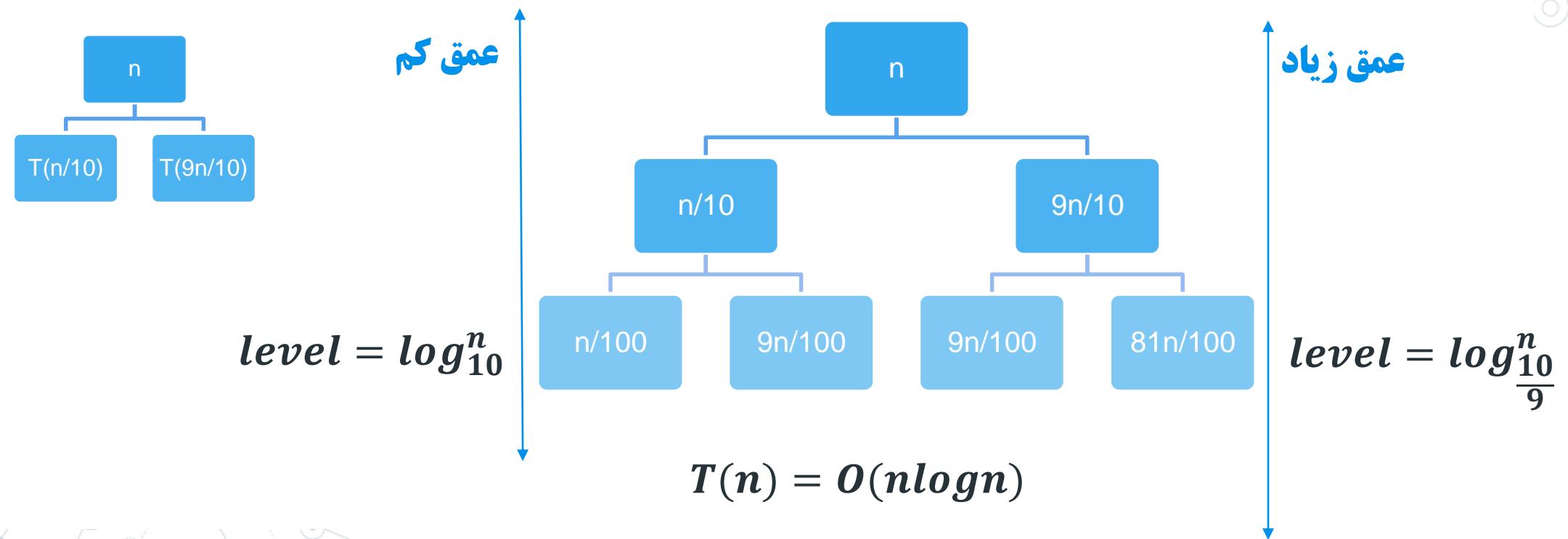


$$T(n) = (\log_2^n + 1) \times n = n \log_2^n + n = O(n \log_2^n)$$

مثال

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + cn$$

$$\begin{aligned} T(100) &= T(10) + T(90) + 100 \\ &= \dots \end{aligned}$$





قضیه اصلی

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$$a \geq 1, b > 1$$

$$T(n) = \begin{cases} \theta(n^{\log_b^a}) & f(n) < n^{\log_b^a} \\ \theta(f(n) \cdot \lg n) & f(n) = n^{\log_b^a} \\ \theta(f(n)) & f(n) > n^{\log_b^a} \end{cases}$$



مثال

$$T(n) = 4T\left(\frac{n}{2}\right) + \lg n$$

$$n^{\log_2^4} \geq \lg n$$

$$\theta(n^2)$$

مثال

$$T(n) = 3T\left(\frac{n}{4}\right) + n \lg n$$

$$n^{\log_4^3} \leq n \lg n$$

$$\theta(n \lg n)$$



مثال

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

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

$$n^{\log_3 \frac{1}{2}} = 1 \rightarrow 1 = 1$$

$\theta(lgn)$





تشریح

سوال

a.golzari@azaruniv.ac.ir

a.golzari@tabrizu.ac.ir

<https://github.com/Amin-Golzari-Oskouei>