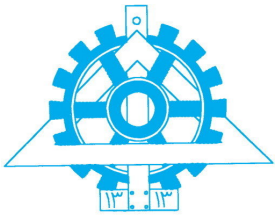


تمرین کامپیوتری شماره ۳

دانشکده مهندسی برق و کامپیوتر

ساختمان داده - بهار ۱۴۰۲

مدرس: **دکتر هشام فیلی**

مهلت تحویل: ۱۴۰۲/۰۲/۱۷ (۱۲ شب)

طراحان تمرین: ارشیا عطایی نایینی، علی عطاءاللهی

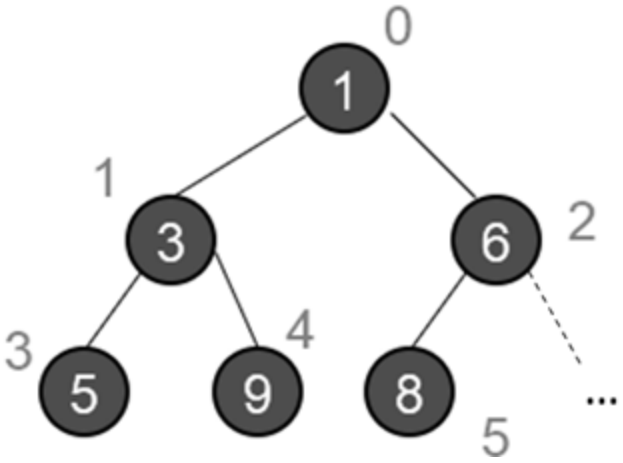
مقدمه

این تمرین کامپیوتری برای آشنایی با مباحث مربوط به درخت و داده ساختار های هیپ می‌باشد. در قسمت اول به شما یک قالب از سه داده ساختار داده می شود و انتظار می رود که با توجه به مطالب گفته شده در رابطه با هر تابع، آنها را کامل کنید.

مسئله‌ی اول: پیاده سازی (۲۵ نمره)

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 نود ها 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 اررورها :

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:

```
@for_all_methods(fix_str_arg)
@for_all_methods(print_raised_exception)
class MinHeap:
    def __init__(self): -> کانستراکتور
        pass

    class Node:
        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:
    def __init__(self): -> کانستراکتور
        pass

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

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

    class Node:
        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

    @fix_str_arg
```

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


@for_all_methods(fix_str_arg)
@for_all_methods(print_raised_exception)
class Bst():
    def __init__(self): -> کانستراکتور
        pass

    class Node:
        Pass

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

    def inorder(self, key): -> 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: میانه (۲۵ نمره)

کیلیان به تازگی یاد گرفته‌است که می‌تواند با مین‌هیپ همواره مقدار مینیمم آرایه‌ای که در حال تغییر است (حذف یک عنصر و اضافه کردن یک عنصر) را بدست آورد. او که دوست دارد خود را به چالش بکشد اکنون می‌خواهد در آرایه ای که همواره در حال تغییر است میانه‌ی اعضای آرایه را بدست آورد. (اگر طول یک آرایه 2k باشد، میانه میانگین عضو k بعد از مرتب‌سازی ایست و اگر طول آن 2k + 1 باشد، میانه برابر عضو k + 1 است. به عنوان مثال میانه آرایه [2, 3, 7,1] برابر 2 و میانه آرایه [4, 3,6,7,1] برابر 4 است.) آیا می‌توانید به او کمک کنید؟

ورودی

در خط اول سه عدد n و q داده می‌شود. در خط دوم n عدد ورودی داده می‌شود که حالت اولیه ی آرایه است. در q بعدی هر خط، عددی جدید به آرایه اضافه می‌شود.

خروجی

به ازای هر n ورودی، میانه آرایه پس از اضافه‌شدن آن را خروجی دهید.

نمونه‌ی ورودی و خروجی ۱

Input:

```
7 3
1 3 4 7 5 8 8
4
5
1
```

Output:

```
4
5
4
```

توضیح :

در ابتدا بعد از اولین ورودی، مرتب‌شده‌ی آرایه برابر [8, 8, 7, 5, 4, 4, 3, 1] است که میانه عضو چهارم یا همان 5 است. در مرحله بعدی آرایه برابر [8, 8, 7, 5, 5, 4, 4, 3, 1] است که میانه عضو پنجم یا همان 5 است. در آخر آرایه پس از اضافه‌شدن 1 برابر [8, 8, 7, 5, 5, 4, 4, 3, 1, 1] است که میانه برابر عنصر پنجم یا همان 4 است.

نمونه‌ی ورودی و خروجی ۲

Input:

3 3
1 2 3
4
10
20

Output:

2

3

3

مسئله‌ی سوم: رشک (۲۵ نمره)

مهدی جان در کار کردن با اعداد بسیار حاذق است. همین امر باعث شد تا پارسا جان به او حسادت کند و سعی در ضایع‌کردن او داشته‌باشد. مهدی جان یک مجموعه(می‌تواند شامل عضو تکراری باشد) تهی دارد و پارسا جان در هر مرحله یا عدد x را به این مجموعه اضافه می‌کند یا یک عدد x را از آن حذف می‌کند (در صورتی که عدد x در این مجموعه وجود نداشت اتفاقی رخ نمی‌دهد). مهدی جان بعد از هر پرسش باید بگوید که Mex این مجموعه چیست. Mex یک مجموعه از اعداد برابر اولین عدد حسابی است که در این مجموعه نیامده باشد. به عنوان مثال Mex مجموعه (0, 2, 4) برابر 1 است. به مهدی جان کمک کنید تا ضایع نشود.

ورودی

خط اول ورودی شامل عدد n است که تعداد عملیات‌ها است.

در n خط بعدی در هر خط یک عملیات آمده است که شامل:

x +: عدد x را به مجموعه اضافه می‌کند (در صورت وجود x در مجموعه، تعدادشان بیشتر می‌شود)

x -: عدد x را از مجموعه حذف می‌کند (در صورت عدم وجود x در مجموعه، اتفاقی رخ نمی‌دهد)

خروجی

در n سطر Mex مجموعه را پس از هر عملیات چاپ کنید.

نمونه‌ی ورودی و خروجی ۱

Input:

```
6
+ 1
+ 0
+ 0
- 0
+ 2
- 1
```

Output:

0

2

2

2

3

1

مسئله‌ی چهارم: نکرومورف‌کشون (۲۵ نمره)

آیزاک کلارک در حال مبارزه با نکرومورف‌های سگ‌جون در اسپرال است. او در یک اتاق گیر افتاده‌است که خودش در ضلع شمالی اتاق است و نکرومورف‌ها در چند لاین مختلف به او از سمت جنوب هجوم می‌آورند. ضلع غربی و شرقی دیوار است و آیزاک خیالش از آنها راحت است.

متاسفانه تیر پلاسما کاتر، مهم‌ترین سلاح آیزاک تمام شده و او باید با استفاده از چند غلتک پلاسمایی کار نکرومورف‌ها را یکسره کند. تعداد لاین‌هایی که نکرومورف‌ها از آن هجوم می‌آورند ۱ تا N است. همچنین از هر لاین یک نکرومورف با جون Z_i به ما حمله می‌کند. آیزاک M غلتک پلاسمایی دارد که هر کدام K_i بار می‌توانند شلیک شوند و در برخورد با نکرومورف‌ها یک جون از آنها کم می‌کند. هر شلیک غلتک هم به لاین R_i تا L_i برخورد می‌کند. اگر جون یک نکرومورف به صفر برسد، نکرومورف می‌میرد و پودر می‌شود.

حال باید ببینیم که آیا آیزاک کلارک می‌تواند تمام نکرومورف ها را بکشد یا نه. اگر می‌تواند با حداقل چند بار شلیک می‌تواند. چون باید در مصرف منابع صرفه جویی کند تا بتواند اهالی کلیسای یونیتالژی را شکست دهد.

ورودی

در خط اول عدد T می‌آید که نشان می‌دهد چند بار این موقعیت را بررسی می‌کنیم (تعداد تست‌کیس ها).

در خط اول هر تست N و M می‌آیند.

در خط بعدی تست اعداد $Z_1, Z_2, \ldots, Z_N$ می‌آیند.

در N خط بعدی تست نیز به ترتیب L_i , R_i , K_i می‌آیند.

خروجی

به ازای هر تست یک خط را خروجی می‌دهیم و در آن می‌گوییم که آیا آیزاک می‌تواند زنده بماند یا خیر. اگر جواب خیر بود Make-us-whole را چاپ می‌کنیم و اگر جواب بله بود، Isak-is-alive را چاپ کرده و سپس حداقل تعداد شلیک غلتک مورد نیاز را چاپ می‌کنیم.

نمونه‌ی ورودی و خروجی ۱

Input:

```
2
5 5
1 2 3 4 5
1 2 3
1 5 1
2 4 3
3 5 3
4 5 3
3 2
2 5 2
1 2 2
2 3 2
```

Output:

```
Isak-is-alive 6

Make-us-whole
```

توضیح تست:

در تست اول آیزاک می‌تواند با 6 بار شلیک غلتک همه نکرومورف‌ها را بکشد. غلتک 2و 3و 4 را یک بار و غلتک 5 را 3بار بیاندازد. البته راه‌حل‌های دیگری هم وجود دارد ولی کمترین عدد بدست آمده 6 خواهد بود.

برای تست دوم هیچ راه‌حلی وجود ندارد و آیزاک می‌میرد.

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.
- در تمامی سوالات به جز مواردی که در ادامه گفته می‌شود **نباید** از کتابخانه های آماده استفاده شود.
  - در سوال اول از کتابخانه sys و functools استفاده شده که برای آپلود استفاده از آن مشکلی ندارد.
  - در سوال ۲ و ۳ و ۴ اجازه استفاده از کتابخانه heapq را دارید.

  در صورتی که از کتابخانه ها غیر از موارد گفته شده استفاده شود، نمره مربوط به سوال را از دست خواهید داد.
- در صورتی که تست‌های تمامی سوالات پاس بشوند و نمره آنها کامل شود، ۱۰ نمره امتیازی اعمال می‌شود (نمره ۱۰۰ ، ۱۱۰ خواهدشد).