Recursion

Recursive Function

- به تابعی گفته می شود که خودش را بصورت مستقیم و یا غیر مستقیم فراخوانی می کند.
- با کمک توابع بازگشتی به نوعی می توانیم ماهیت تکرار را پیاده سازی کنیم.
- هر تابع بازگشتی شامل دو قسمت می باشد:
  - **Base case – حالت مبنا**
    - شرط توقف اجرای تابع بازگشتی
  - **Recursive step– حالت بازگشت**
    - تمام زنجیره های تکرار که خلاصه در یک مرحله به base case می رسند.
- مثال تابع factorial (n!)

$$n! = \begin{cases} 1 & n = 1 \\ \mathrm{n} \, * (n-1)! & if\ \mathrm{n} > 1 \end{cases}$$

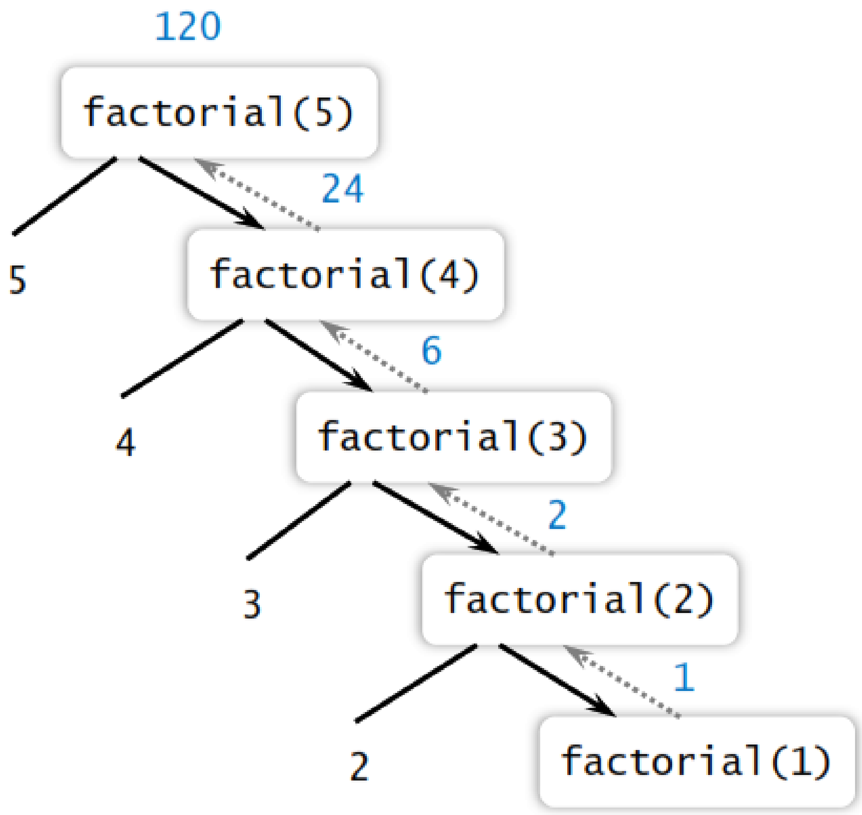
Factorial - مثال

• محاسبه فاکتوریل n

$$n! = 1 \ * 2 \ * 3 \ * \ \ldots * (n-1) * n$$

$$n! = \begin{cases} 1 & n = 1 \\ \text{n} \ * (n-1)! & if \ \text{n} > 1 \end{cases}$$

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



مثال - Fibonacci

- دنباله فیبوناچی

$$0,1,1,2,3,5,8,13,21,34,55, \ldots$$

$$fib(n) = \begin{cases} 0 & if\ n = 0 \\ 1 & if\ n = 1 \\ fib(n-1) + fib(n-2) & otherwhise \end{cases}$$

```
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fib(n-1) + fib(n-2)
```

بزرگترین مقسوم علیه مشترک - مثال

- الگوریتم اقلیدوس

$$\gcd(p,q) = \begin{cases} \text{p} & q = 0 \\ \gcd(q, p\%q) & otherwhise \end{cases}$$

```
def gcd(p, q):
    if q == 0:
        return p
    return gcd(q, p%q)
```

مثال - Binomial Coefficient

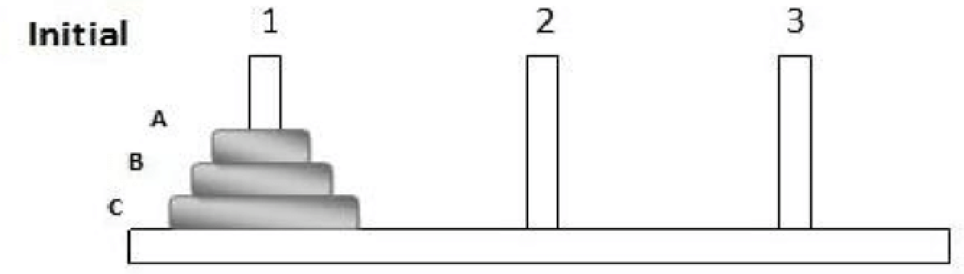
- ضریب دوجمله ای

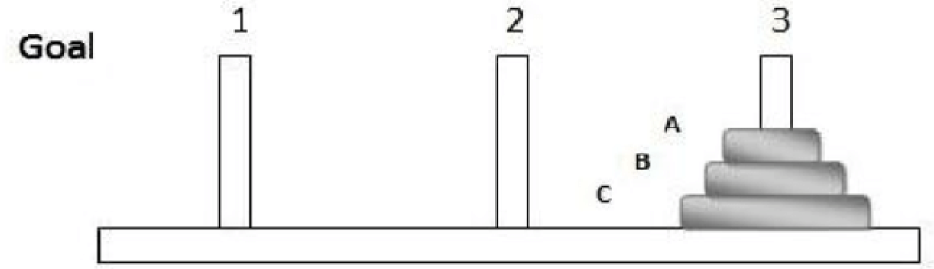
$$\binom{n}{k} = \binom{n}{k-1} + \binom{n-1}{k-1}$$

$$\binom{n}{0} = \binom{n}{n} = 1$$

```
def bin(n):
    if k == 0 or k == n:
        return 1
    return bin(n-1,k)+bin(n-1,k-1)
```

برج های هانوی

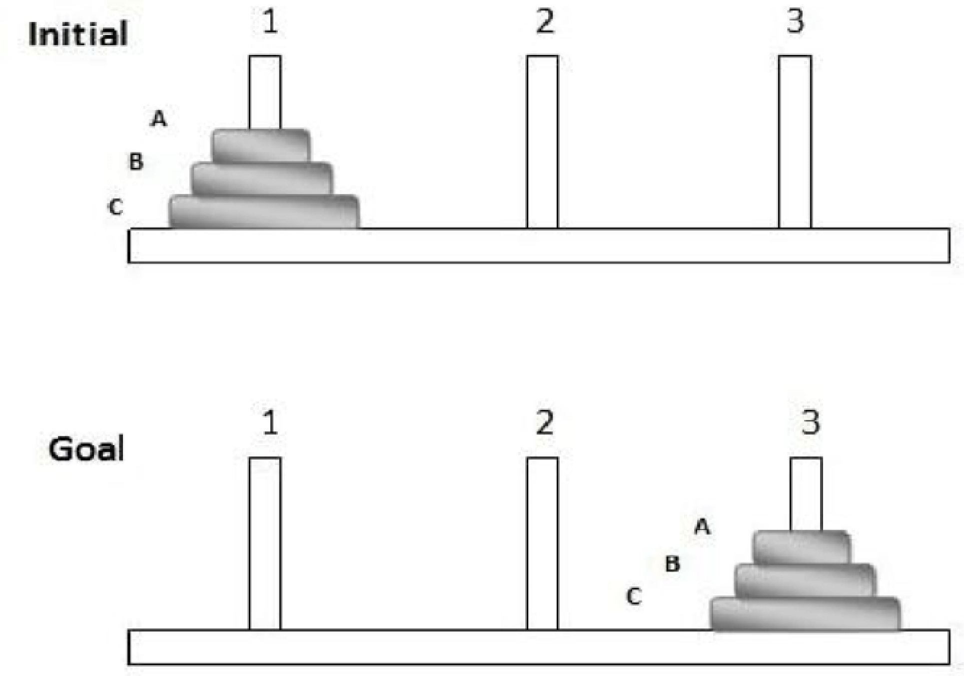




- هدف انتقال دیسک ها از میله سمت چپ به میله سمت راست با حفظ ترتیب است.

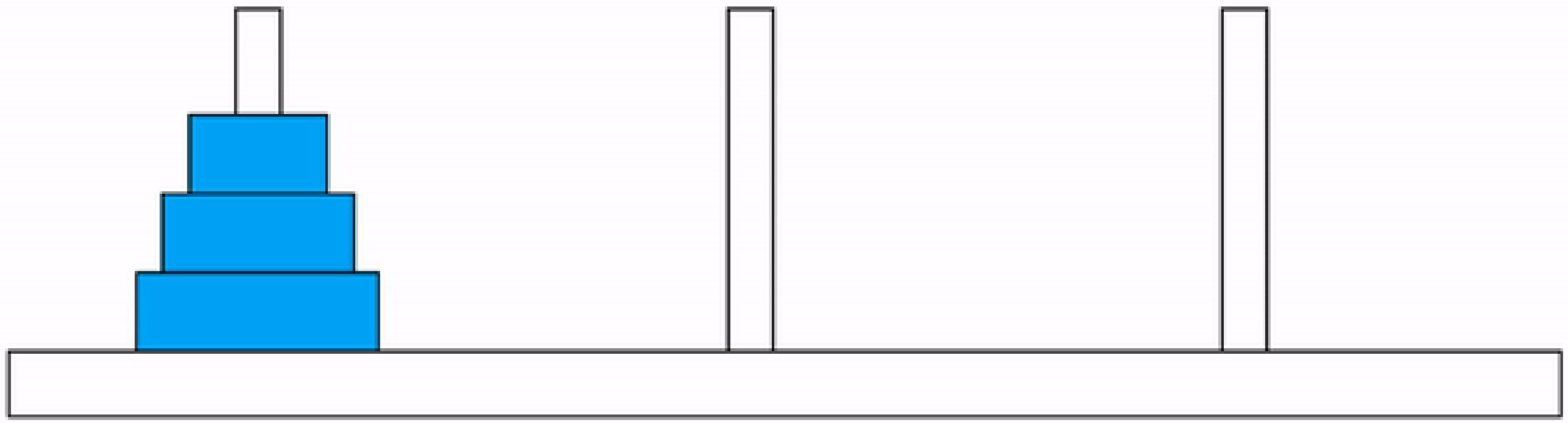
- قوانین:
  - در هر حرکت تنها مجاز به انتقال یک دیسک هستیم.
  - هیچ گاه مجاز نیستیم یک دیسک بزرگ تر را بروی یک دیسک کوچکتر قرار دهیم.

برج های هانوی



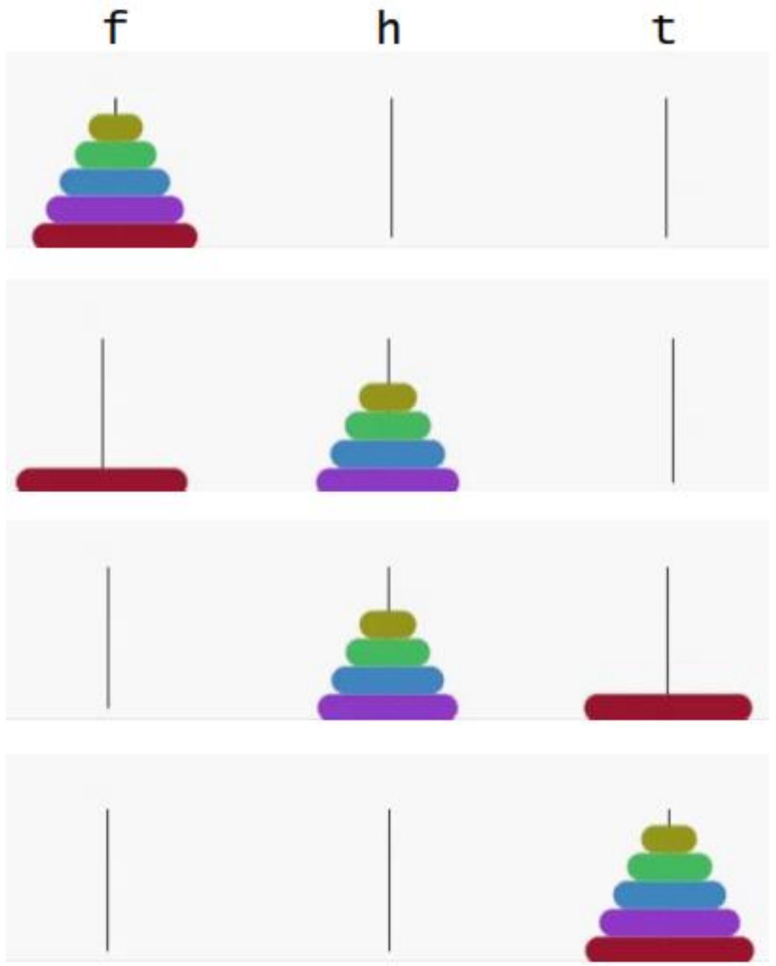
- **افسانه زمان پایان دنیا ...**
- ۶۴ دیسک طلا بر روی سه میله الماس در ابتدای آفرینش
- با اتمام کار، دنیا به پایان می رسد!
- **آیا می توان از کامپیوتر برای پیش بینی این زمان استفاده کرد؟**

برج های هانوی

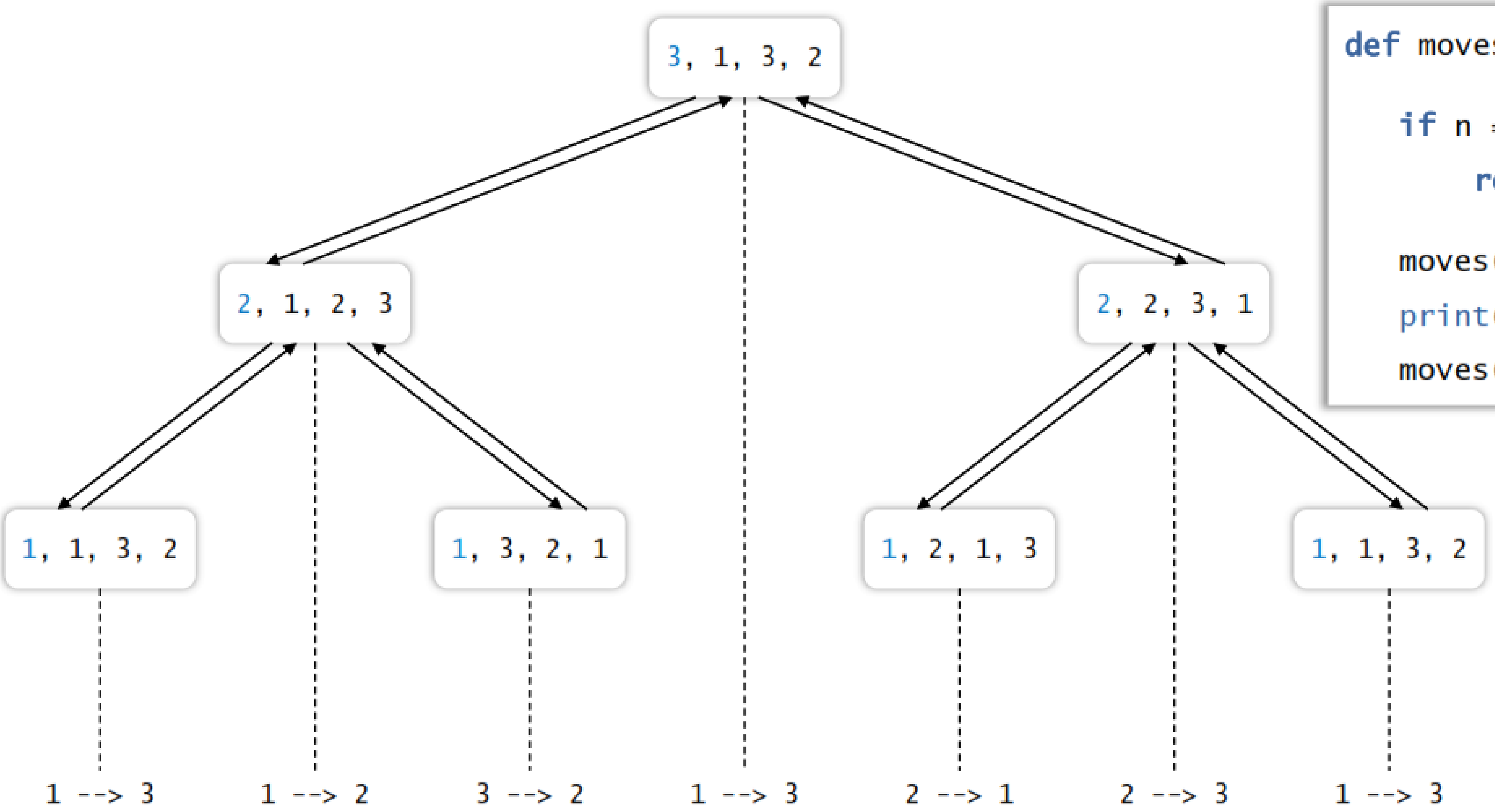


برج های هانوی، راه حل بازگشتی

```
def moves(n,f,t,h):
    if n == 0:
        return
    moves(n-1,f,h,t)
    print(str(f)+'→'+str(t))
    moves(n-1,h,t,f)
```



برج های هانوی، راه حل بازگشتی




```
def moves(n, f, t, h):

    if n == 0:
        return

    moves(n - 1, f, h, t)
    print(str(f) + ' --> ' + str(t))
    moves(n - 1, h, t, f)
```

برج های هانوی - ویژگی های راه حل

- برای حل یک مساله با n دیسک به 2 بتوان n جابجایی نیاز است !!
- اگر فرض کنیم که هر جابجایی تنها به یک ثانیه زمان نیاز داشته باشد، پس دنیا ۵۸۵ میلیارد سال پس از خلقت آن به پایان می رسد.
- این زمان چگونه بدست می آید؟؟

مزایا و معایب توابع بازگشتی

مزایا

- ایجاد نظم و وضوح در کد
- قابلیت شکستن مساله به چند مساله کوچکتر
- آسان تر نسب به حلقه های تودرتو

معایب:

- مصرف حافظه بسیار زیاد
- زمانبر بودن اجرا
- سخت بودن درک جریان اجرا
- ساده نبودن اشکال زدایی

Memoization

- عموما گام بازگشتی نتایج یکسانی را در مراحل مختلف ایجاد می کند و این تکرار محاسبات برای بدست آوردن نتایج یکسان می تواند باعث ناکارآمدی توابع بازگشتی شود.
- برای حل این مشکل می توان قسمتی از نتایج را برای استفاده آینده ذخیره نمود. که به این تکنیک **Momoization** گفته می شود.

```
memo = {}
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    if n in memo:
        memo[n]
    memo[n] = fib(n-1)+fib(n-2)
    return memo[n]
```

مثال - Fast Exponentiation

- فرض کنید ماتریسی داریم به نام A و می خواهیم این ماتریس را به توان n برسانیم.
- **ساده ترین راه حل:** فرض کنید تابعی داریم به نام mat_prod که دو ماتریس را گرفته و آنها را در هم ضرب می کند. اگر این تابع را n بار اجرا کنیم در واقع ماتریس به توان n رسیده است. (**بسیار ناکارآمد!**)
- راه حل بهینه می تواند استفاده از **تکنیک تقسیم و غلبه** باشد.

$$A^n = \begin{cases} A^{n/2} * A^{n/2} & if\ n\ is\ even \\ A^{\lfloor n/2 \rfloor} * A^{\lfloor n/2 \rfloor} * A & if\ n\ is\ odd \\ I \quad if\ n = 0 & \end{cases}$$

مثال - Fast Exponentiation

```
def exp(A,n):
    if n == 0:
        return [[1,0],[0,1]]
    temp = exp(A, n//2)
    prod = mat_prod(temp,temp)
    if n%2 == 0:
        return prod
    else:
        return mat_prod(prod,A)
```