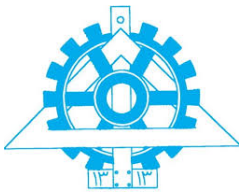

به نام خداوند بخشنده مهربان

معین سرخه ای-سید محمدرضا صالحی دهنوی
m.moein.sorkhei@gmail.com

۱ اردیبهشت ماه ۱۳۹۶

درخت ها در سازمان سنجش!!

سازمان سنجش تصمیم در بالا بردن سرعت اعلام نتایج کنکور سراسری دارد. بدین منظور یک کامپیوتردان[!!] این سازمان با شما نشستی تشکیل داده و اطلاعات زیر را در اختیار شما می‌گذارد :

۱. میخواهیم برای هر فرد یک شناسه و یک رتبه قرار دهیم و آن ها را با دستور :

insert NAME RANK

وارد برنامه کنیم.

۲. میخواهیم اگر نیاز شد بتوانیم افرادی را از سامانه خود حذف کنیم بدین ترتیب باید دستور زیر موجود باشد.

delete NAME

۳. میخواهیم در مواقع لزوم بتوانیم همه دانش اموزان را به صورت صعودی بر اساس رتبه چاپ کنیم بدین منظور برنامه شما باید از دستور زیر پشتیبانی کند.

print

۴. قطعا بعد از اعلام نتایج عده ای برای اعتراض خواهند امد برنامه شما باید بتواند با گرفتن $NAME$ رتبه شخص را در سامانه پیدا کند تا بتوانیم بر اساس ان تصمیم گیری کنیم.بنابراین دستور زیر را هم در برنامه‌ی خود داشته باشید.

getnum NAME

فقط توجه داشته باشید از آن جا که تعداد درخواست های اخر ممکن است زیاد باشد باید بتوانید آن را باپیچیدگی $log(n)$ پیاده سازی کنید و باید پیچیدگی تمام عملیات های شما از $O(log(n))$ باشد. همچنین از آن جا که در دستورات نوع ۲ و ۴ فقط نام را دارید باید برای نگهداری رتبه و نام ساختمان داده‌ی مناسبی انتخاب کنید.

Input:

insert 1 2

insert 2 10

insert 3 14

insert 4 1

insert 5 23

insert 6 70000

insert 7 12000

getnum 2

print

Output:

2

NAME : 4 RANK : 1 NAME : 1 RANK : 2 NAME : 2 RANK : 10 NAME 3 RANK : 14 NAME : 5

RANK : 23 NAME : 7 RANK : 12000 NAME : 6 RANK : 70000

مدیریت بسته های شبکه

بخش اول

میدانیم که $AVLtree$ در علم کامپیوتر کاربرد های گوناگونی دارد. برای مثال در سیستم عامل $Linux$ جهت مدیریت حافظه $process$ ها از این ساختمان داده استفاده میشود. به عنوان کاربردی خاص در نظر بگیرید که شما قرار است بخشی از سیستم عامل را طراحی کنید که نگهداری بسته های شبکه را مدیریت میکند. به دلیل ارجاع های فراوانی که (در سیستم عامل خاص منظوره ما!) به $packet$ ها وجود دارد و این که با رسیدن $packet$ های جدید باید آنها را نیز نگهداری کنیم، تصمیم گرفتیم از یک $AVLtree$ برای نگهداری $packet$ها (بر اساس idآنها) استفاده کنیم تا بتوانیم در زمان مناسب عملیات جست و جو و اضافه کردن یک $packet$ جدید به مجموعه $packet$ ها را انجام دهیم. رسیدن هر $packet$ جدید به کارت شبکه با دستور $insert(PacketId)$ مشخص میشود و شما باید آن را به درخت خود اضافه کنید (توجه کنید که با اضافه کردن همچنان باید خاصیت $AVLtree$ برقرار باشد)

حال فرض کنید نیاز است که سرویس جدیدی به نام $CountInRange(k1, k2)$ در سیستم عامل طراحی کنیم که تعداد $packet$ هایی که id آنها بین $k1$ و $k2$باشد را برگرداند $(k1 \leq PacketId \leq k2)$. هزینه زمانی پاسخگویی این سرویس باید از مرتبه زمانی $O(logn)$ باشد که n تعداد $packet$ ها موجود در درخت است (اگر نیاز به فیلد اضافه ای درون گره های درخت دارید آن را تعریف و از آن استفاده کنید).

ورودی

ورودی شامل تعدادی مورد آزمون است. به ازای هر مورد آزمون تعدادی دستور $Insert$ و تعدادی دستور $CountInRange$ وجود دارد (تمام دستورات $CountInRange$ بعد از دستورات $Insert$ قرار دارند). پایان هر مورد آزمون با علامت # مشخص میشود. پایان ورودی با علامت ## مشخص میشود.

خروجی

در هر مورد آزمون، به ازای هر دستور $CountInRange$ باید تعداد $packet$ هایی که در بازه مورد نظر (مطابق آن چه در صورت مساله توضیح داده شد) را به عنوان خروجی برگردانید و در پایان هر مورد آزمون علامت$ قرار دهید.

Input:

Insert 8

Insert 5

Insert 10

Insert 4

Insert 6

CountInRange 1 15

#

Insert 15

Insert 6

Insert 5

Insert 7

Insert 4

Insert 71

Insert 50

Insert 23

CountInRange 23 75

CountInRange 15 80

##

Output:

5

$

3

4

$

بخش دوم

فرض کنید که تا کنون دو $AVLtree$ در سیستم عامل ما به نام های $T1$ و $T2$ استفاده میشده است. اکنون به دلایل تصمیماتی که برای مدیریت حافظه گرفته ایم نیاز داریم تا $packet$ ها را بین درخت ها جابجا کنیم. قصد داریم سرویسی به سیستم عامل اضافه کنیم که به صورت $MoveFrom(T1, T2, PacketId)$ و $MoveFrom(T2, T1, PacketId)$ فراخوانی می شود و باید $packet$ ای با Id مشخص شده را از درخت $T1$ به $T2$ (در مثال اول) و یا برعکس (در مثال دوم) جابجا کند. هزینه پاسخگویی این سرویس باید از مرتبه زمانی $O(log(n1) + log(n2))$ باشد ($n1$ و $n2$ تعداد $packet$ های موجود در هر درخت است).

ورودی

در هر مورد آزمون در خط اول عبارت $T1$ است که نشان دهنده درخت اول است. بعد از آن تعدادی $Insert$ وجود دارد که برای ساختن $T1$ استفاده می شود. پس از آن عبارت $T2$ به همراه تعدادی $Insert$ وجود دارد که برای تشکیل درخت دوم است. در نهایت دستور $Move$ می آید که پارامترهایش به ترتیب درخت مبدا، درخت مقصد و $packetId$ برای $packet$ ای که باید منتقل شود است. بعد از دستور $Move$ مورد آزمون بعدی شروع می شود. پایان ورودی با علامت $\#\#$ مشخص می شود.

خروجی

به ازای هر دستور $Move$ شما باید پیمایش $PreOrder$ درخت $T1$ و $T2$ (هر کدام در یک خط) را چاپ نمایید. در پایان هر مورد آزمون علامت $ را چاپ نمایید.

Input:

T1

Insert 3

Insert 2

Insert 4

T2

Insert 8

Insert 1

Insert 9

Move T1 T2 12

T1

Insert 15

Insert 7

Insert 18

Insert 20

T2

Insert 6

Insert 3

Insert 25

Insert 2

Insert 4

Move T2 T1 25

##

Output:

3 4

8 1 2 9

$

15 7 20 18 25

3 2 6 4

$

باید کدها و توابع شما به صورت مناسب پیاده سازی شده باشند و این بخشی از نمره شما را تشکیل خواهد داد.
برای دیدن اعمالی مانند $insertion$، $deletion$ و ... روی $AVLTree$ می توانید از لینک زیر استفاده کنید:
$https://visualgo.net/en/bst$