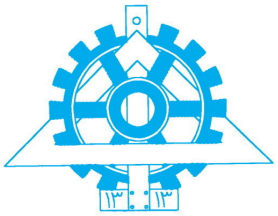


تمرین کامپیوتری شماره ۳

دانشکده مهندسی برق و کامپیوتر

ساختمان داده - بهار ۱۴۰۳

مدرس: **دکتر هشام فیلی**

مهلت تحویل: ۱۴۰۲/۰۳/۰۶ (۱۲ شب)

طراحان تمرین: شایان کاشفی، میثاق محقق

مقدمه

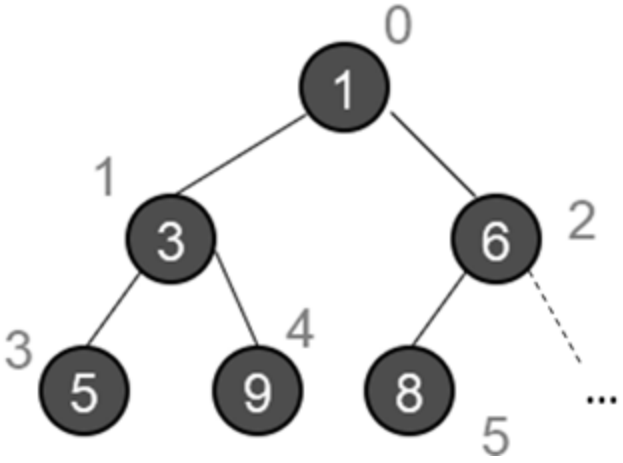
این تمرین کامپیوتری برای آشنایی با مباحث مربوط به درخت و داده ساختارهای هیپ می‌باشد. در قسمت اول به شما یک قالب از سه داده ساختار داده می‌شود و انتظار می‌رود که با توجه به مطالب گفته‌شده در رابطه با هر تابع، آنها را کامل کنید.

مسئله‌ی اول: دستگرمی (۲۵ نمره)

- محدودیت زمان ۱ ثانیه
- محدودیت حافظه ۲۵۶ مگابایت
- طراح: میثاق محقق

توضیح داده ساختارها:

داده ساختار min-heap: برای هر عنصری که اضافه می‌شود یک value و یک index وجود دارد که برای مثال در شکل زیر مقادیر داخل نودها، value و مقادیر بیرون آنها، index آنها هستند.



داده ساختار huffman-tree: در این قسمت به دو شکل ورودی می‌دهیم. اولی آنکه لیست کارکتر ها و تعداد تکرارشان را ست می‌کنیم. دومین روش این است که یک متن می‌دهیم. بعد از هر کدام از این دو روش، درخت مربوط به ورودی را تشکیل می‌دهیم.

داده ساختار bst: تعدادی عنصر را به آن اضافه شده و سپس تعدادی عملیات روی درخت دودویی انجام می‌شود.

توضیح ارورها:

در هر تابع، حالت هایی وجود دارد که موجب رخ دادن ارور می‌شود (مانند پاپ کردن از هیپ خالی). در صورت رخ دادن آنها، صرفا آنها را به صورت زیر هندل کنید:

```
raise Exception('error_text')
```

تمام این ارور ها عبارت اند از (بقیه ارور ها بررسی نمی‌شوند):

```
raise Exception('invalid index') -> ایندکس وارد شده عدد نباشد یا تایپ آن درست نباشد

raise Exception('out of range index') -> ایندکس وارد شده در محدوده سایز نباشد

raise Exception('empty') -> از هیپ یا درخت خالی مقداری خارج شود
```

توضیح توابع:

```
class MinHeap:
    class Node:
        pass

    def __init__(self): -> کانستراکتور
        pass

    def bubble_up(self, index): -> نود داده‌شده (مشخص شده با ایندکس) را تا جای ممکن به بالای هیپ می‌برد
        pass

    def bubble_down(self, index): -> نود داده‌شده (مشخص شده با ایندکس) را تا جای ممکن به پایین هیپ می‌برد
        pass

    def heap_push(self, value): -> عنصر جدید وارد هیپ می‌شود
        pass

    def heap_pop(self): -> * روت را خارج می‌کند و مقدارش را ریترن می‌کند
        pass
```

```
    def find_min_child(self, index): -> * ایندکس کوچکترین فرزند نود داده شده را برمی‌گرداند
        pass

    def heapify(self, *args): -> تعدادی آرگومان دریافت کرده و آنها را وارد هیپ می‌کند
        pass

class HuffmanTree:
    class Node:
        pass

    def __init__(self): -> کانستراکتور
        pass

    def set_letters(self,*args): -> آرگومان های دریافتی را به‌عنوان حروف ست می‌کند
        pass

    def set_repetitions(self,*args): -> آرگومان های دریافتی را به‌عنوان تعداد تکرار حروف ست می‌کند
        pass

    def build_huffman_tree(self): -> درخت هافمن مربوطه را می‌سازد
        pass

    def get_huffman_code_cost(self): -> * هزینه انکودینگ هافمن متن داده‌شده را برمی‌گرداند
        pass

    def text_encoding(self, text): -> از روی متن داده شده کد هافمن را می‌سازد
        pass
```

```
class Bst:
    class Node:
        Pass

    def __init__(self): -> کانستراکتور
        pass

    def insert(self, key): -> عنصر جدید را وارد درخت می‌کند
        pass

    def inorder(self): -> * درخت را به ترتیب میانوندی پیمایش و برمی‌گرداند
        pass
```

نکته: توابعی که مقداری را ریترن می‌کنند با * مشخص شده اند.

توضیح در مورد قالب

قالب شامل چند کلاس و تابع می باشد که کافی است توابع مشخص شده در بالا را کامل کنید و نیازی به یادگیری مابقی قالب نیست. در صورت صلاحدید می‌توانید متدهای دلخواه را به داده ساختارها اضافه کنید.

ورودی

با توجه به قالب داده شده ابتدا یک یا چند آبجکت از نوع پشته یا صف یا لینکد لیست ایجاد می‌شود. سپس توابع مشخص شده برای هر کدام صدا زده می‌شوند که همگی در قالب آمده است و توضیح مربوط به هرکدام در pdf تمرین آمده است.

نمونه‌ی ورودی و خروجی 1

```
INPUT:
make min_heap m1
call m1.heapify(10,5,30,50)
call m1.find_min_child(0)
call m1.heap_pop()
```

```
call m1.heap_pop()
call m1.heap_pop()
call m1.heap_pop()
call m1.find_min_child(-1)
call m1.find_min_child(1)
call m1.find_min_child('salap')
```

```
OUTPUT:
1
5
10
30
50
out of range index
out of range index
invalid index
```

نمونه‌ی ورودی و خروجی 2

```
INPUT:
make bst b1
call b1.insert(50)
call b1.insert(15)
call b1.insert(20)
call b1.insert(10)
call b1.insert(40)
call b1.insert(60)
call b1.inorder()
```

```
OUTPUT:
10 15 20 40 50 60
```

نمونه‌ی ورودی و خروجی 3

```
INPUT:
make huffman_tree h1
make huffman_tree h2
call h1.set_letters('a','b','c','d','e','f')
call h1.set_repetitions(1,3,12,13,16,1000)
call h1.build_huffman_tree()
call h1.get_huffman_code_cost()
call
h2.text_encoding('chaii-migholam-garm-sham-va-sard-va-tondkhoo-nabasham')
call h2.get_huffman_code_cost()
```

```
OUTPUT:
1139
198
```

مسئله‌ی دوم: مهمونی

- محدودیت زمان: ۴ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- طراح: شایان کاشفی شایسته

در سری جدید مهمونی تعداد مهمان‌های آقای مجری بشدت افزایش یافته و آقای مجری مجبور است هر شب تعداد زیادی رخت‌خواب در اتاق بیندازد. به دلیل خستگی میزبانی از تعداد بالای مهمان‌ها، آقای مجری می خواهد هر شب در دورترین فاصله ممکن از بچه فامیل بخوابد تا خواب آرامی داشته باشد. او برای این کار از شما کمک خواسته است.

اگر اتاق را به صورت صفحه مختصات ببینیم، آقای مجری هر بار یکی از اعمال زیر را انجام میدهد:

1. یک رخت‌خواب در مختصات (x, y) می‌اندازد
2. رخت‌خواب پهن شده در مرحله i ام را جمع میکند
3. فاصله دورترین رخت‌خواب از نقطه (x, y) را از شما میپرسد (فاصله را منهتنی در نظر بگیرید)

ورودی

در خط اول ورودی یک عدد q که نشان‌دهنده تعداد اعمال آقای مجری است آمده است.

$$(1 \leq q \leq 500000)$$

در هر کدام از q خط بعدی یک عمل از 3 نوع عمل تعریف شده آمده است

نوع۱: "+ X Y" انداختن رخت‌خواب در نقطه (X, Y) اتاق

نوع۲: "- N" جمع کردن رخت‌خوابی که در عمل N ام پهن شده است

نوع۳: "? X Y" پرسیدن دورترین رخت‌خواب از نقطه (X, Y)

$$(1 \leq X, Y \leq 10^9)$$

خروجی

خروجی برنامه‌ی شما باید به ازای هر عمل نوع۳ یک عدد به عنوان جواب آن بخش چاپ کند.

نمونه‌ی ورودی و خروجی

```
INPUT:
10
+ 8 1
- 1
+ 3 9
? 8 4
? 2 2
? 4 8
+ 4 7
? 5 10
? 0 1
- 2
```

```
OUTPUT:
10
8
2
4
11
```

مسئله‌ی سوم: مسیریابی در کشور درخت‌ها

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- طراح: شایان کاشفی شایسته

عید نوروز به کشور درخت‌ها هم رسیده و مردم پایتخت این کشور قصد دارند برای تعطیلات به دیگر شهرهای این کشور سفر کنند. ساختار شهرها و جاده‌های کشور درخت‌ها به صورت یک درخت دودویی است که ریشه آن، پایتخت است.

برای اینکه مردم پایتخت در مسیریابی سفرها دچار سردرگمی نشوند، وزارت راه به سراغ ایده جالبی رفته است. وزارت راه در هر شهر یک تابلو نصب‌کرده است و روی آن یک عدد نوشته است و برای مسیریابی به مسافران گفته است که اگر عدد شهر مقصدتان کوچکتر از عدد تابلوی شهر فعلی بود، به شهر سمت چپ در درخت دودویی شهرها سفر کنید و اگر عدد شهر مقصدتان بزرگتر از عدد تابلوی شهر فعلی بود، به شهر سمت راست در درخت دودویی سفر کنید تا اینکه به مقصد برسید.

اما به دلیل اینکه تعطیلات نزدیک است و ماموران وزارت نیز میخواهند به سفر بروند، آنها وظیفه خود را به درستی انجام نداده و تابلوها را به درستی نصب نکرده‌اند. حال وزیر راه از شما میخواهد که به او کمک کنید که محاسبه کند مسیریابی به چند شهر دارای مشکل است.

ورودی

در خط اول ورودی یک عدد n که نشان‌دهنده تعداد شهرهای کشور است آمده است

$$(1 \leq n \leq 10^5)$$

در هر کدام از n خط بعدی سه عدد v, l, r آمده است که به ترتیب نشان دهنده عدد تابلوی روی آن شهر، ایندکس شهر سمت چپ در درخت و ایندکس شهر راست در درخت می‌باشد. -1 نشان‌دهنده این است که به شهری در آن قسمت متصل نیست

$$(0 \leq v, l, r \leq 10^9)$$

خروجی

تعداد شهرهایی که مسیریابی‌شان دارای مشکل است را چاپ کنید

نمونه‌ی ورودی و خروجی

```
INPUT:
3
15 -1 -1
10 1 3
5 -1 -1
```

```
OUTPUT:
2
```

```
INPUT:
8
6 2 3
3 4 5
12 6 7
1 -1 8
4 -1 -1
5 -1 -1
14 -1 -1
2 -1 -1
```

```
OUTPUT:
1
```

نکات تکمیلی

- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.
- استفاده از کدهای آماده برای پیاده‌سازی این مباحث (جستجو شده در اینترنت و ...)، مجاز نمی‌باشد. در صورت کشف، مانند تقلب برخورد می‌شود.
- به جز سوال یک، در تمامی سوالات می‌توانید از کتابخانه‌های پیش‌فرض پایتون استفاده نمایید.
- در صورتی که تست‌های تمامی سوالات پاس شوند و نمره آنها کامل شود، ۱۰ نمره امتیازی اعمال می‌شود (نمره ۱۰۰، ۱۱۰ خواهدشد).