انجمن علمی مهندسی کامپیوتر
دانشگاه مازندران

به نام خدا

تمرین عملی دوم ساختمان داده‌ها و الگوریتم

درخت و کاربردهایش

استاد:

دکتر روستایی

استادیار:

عرشیا عموزاد

۱۶ آذر ۱۴۰۴

نحوه‌ی ارسال و ارائه‌ی پاسخ تمرین

مهلت و نوع فایل های ارسالی:

- پاسخنامه باید شامل فایل های 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.
- فایل های template کلاس ها در لینک گیت هاب زیر موجود است.(به نوع خروجی های توابع دقت شود)
- لینک گیت‌هاب

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 ارزش‌ترین رقم مربوط به ریشه است). باید مجموع اعداد باینری تمام مسیرهای ریشه به برگ را محاسبه کنید.

تابع **sumRootToLeaf** را پیاده‌سازی کنید:

۱. **sumRootToLeaf(root)**:
ورودی: ریشه درخت دودویی (از نوع TreeNode یا None)
خروجی: عدد صحیح (مجموع اعداد باینری همه مسیرهای ریشه به برگ)

فرمت ورودی:

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

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

مثال ۱:

```
#        1
#       / \
#      0   1
#     / \ / \
#    0  1 0  1

root = TreeNode(
    1,
    TreeNode(0, TreeNode(0), TreeNode(1)),
    TreeNode(1, TreeNode(0), TreeNode(1))
)

#             :
sumRootToLeaf)root)  # ==> 22
```

توضیح مثال ۱:

چهار مسیر از ریشه به برگ وجود دارد:

- مسیر ۱: 0 → 0 → 1 = عدد باینری 100 = ۴ (در مبنای ۱۰)
- مسیر ۲: 1 → 0 → 1 = عدد باینری 101 = ۵
- مسیر ۳: 0 → 1 → 1 = عدد باینری 110 = ۶
- مسیر ۴: 1 → 1 → 1 = عدد باینری 111 = ۷

مجموع = ۴ + ۵ + ۶ + ۷ = ۲۲

مثال ۲:

```
root = TreeNode(0)

sumRootToLeaf(root)  # ==> 0
```

توضیح مثال ۲:

یک مسیر داریم: 0 = عدد باینری 0 = ۰

مثال ۳:

```
#        1
#       / \
#      1   0

root = TreeNode(1, TreeNode(1), TreeNode(0))

sumRootToLeaf(root)  # ==> 5
```

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

- مسیر ۱: 1 → 1 = عدد باینری 11 = ۳
- مسیر ۲: 0 → 1 = عدد باینری 10 = ۲

مجموع = ۳ + ۲ = ۵

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. سطرها 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:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val      #
        self.left = left    #
        self.right = right  #
```

مثال ۱:

```
#        1
#       / \
#      3   2
#     / \   \
#    5   3   9

root = TreeNode(1,
                TreeNode(3, TreeNode(5), TreeNode(3)),
                TreeNode(2, None, TreeNode(9)))

largestValues(root)  # ==> [1, 3, 9]
```

توضیح مثال ۱:

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

مثال ۲:

```
#        1
#       / \
#      2   3

root = TreeNode(1, TreeNode(2), TreeNode(3))

largestValues(root)  # ==> [1, 3]
```

توضیح مثال ۲:

- سطر ۰: [1] ← بزرگترین: ۱
- سطر ۱: [2, 3] ← بزرگترین: ۳

مثال ۳:

```
#        7

root = TreeNode(7)

largestValues(root)  # ==> [7]
```

مثال ٤:

```
root = None

largestValues(root)  # ==> []
```

مثال ٥:

```
#        -1
#        / \
#      -2  -3
#      /   / \
#    -5  -4  -6

root = TreeNode(-1,
                TreeNode(-2, TreeNode(-5)),
                TreeNode(-3, TreeNode(-4), TreeNode(-6)))

largestValues(root)  # ==> [-1, -2, -4]
```

توضیح مثال ٥:

- سطر ۰: [-1] ← بزرگترین: -۱
- سطر ۱: [-2, -3] ← بزرگترین: -۲
- سطر ۲: [-5, -4, -6] ← 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.