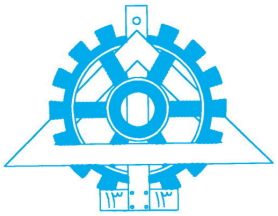


تمرین کامپیوتری شماره ۳

دانشکده مهندسی برق و کامپیوتر

ساختمان داده - پاییز ۱۴۰۱

مدرس: **دکتر هشام فیلی**

مهلت تحویل: ۱۴۰۱/۰۹/۲۳ (۱۲ شب)

طراحان تمرین: آریان سلطانی ، علی عطاءاللهی

مقدمه

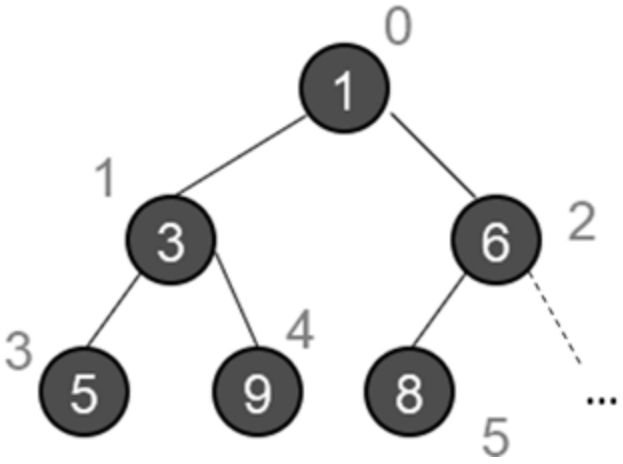
این تمرین کامپیوتری برای آشنایی با مباحث مربوط به درخت و داده ساختار های هیپ می‌باشد. در قسمت اول به شما یک قالب از سه داده ساختار red black tree ، huffman tree ، min heap داده می شود و انتظار می رود که با توجه به مطالب گفته شده در رابطه با هر تابع، آنها را کامل کنید.

مسئله‌ی اول: پیاده سازی (۲۵ نمره)

- محدودیت زمان ۱ ثانیه
- محدودیت حافظه ۲۵۶ مگابایت

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 red-black-tree : تعدادی عنصر را به آن اضافه شده و سپس تعدادی عملیات روی درخت 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 RedBlackTree():
    def __init__(self): -> کانستراکتور
        pass

    class Node():
        pass

    def find_node_color(self, value): ->  * رنگ نود که مقدار آن داده‌شده را به دو صورت
                                                "RED" یا "BLACK" برمی‌گرداند
        pass


    def left_rotate(self, node): -> دوران چپ‌گرد نود داده شده
        pass

    def right_rotate(self, node): -> دوران راست‌گرد نود داده شده
        pass

    def fix_insert(self, node): -> برای اصلاح درخت بعد از اضافه کردن عنصر جدید
        pass
```

```
    def insert(self, key): -> عنصر جدید را وارد درخت می‌کند
        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 red_black_tree rb1
call rb1.insert(100)
call rb1.insert(40)
call rb1.insert(300)
call rb1.insert(20)
call rb1.insert(75)
call rb1.insert(57)
call rb1.find_node_color(40)
call rb1.find_node_color(300)
call rb1.find_node_color(20)
call rb1.find_node_color(57)
call rb1.find_node_color(100)
call rb1.find_node_color(75)
```

Output:

```
RED
BLACK
BLACK
RED
BLACK
BLACK
```

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: kامین (۲۵ نمره)

- محدودیت زمان ۴ ثانیه
- محدودیت حافظه ۲۵۶ مگابایت

آلایا به تازگی یاد گرفته است که می تواند با مین هیپ همواره مقدار مینیمم آرایه ای که همواره 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 k امین کوچکترین عنصر آرایه را بدست آورد. آیا می توانید به او کمک کنید؟

ورودی

در خط اول سه عدد n و q و k داده می شود. در خط دوم n عدد ورودی داده می شود که حالت اولیه ی آرایه است. در q بعدی هر خط یک عملیات رو آرایه داده می شود.

عملیات:

+ x : عددی با مقدار x به آرایه اضافه می شود.

\- : عنصری با مقدار k امین کوچکترین عنصر آرایه(در صورتی که آرایه حداقل k عنصر دارد) حذف می شود.

print: مقدار k امین کوچکترین عنصر آرایه را خروجی دهید.

خروجی

پس از هر print خروجی مورد نظر را چاپ کنید. در صورتی که تعداد عناصر آرایه کمتر از k بود 1- چاپ کنید.

محدودیت‌ها

$$1 \leq n,\ q,\ k\ \leq\ 10^5$$
$$1 \leq a_i\ ,\ x\ \leq\ 10^9$$

نمونه‌ی ورودی و خروجی ۱

Input:

6 7 3
1 3 4 7 5 8
print
+ 3
print
-
print
-
print

Output:

4
3
4
5

توضیح :

در ابتدا سومین کوچکترین عنصر برابر ۴ است چون مرتب شده ی آرایه به صورت 1,3,4,5,6,7 است که سومین آن 4 است پس از اضافه شدن عدد 3 آرایه تبدیل به 1,3,3,4,5,6,7 می شود که سومین عدد آن 3 است. پس از حذف عدد 3 آرایه تبدیل به 1,3,4,5,6,7 می شود و بار دیگر سومین عدد کوچک ۴ می شود و پس از حذف عدد 4 آرایه تبدیل به 3,3,5,6,7 تبدیل می شود که سومین عنصر آن 5 است.

نمونه‌ی ورودی و خروجی ۲

Input:

```
3 8 4
1 2 3
print
+ 4
print
+ 10
+ 20
print
-
print
```

Output:

```
-1
4
4
10
```

توضیح: در ابتدا تعداد عناصر آرایه برابر ۳ است که از ۴ کمتر است بنابراین خروجی ۱- است. در print بعدی آرایه برابر 1,2,3,4 است بنابراین چهارمین کوچکترین عنصر برابر ۴ است. در print بعدی دو عدد که از 4 بزرگترند به آرایه اضافه شده اند بنابراین خروجی تغییر نمی کند. در نهایت در آخرین print آرایه برابر 1,2,3,10,20 می شود که جواب برابر ۱۰ است.

مسئله‌ی سوم: بلوک‌های چسبیده (۲۵ نمره)

- محدودیت زمان ۴ ثانیه
- محدودیت حافظه ۲۵۶ مگابایت

آلانا مشغول به رنگ آمیزی یک جدول است. در ابتدا جدول کاملا سفید است. او در مرحله خانه (i, j) را که داخل جدول است تصمیم می گیرد به رنگ سیاه در بیاورد. او دوست دارد در هر مرحله تعداد بلوک های متوالی سطری و ستونی ماکسیمال در این جدول را بشمارد. بلوک سطری و ستونی بلوک های به فرم $l * 1$ و $1 * l$ هستند که تماما سیاه هستند و ماکسیمال باشند به این معنی که بلوکی سطری یا ستونی نباشد که طول آن $1 + l$ باشد و همه ی این l خانه را داشته باشد. دقت کنید بلوک ماکسیمال ۱ * ۱ می تواند هم به صورت سطری شمرده شود هم ستونی.

ورودی

در خط اول سه عدد n, m, q داده می شود که به معنی این است که جدول $n * m$ است و q به معنی تعداد رنگ آمیزی هایی است که روی جدول انجام شده است. در هر کدام از q خط بعدی دو عدد i و j داده می شود که به معنی این است خانه ی (i, j) در جدول به رنگ سیاه در می آید.

خروجی

در q خط بعدی در هر خط تعداد مستطیل های ماکسیمال را خروجی دهید.

محدودیت‌ها

$$1 \leq n, m \leq 10^5$$
$$1 \leq q \leq 10^5$$
$$1 \leq i \leq n$$
$$1 \leq j \leq m$$

نمونه‌ی ورودی و خروجی ۱

Input:

3 3 5

1 1

1 2

2 1

2 2

1 3

Output:

0

1

2

4

4

مسئله‌ی چهارم: آشوب در ریچر (۲۵ نمره)

- محدودیت زمان ۱ ثانیه
- محدودیت حافظه ۲۵۶ مگابایت

اندرو رایان که از بیگ ددی ها برای محافظت از لیتل سیترز قطع امید کرده‌است خود دست‌به‌کار شده تا از آنها نگه‌داری کند. بدین منظور تعدادی از لیتل سیترها را از چنگ بیگ ددی‌های سمج با گفتن would you kindly درمی‌آورد.

حال می‌خواهد که از آنها در خانه خود که در شهر زیرآبی ریچر قرار دارد محافظت کند. ولی راه و رسم نگه‌داری از آنها را بلد نیست و از شما می‌خواهد که او را در این مسیر کمک کنید. او می‌داند که لیتل سیسترها به نوعی ماده دریایی به نام Adam اعتیاد دارند. بنابراین آنها را در یک ردیف در پشت گیت مخزن Adam قرار می‌دهد. Adam ها برای تهیه‌شدن نیازمند زمان هستند. به طوری که در بازه های t تایی تولید می‌شوند (اندرو رایان فرانک فانتین را مجبور کرده که از ثانیه صفر شروع به تولید کند). همچنین در مورد لیتل سیترها دو مورد را می‌داند.

یکی آنکه آنان تنبل بوده و از لحظه خاصی بلند می‌شوند تا Adam خود را بگیرند و دیگری آنکه هر کدام به گیت نزدیک تر باشد زور بیشتری برای گرفتن Adam دارد چون پلاسمید کمتری از او استخراج شده است! این را هم توجه کنیم که اگر در ثانیه t لیتل سیتری بلند شد تا Adam بگیرد ، از t+1 به بعد این کار را می‌کند.

حال اندرو می‌خواهد بداند که هر لیتل سیتر در چه زمانی Adam خود را دریافت می‌کند.

ورودی

در خط اول n ، تعداد لیتل سیترها و t ، مدت زمانی که طول می‌کشد یک واکسن Adam تولید شود را دریافت می‌کنید.

در خط دوم زمان بلند شدن هر لیتل سیتر را به ترتیب دریافت می‌کنید (به ترتیب نزدیک بودن به گیت مخزن Adam).

خروجی

در تنها خط برای هر لیتل سیستر زمانی که طول می‌کشد تا Adam دریافت کند را چاپ می‌کنید.

محدودیت ها

$$1 \leq n < 10^5, 1 \leq t < 2 * 10^7$$

نمونه‌ی ورودی و خروجی ۱

Input:

6 100

0 95 300 200 1000 0

Output:

100 200 400 300 1100 500

توضیح : برای هر نفر خروجی اینگونه است:

در زمان t = 0 لیتل سیستر اول بلند شده تا Adam دریافت کند و در t = 100 دریافت می‌کند.

در زمان t = 95 لیتل سیستر دوم بلند شده تا Adam دریافت کند و در t = 200 دریافت می‌کند.

در زمان t = 300 لیتل سیستر سوم بلند شده تا Adam دریافت کند و با توجه به اینکه نفر چهارم زودتر از او Adam را می‌گیرد باید صبر کند تا Adam بعدی تولید شود و در t = 400 آدام دریافت می‌کند.

در زمان t = 200 لیتل سیستر چهارم بلند شده تا Adam دریافت کند و در t = 300 دریافت می‌کند.

در زمان t = 1000 لیتل سیستر پنجم بلند شده تا Adam دریافت کند و در t = 1100 دریافت می‌کند.

در زمان t = 0 لیتل سیستر ششم بلند شده تا Adam دریافت کند ولی در زمان‌های 100 200 300 400 آدام به ترتیب توسط نفر اول دوم چهارم سوم گرفته می‌شود (چون به مخزن نزدیک تر هستند) و در t = 500 آدام دریافت می‌کند.

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).