جزوه ساختمان داده

مهدی صفریان

۱۱ آذر ۱۳۹۹

فهرست مطالب

از انجایی که درس ساختمان داده هیچ ویدیویی ندارد و جزوه اصلی درس صرفا شامل تمارین مدرس است، فلذا توضیحات درون این جزوه فقط برداشت شخصی بنده است.

۱ توابع بازگشتی

تابع بازگشتی تابعی است که حداقل شامل یک دستور باشد و درون خود، خودش را صدا بزند؛ توابع بازگشتی به تعداد محدود اجرا می‌شوند و پس از رسیدن به نتیجه مطلوب متوقف می‌شوند.

۱.۱ مثال فاکتوریل

فاکتوریل حاصلضرب اعداد n تا ۱ را حساب می‌کند. برای مثال فاکتوریل عدد ۳ برابر است با 1 * 2 * 3.

کد ۱: فاکتوریل

```
function fact(n) {
    if n==1                          #Fact function rules:
        return 1                     #fact(n) = { 1,              n=1.
                                                { n * fact(n-1),  n>1.
    else
        return n * fact(n-1)
}
```

۱.۱.۱ توضیحات

همانطور که گفته شد تابع فاکتوریل حاصل‌ضرب اعداد n تا ۱ را به دست می‌آورد. ۱

- ابتدا به تابع مقداری را به عنوان ورودی می‌دهیم.
- سپس اگر عدد ورودی تابع برابر ۱ بود، تابع متوقف شود.
- در غیر این صورت اگر عددی غیر از ۱ بخش دوم تابع اجرا می‌شود.

مثال: فاکتوریل عدد ۴ را محاسبه کنید.

$fact(4) = 4 * 3 * 2 * 1 = 24$

- $fact(4) = 4 * fact(3) => 4 * 6 = 24$
- $fact(3) = 3 * fact(2) => 3 * 2 = 6$
- $fact(2) = 2 * fact(2) => 2 * 1 = 2$

۲.۱ مجموع اعداد ۱ تا n

این تابع به سادگی همانطور که نامش مشخص است مجموع اعداد ۱ تا n را محاسبه می‌کند.

کد ۲: مجموع اعداد ۱ تا n

```
۱ def sum(n):
۲     if n == 1:                    #sum function rules:
۳         return 1                  #sum(n) = { 1,            n = 1.
۴     else:                                   { n + sum(n-1)  n > 1.
۵         print(n)
۶         return n + sum(n-1)
۷ print(sum(7))
```

مثال: مجموع اعداد ۱ تا ۳ را بیابید.

$sum(3) = 3 + 2 + 1 = 6$

- $sum(3) = 3 + sum(2) => 3 + 3 = 6$
- $sum(2) = 2 + sum(1) => 2 + 1 = 3$

۳.۱ توان

تابع توان مقدار n^m را حساب می‌کند(ورودی‌ها صحیح و مثبت هستند) به سادگی عمل توان رسانی را حساب می‌کند.

```
۱ function power(n, m){
۲     if m==1                       #power function rules:
۳         return n                  #power(n,m) = { n,            m = 1.
۴     else                                        { n * power(n,m-1)  m > 1.
۵         return n * power(n,m-1)
۶
۷     }
```

مثال: تابع توان را فراخوانی کنید و با مقدار دلخواه مثال بزنید.

$power(3, 4) => 3 * 3 * 3 * 3 = 81$

- $power(3,4) = 3 * power(3,3) = 3 * 3 * 3 * 3$
- $power(3,3) = 3 * power(3,2) = 3 * 3 * 3$
- $power(3,2) = 3 * power(3,1) = 3 * 3$

۴.۱ خارج قسمت تقسیم

تابع خارج قسمت تقسیم مقدار خارج قسمت تقسیم صحیح a بر b را محاسبه می‌کند. تاکید می‌کنم که هدف ما تقسیم صحیح است.

```
۱  def div(a,b):
۲      if a<b:                  #div function rules:
۳          return 0             #div(a,b) = { 0,                a < b.
۴      if a≥b:                              { div(a-b,b)+1,     a >= b.
۵          return div(a-b,b)+1
```

۱.۴.۱ توضیحات

- در مرحله اول بررسی می‌شود آیا عدد a بزرگتر از b است یا خیر.
  - اگر کوچک‌تر باشد مقدار صفر برمی‌گردد چرا که ما مقدار تقسیم صحیح را می‌خواهیم.
  - اگر بزرگتر یا مساوی باشد عدد a برابر می‌شود با a-b و در نهایت مقدار نهایی یه واحد به آن اضافه می‌شود.

مثال: با استفاده از تابع div مقادیر a = ۱۱ و b = ۳ را بررسی کنید.

$div(11,3) => 2 + 1 = 3$

- $div(11,3) = div(8,3) + 1 => 2 + 1 = 3$
- $div(8,3) = div(5,3) + 1 => 1 + 1 = 2$
- $div(5,3) = div(2,3) + 1 => 0 + 1 = 1$

۵.۱ تابع باقی مانده تقسیم

```
۱  def r(a,b):
۲      if a<b:
۳          return a
۴      else:
۵          return r(a-b,b)
۶  print(r(11,3))
```

۶.۱ هانوی

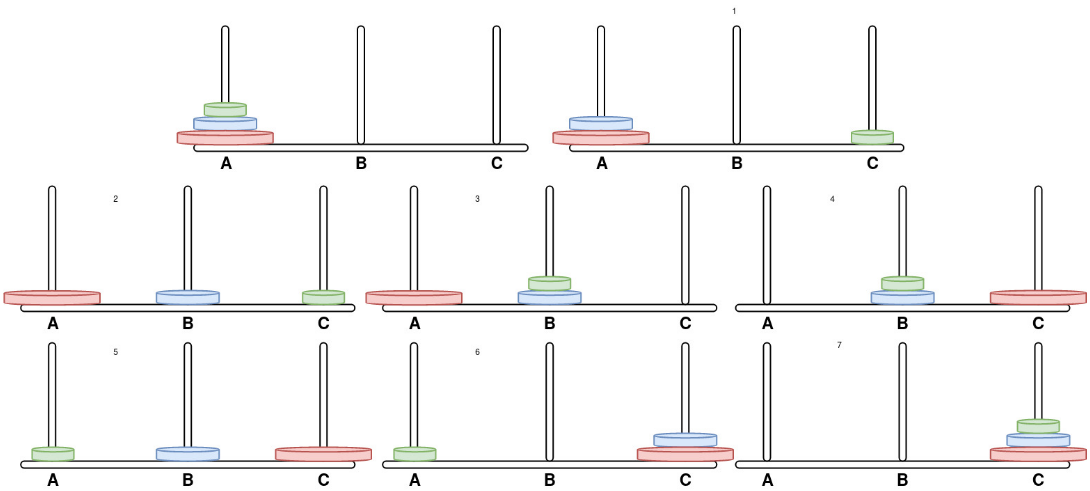
در مسئله برج هانوی ما می‌خواهیم تعداد n مهره را از میله A به C ببریم و میله B به عنوان میله کمکی استفاده کنیم.

```
۱  def tower(n, A, B, C):
۲      if n==1:
۳          return A to C
۴      else:
۵          tower(n-1, A, B, C)
۶          A to C
۷          tower(n-1, B, A, C)
```

۱.۶.۱ توضیحات

- ابتدا n-۱ مهره را به میله B انتقال می‌دهیم و در نهایت بزرگترین مهره در A باقی می‌ماند.
- بعد مهره بزرگ را به میله C منتقل می‌کنیم.
- در آخر مانند مرحله اول مهره‌ها را از B به C انقال می‌دهیم.

یادتان باشد که برای انتقال مهره‌ها باید از میله‌های کمکی استفاده کنیم.



۷.۱ فیبوناچی

تابع فیبوناچی جمله امn اعداد فیبوناچی را به دست می‌اورد.

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

#Fibonacci function rules:

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