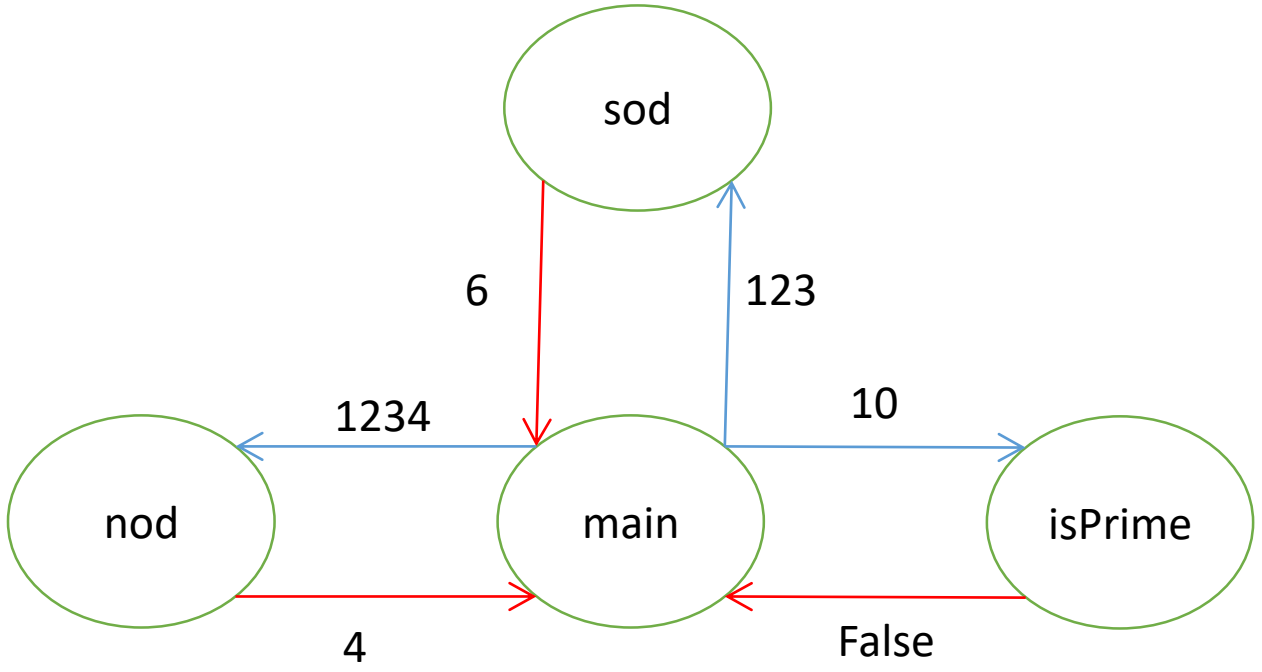
برنامه سازی پیشرفته
(توابع: معرفی و کاربرد)

صادق اسکندری – دانشکده علوم ریاضی، گروه علوم کامپیوتر

eskandari@guilan.ac.ir

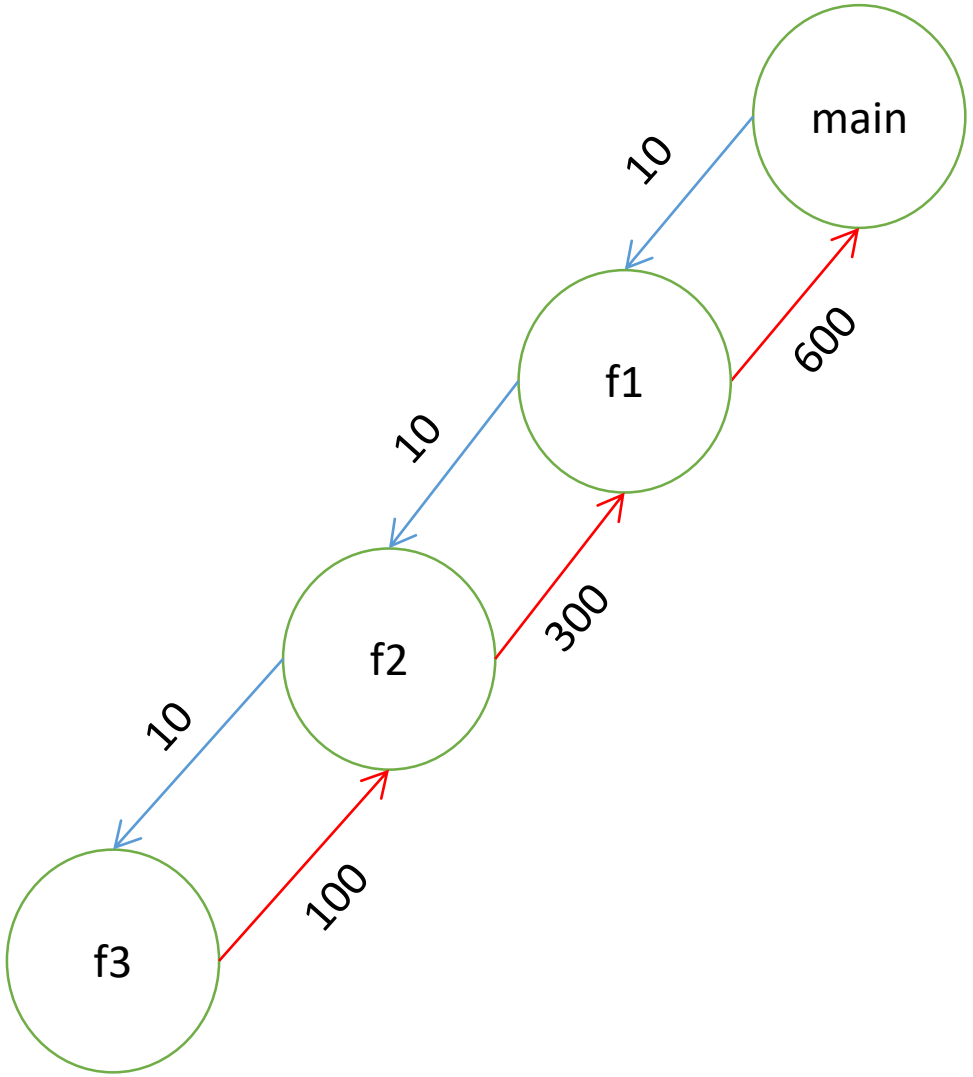
بازگشتی: درخت فراخوانی توابع

```python
def nod(n):
    nod = 1
    while n >= 10:
        nod += 1
        n //= 10
    return nod

def sod(n):
    sum = 0
    while(n != 0):
        sum += n%10
        n //= 10
    return sum

def isPrime(n):
    for i in range(2,n):
        if n%i == 0:
            return False
    return True

print(isPrime(sod(123) + nod(1234) ))
```



بازگشتی: درخت فراخوانی توابع

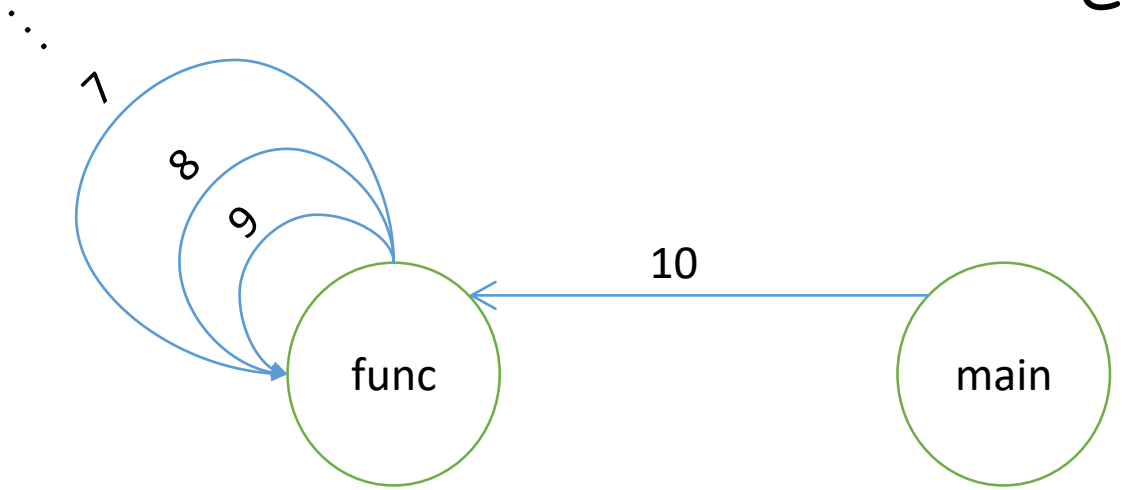
```python
def f1(n):
    a = 2*f2(n)
    return a

def f2(n):
    x = 3*f3(n)
    return x

def f3(n):
    return n**2

f1(10)
```



بازگشتی: تعریف

فراخوانی یک تابع به وسیله خودش

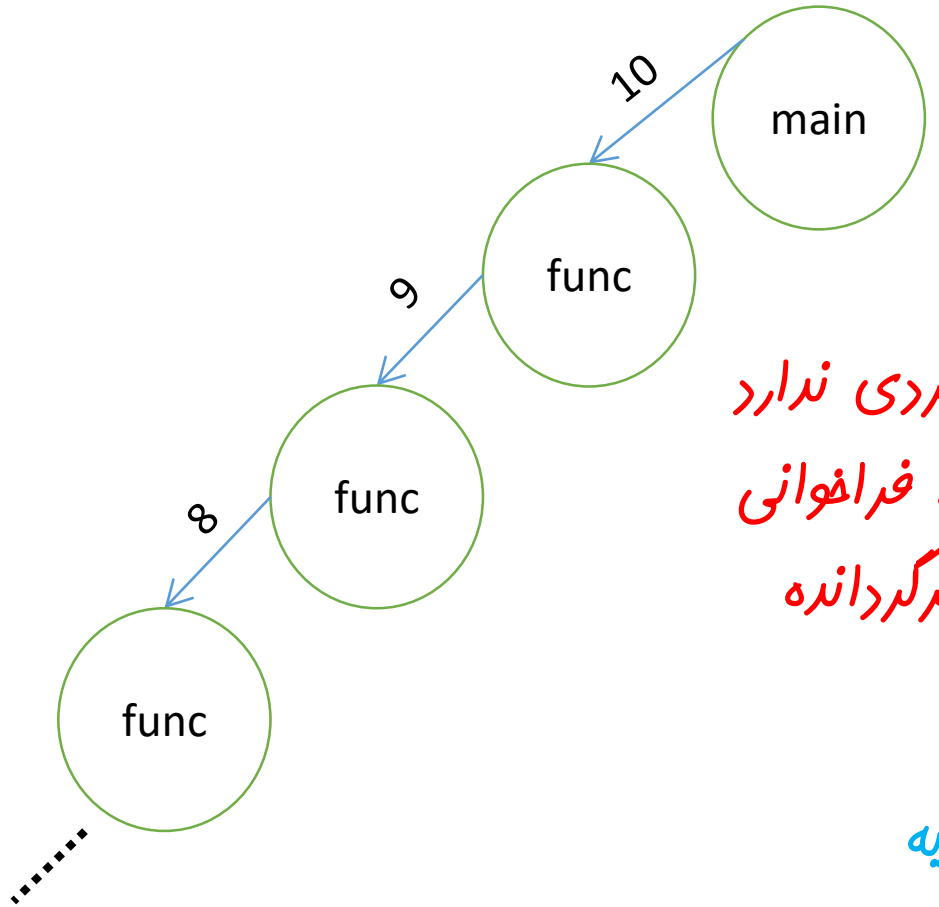
```
1  def func(n):
2      a = func(n-1)
3      return a
4
5  func(10)
```



بازگشتی: تعریف

فراخوانی یک تابع به وسیله خودش

```
1  def func(n):
2      a = func(n-1)
3      return a
4
5  func(10)
```



بازگشتی به این شکل، هیچ کاربردی ندارد
زیرا تابع می تواند تا ابد خود را فراخوانی
کند و در نتیجه، هیچگاه مقداری برگردانده
نمی شود ☹

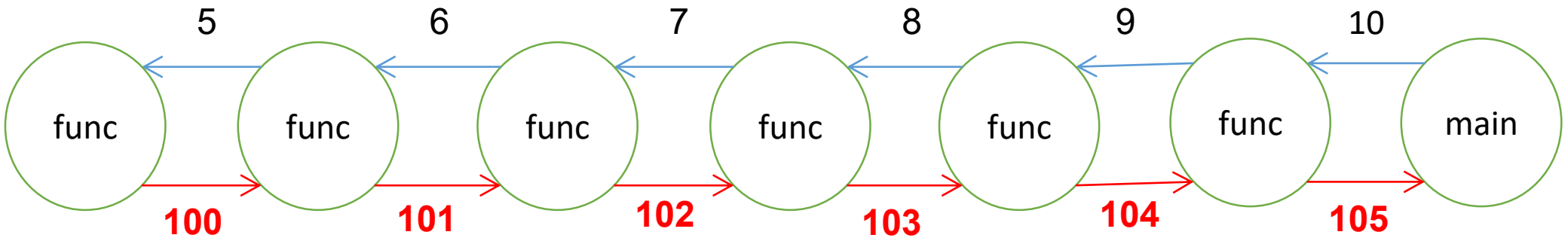
سوال: راهکار چیست؟ مقدار پایه

بازگشتی: مقدار پایه

```
1  def func(n):
2      if n == 5:
3          return 100
4      a = 1+func(n-1)
5      return a
6
7  func(10)
```

بررسی مقدار پایه:

مقدار پایه: ۵



تمرین: درخت های فراخوانی هر یک از توابع زیر را رسم نموده و خروجی نهایی (آنچه که در صفحه نمایش نشان داده می شود) را برای هر یک مشخص کنید.

```python
def f(n):
    return 1+n**2

def func(n):
    if n == 5:
        return f(n)
    a = f(n) + func(n-1)
    return a

func(10)
```

```python
def func(n):
    print(n)
    if n == 5:
        return 100
    a = 1 + func(n-1)
    print(n)
    return a

func(10)
```

بازگشتی: کاربرد

برای حل یک مسئله در قالب بازگشتی، نیاز به یک رابطه بازگشتی داریم.

رابطه بازگشتی جمله ای است که یک مسئله با ورودی های داده شده را در قالب یک یا چند مسئله با ورودی های کوچکتر بیان می کند.

$$f(n) = \begin{cases} g(n) : base\ case = True \\ f(t) : base\ case = False \end{cases}$$

پایه بازگشتی: g موجب فراخوانی مجدد f نمی شود.

اسقرای بازگشتی: تابع f مجددا با ورودی های جدید فراخوانی می شوند. غالبا t نسبت به n به پایه نزدیکتر است.

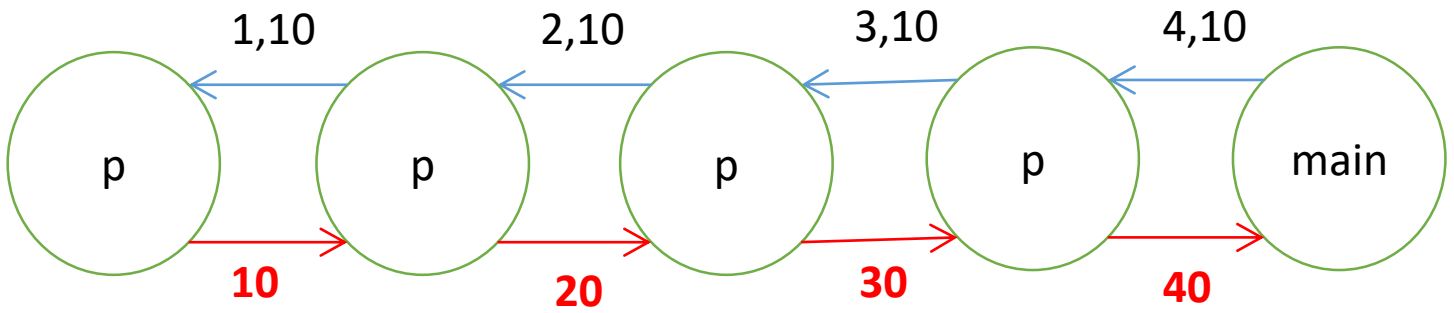
بازگشتی: کاربرد

مثال: رابطه بازگشتی برای عمل ضرب

$$p(x,y) = x \times y$$

$$p(x,y) = \begin{cases} y & if\ x == 1 \\ y + p(x-1,y) & otherwise \end{cases}$$

```
1  def p(x,y):
2      if x == 1:
3          return y
4      else:
5          return y + p(x-1,y)
6
7  print(p(4,10))
```



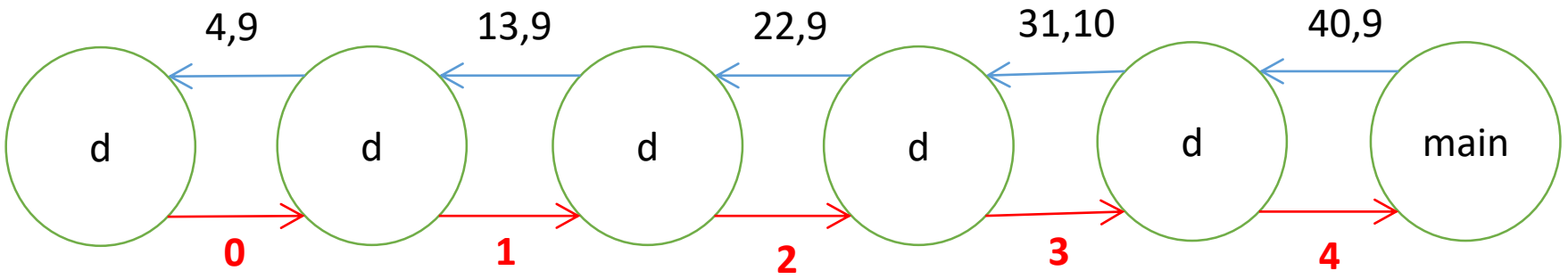
بازگشتی: کاربرد

مثال: رابطه بازگشتی برای عمل تقسیم

$$d(x,y) = x//y$$

$$d(x,y) = \begin{cases} 0 & if\ x < y \\ 1 + d(x-y,y) & otherwise \end{cases}$$

```
1  def d(x,y):
2      if x < y:
3          return 0
4      else:
5          return 1+d(x-y,y)
6
7  print(d(40,9))
```

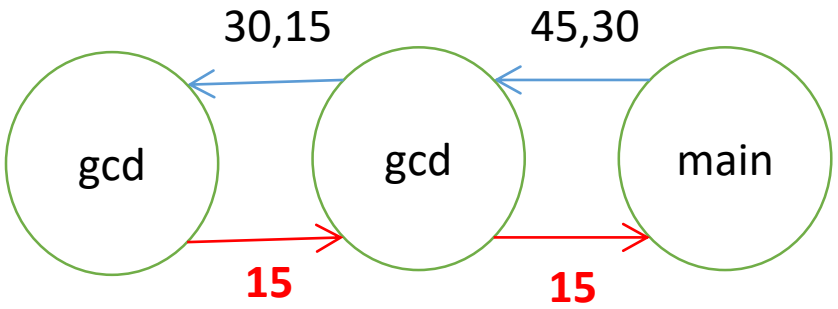


بازگشتی: کاربرد

مثال: رابطه بازگشتی برای بزرگترین مقسوم علیه مشترک (الگوریتم اقلیدس)

$$gcd(x,y) = \begin{cases} y & if\ \ x\%y == 0 \\ \gcd(y, x\%y) & otherwise \end{cases}$$

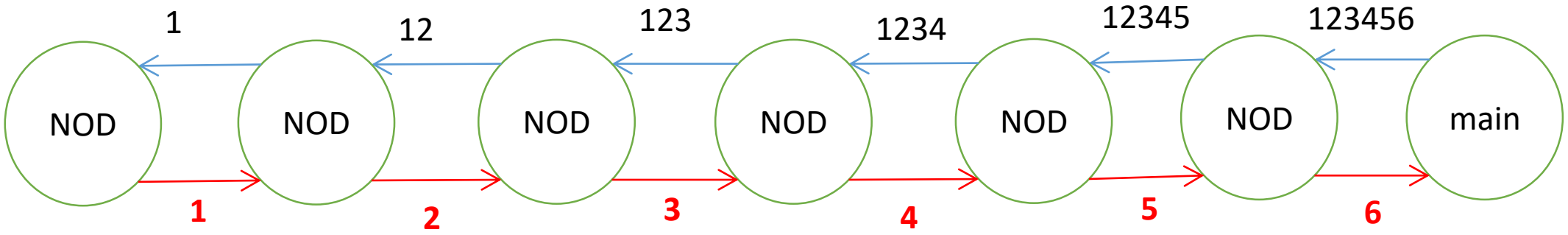
```
1  def gcd(x,y):
2      if x%y == 0:
3          return y
4      else:
5          return gcd(y,x%y)
6
7  print(gcd(45,30))
```



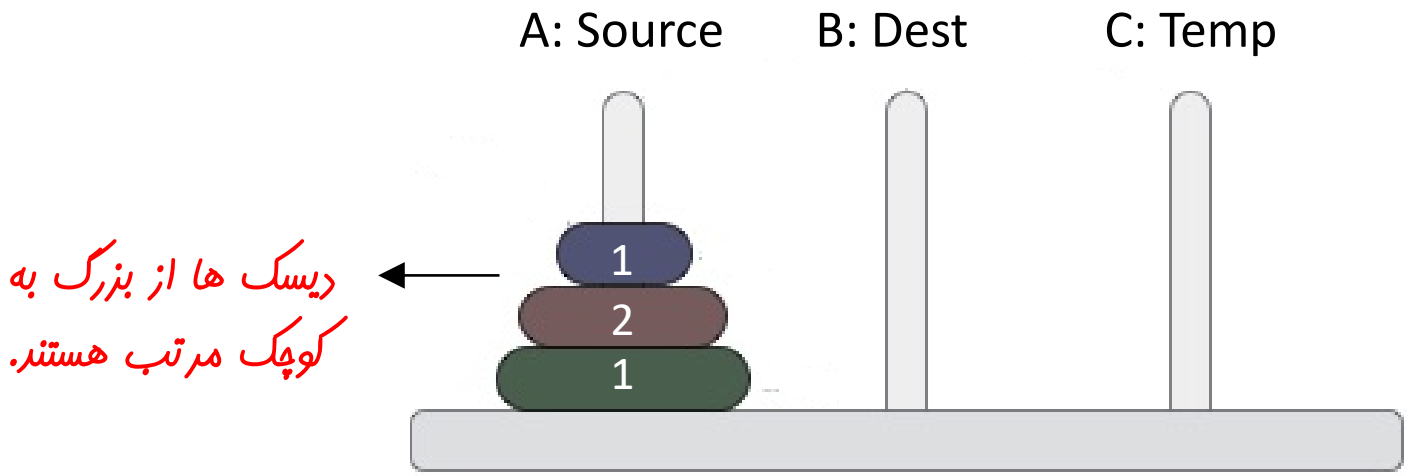
بازگشتی: مثالهای بیشتر

رابطه بازگشتی برای محاسبه تعداد ارقام یک عدد صحیح

```python
def NOD(n):
    if n < 10:
        return 1
    else:
        return 1 + NOD(n/10)

print(NOD(123456))
```



برجهای هانوی: توصیف مسئله

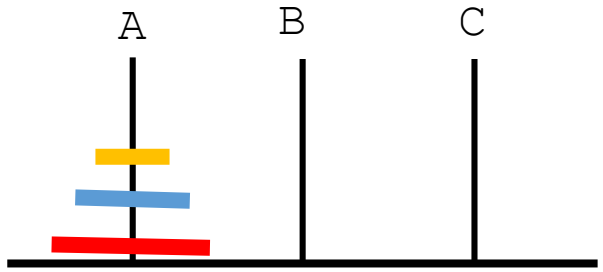


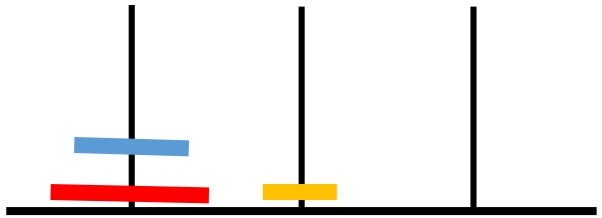
با فرض این که n تا دیسک بر روی برج A داده شده باشد، یک راهکار برای انتقال این دیسک ها به برج B با کمک برج C ارائه کنید به شرط آن که:

- در هر حرکت تنها یک دیسک حرکت داده شود.
- هیچگاه یک دیسک بر روی یک دیسک کوچکتر قرار نگیرد.

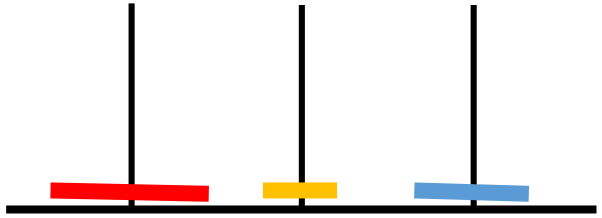
بازگشتی: مثالهای بیشتر

برجهای هانوی: مثال برای $n = 3$

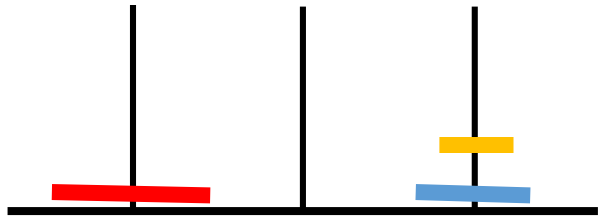




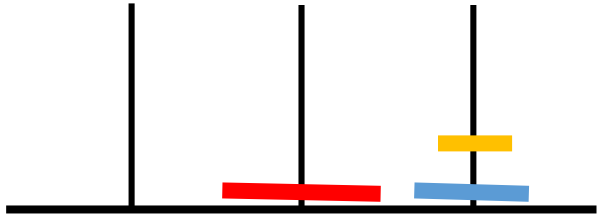
1 move 1 from A to B



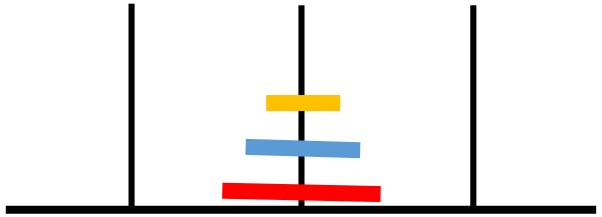
2 move 2 from A to C



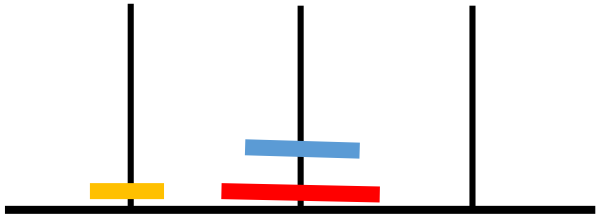
3 move 1 from B to C



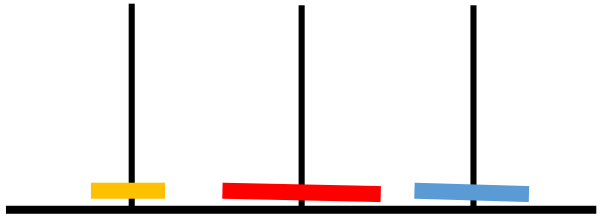
4 move 3 from A to B



7 move 1 from A to B



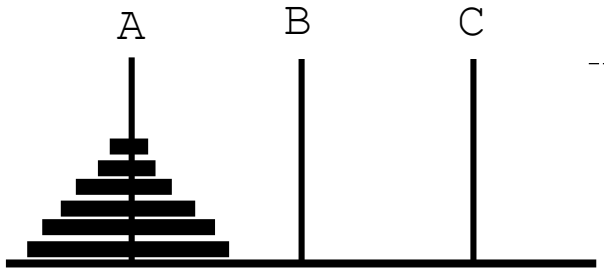
6 move 2 from C to B



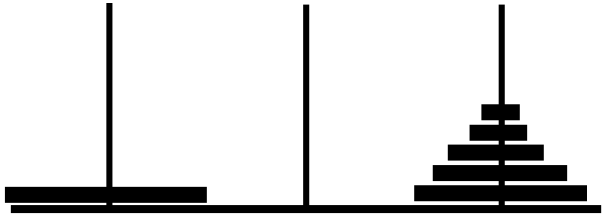
5 move 1 from C to A

بازگشتی: مثالهای بیشتر

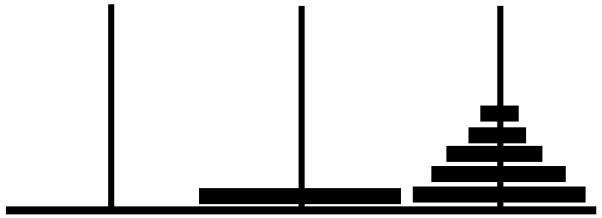
برجهای هانوی: راه حل بازگشتی



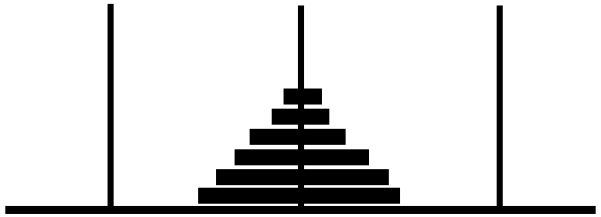
$N-1$ دیسک بالایی را به شکل بازگشتی به برج C منتقل کن



دیسک بزرگ (N) را از برج A به برج B منتقل کن



$N-1$ دیسک بالایی را به شکل بازگشتی به برج B منتقل کن



بازگشتی: مثالهای بیشتر

برجهای هانوی: تابع بازگشتی

```
1  def H(n, source, dest, temp):
2      if n == 1:
3          print("move %d from %s to %s"%(n,source,dest))
4      else:
5          H(n-1, source, temp, dest)
6          print("move %d from %s to %s"%(n,source,dest))
7          H(n-1, temp, dest, source)
8
9  H(3,"A","B","C")
```

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