



به نام خدا

# تمرین عملی دوم ساختمان داده‌ها و الگوریتم

درخت و کاربردهایش

استاد:

دکتر رستمی

استادیار:

عرشیا عموزاد

# فحوهی ارسال و ارائه‌ی پاسخ تمرین

## مهلت و نوع فایل‌های ارسالی:

- پاسخنامه باید شامل فایل‌های اجرایی (py) و فایل PDF شرح کدها باشد.
- مهلت ارسال: ۱۴۰۴/۰۹/۳۰ تا ساعت ۲۳:۵۹
- پاسخ‌های خود را به آیدی تلگرام Arshi82\_94 بفرستید.
- روز ارائه‌ی تمرین عملی بعد از هماهنگی در گروه تلگرامی درس اعلام خواهد شد.

## ملاحظات پیاده‌سازی:

- از پیاده‌سازی آماده کتابخانه‌ها **جلوگیری** شود.
- در پیاده‌سازی های خود از داده ساختار های آموخته شده(لیست پیوندی، صف، پشته و...) استفاده کنید.
- فایل‌های template کلاس‌ها در لینک گیت هاب زیر موجود است.(به نوع خروجی‌های توابع دقت شود)

موفق باشید

## ۱. مسئله ۱: مجموع اعداد باینری مسیرهای ریشه به برگ

یک درخت دودویی داده شده که هر گره مقدار ۰ یا ۱ دارد. هر مسیر از ریشه به برگ نشان‌دهنده یک عدد باینری است (با ارزش ترین رقم مربوط به ریشه است). باید مجموع اعداد باینری تمام مسیرهای ریشه به برگ را محاسبه کنید.

تابع `sumRootToLeaf` را پیاده‌سازی کنید:

`sumRootToLeaf(root)`.

ورودی: ریشه درخت دودویی (از نوع `TreeNode` یا `None` یا

خروجی: عدد صحیح (مجموع اعداد باینری همه مسیرهای ریشه به برگ)

**فرمت ورودی:**

ورودی به صورت یک شیء از `TreeNode` به شما داده می‌شود. ساختار کلاس `TreeNode` به صورت زیر است:

```
1 class TreeNode:
2     def __init__(self, val=0, left=None, right=None):
3         self.val = val      #      (0    1)
4         self.left = left    #
5         self.right = right   #
```

مثال ۱:

```
1 #      1
2 #      / \
3 #      0   1
4 #      / \ / \
5 #      0   1 0  1
6
7 root = TreeNode(
8     1,
9     TreeNode(0, TreeNode(0), TreeNode(1)),
10    TreeNode(1, TreeNode(0), TreeNode(1))
11 )
12
13 #       :
14 sumRootToLeaf(root)  # ==> 22
```

**توضیح مثال ۱:**

چهار مسیر از ریشه به برگ وجود دارد:

• مسیر ۱:  $1 \rightarrow 0 \rightarrow 0 = 100$  (در مبنای ۱۰)

• مسیر ۲:  $1 \rightarrow 0 \rightarrow 1 = 101$

• مسیر ۳:  $1 \rightarrow 1 \rightarrow 0 = 110$

• مسیر ۴:  $1 \rightarrow 1 \rightarrow 1 = 111$

$$22 = 7 + 6 + 5 + 4$$

مثال ۲:

```
1 root = TreeNode(0)
2
3 sumRootToLeaf(root)  # ==> 0
```

**توضیح مثال ۲:**

یک مسیر داریم:  $0 = 000$  (در مبنای ۱۰)

مثال ۳:

```
1 #      1
2 #      / \
3 #      1   0
4
5 root = TreeNode(1, TreeNode(1), TreeNode(0))
6
7 sumRootToLeaf(root)  # ==> 5
```

**توضیح مثال ۳:**  
دو مسیر از ریشه به برگ وجود دارد:

- مسیر ۱:  $1 \rightarrow 1 =$  عدد باینری  $11 = 3$
- مسیر ۲:  $0 \rightarrow 1 =$  عدد باینری  $10 = 2$

$$\text{مجموع} = 2 + 3 = 5$$

**نکات قابل توجه:**

- تعداد گره‌های درخت بین ۱ تا ۱۰۰۰ است.
- مقدار هر گره ۰ یا ۱ است.
- پاسخ تضمین شده است که در محدوده عدد ۳۲ بیتی علامت‌دار قرار می‌گیرد.
- پشته (Stack) برای پیمایش درخت می‌توانید استفاده کنید.
- در صورت مواجهه با درخت خالی (None)، مقدار ۰ را برگردانید.

## ۲. مسئله ۲: یافتن بزرگ‌ترین مقدار در هر سطر درخت

یک درخت دودویی داده شده است. باید بزرگ‌ترین مقدار موجود در هر سطر (سطح) درخت را پیدا کرده و به صورت یک لیست بازگردانید. سطراها از ریشه (سطر ۰) شروع می‌شوند.

تابع `largestValues` را پیاده‌سازی کنید:

**۱. `largestValues(root)`:**

ورودی: ریشه درخت دودویی (از نوع `TreeNode` یا `None`)  
خروجی: لیستی از اعداد صحیح (بزرگ‌ترین مقدار هر سطر)

**فرمت ورودی:**

ورودی به صورت یک شیء از کلاس `TreeNode` به شما داده می‌شود. ساختار کلاس `TreeNode` به صورت زیر است:

```
1 class TreeNode:  
2     def __init__(self, val=0, left=None, right=None):  
3         self.val = val      #  
4         self.left = left    #  
5         self.right = right   #
```

**مثال ۱:**

```
1      #      1  
2      #      / \  
3      #      3   2  
4      #      / \   \  
5      #      5   3   9  
6  
7 root = TreeNode(1,  
8                 TreeNode(3, TreeNode(5), TreeNode(3)),  
9                 TreeNode(2, None, TreeNode(9)))  
10  
11 largestValues(root)  # ==> [1, 3, 9]
```

**توضیح مثال ۱:**

- سطر ۰ (ریشه):  $[1] \leftarrow$  بزرگ‌ترین: ۱
- سطر ۱:  $[3, 2] \leftarrow$  بزرگ‌ترین: ۳
- سطر ۲:  $[5, 3, 9] \leftarrow$  بزرگ‌ترین: ۹

**مثال ۲:**

```
1 #      1
2 #      / \
3 #      2   3
4
5 root = TreeNode(1, TreeNode(2), TreeNode(3))
6
7 largestValues(root)  # ==> [1, 3]
```

توضیح مثال ۲:

- سطر ۰:  $[1] \leftarrow$  بزرگترین: ۱
- سطر ۱:  $[2, 3] \leftarrow$  بزرگترین: ۳

مثال ۳:

```
1 #      7
2
3 root = TreeNode(7)
4
5 largestValues(root)  # ==> [7]
```

مثال ۴:

```
1 root = None
2
3 largestValues(root)  # ==> []
```

مثال ۵:

```
1 #      -1
2 #      / \
3 #      -2   -3
4 #      /   / \
5 #      -5   -4   -6
6
7 root = TreeNode(-1,
8                 TreeNode(-2, TreeNode(-5)),
9                 TreeNode(-3, TreeNode(-4), TreeNode(-6)))
10
11 largestValues(root)  # ==> [-1, -2, -4]
```

توضیح مثال ۵:

- سطر ۰:  $[-1] \leftarrow$  بزرگترین: -۱
- سطر ۱:  $[-2, -3] \leftarrow$  بزرگترین: -۲
- سطر ۲:  $[-5, -4, -6] \leftarrow$  بزرگترین: -۴

نکات قابل توجه:

- تعداد گره‌های درخت در محدوده  $[0, 10^4]$  است.
- می‌توانید از صف برای نگهداری گره‌های هر سطح استفاده نمایید.
- در صورت مواجهه با درخت خالی (None)، لیست خالی برگردانید.