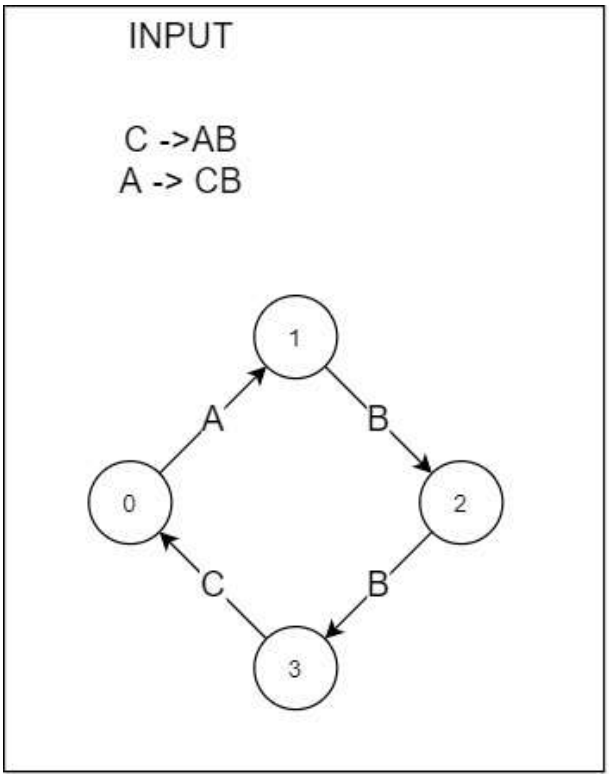
بازسازی گراف

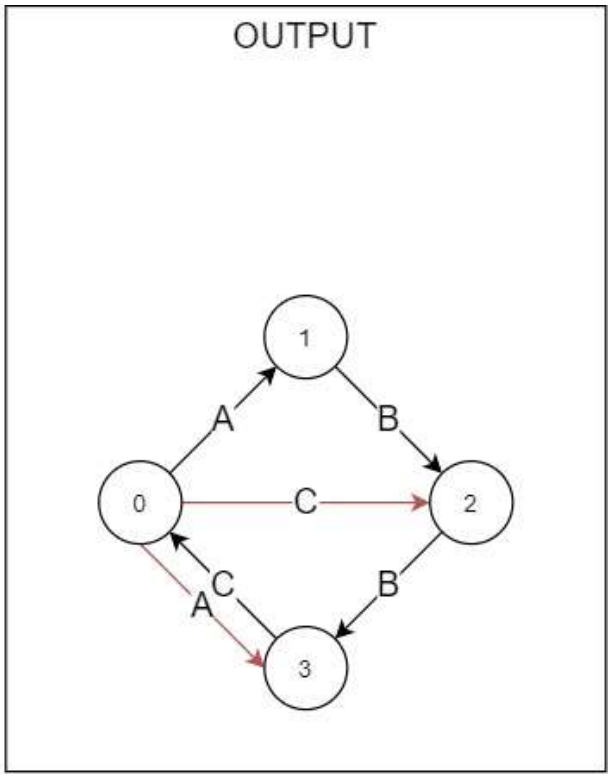
این پروژه به صورت دستی بررسی و تصحیح می‌شود.

یک گراف جهت دار، غیر ساده و وزن دار با n راس به شما داده میشود. همچنین یک مجموعه قوانین نیز برای شما قرار داده شده است. مجموعه قوانین راجع به گراف به این صورت است که دو یال از گراف را میگیرد و به ازای آنها یک یال جدید اضافه میشود.

به عنوان مثال فرض کنید:

در این مثال قوانین به این صورت است که هرجا AB دید، باید C قرار دهد و هرجا CB دید باید A قرار دهد. یعنی از راس 0 به راس 1 با یال A و از راس 1 به راس 2 با یال B میرود. با توجه به قانون داده شده به ازای هر AB قرار است C نمایش بدهد پس از 0 به 2 یک C اضافه میکند و در ادامه از 0 به 2 با C میرود و از 2 به 3 با B میرود و در نتیجه طبق قانون از 0 به 3 یال A اضافه میشود. این کار را تا زمانی که دیگر یالی اضافه نشود، تکرار میکنیم.





ورودی

ورودی شما باید دو عدد فایل متنی .txt باشد. یکی به اسم graph.txt که برای گرفتن اطلاعت مربوط گراف است و دیگری grammar.txt که مربوط به قواعد مربوط به اضافه کردن یال می‌باشد.

برای فایل grammar.txt:

برای قاعده C->AB در یک خط به شکل زیر قرار میدهیم.

```
C	A	B
```

توجه کنید: در یک خط دقیقا 3 متغیر قرار دارد. یعنی قاعده حتما دو یال را به یک یال تبدیل می‌کند.

برای فایل graph.txt:

در سطر اول گراف تعداد راس ها (n) را میبینید. و در سطرهای بعدی به ترتیب ابتدا شماره راس مبدا (A)، شماره راس مقصد (B) و با وزن یال از A به B

$$1 \leq A, B, n \leq 40000$$

خروجی

در خروجی فقط تعداد یال‌های اضافه شده در انتها بنویسید. (عملکرد کد بررسی خواهد شد)

مثال

ورودی 1 مربوط به فایل grammar.txt

```
C	A	B
A	C	B
```

که همان C->AB و A->CB است.

ورودی 1 مربوط به فایل graph.txt

```
4
0    1    A
1    2    B
2    3    B
3    0    C
```

مشابه تصویر بالا می‌باشد.

##خروجی 1

```
2
```

برای تست کیس‌های بیشتر میتوانید از دیتاست‌های گوگل درایو زیر استفاده کنید:

https://drive.google.com/drive/folders/1yR_hWrWdvCF9xtg1qudCt88X-CYWySU4?usp=sharing

اطلاعات عمومی: کاربرد این پروژه در زیر رشته Programming Languages ، Software Analysis و کامپایلر است و به آن CFL-REACHABILITY می‌گویند.

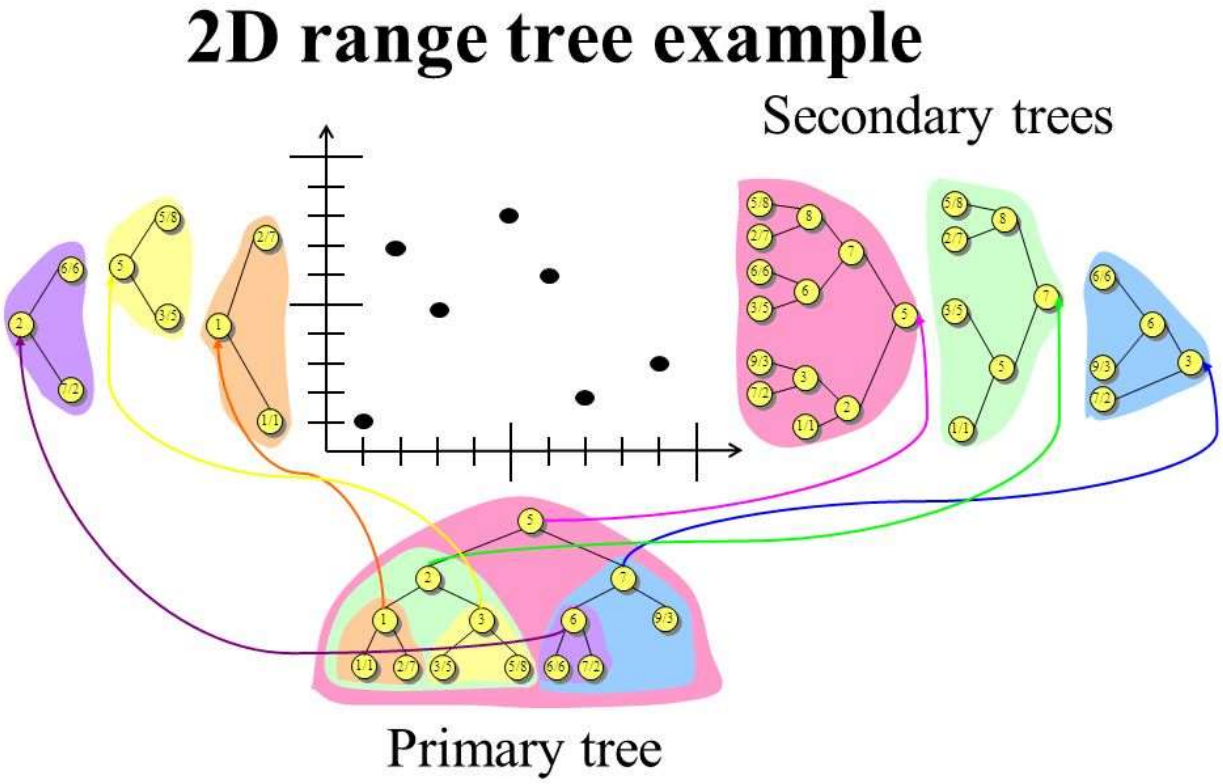
Range Tree

ما قصد پیاده سازی یکی از ساختمان داده های مکانی (spatial) برای R2 را داریم. نام این ساختمان داده range-tree است. این درخت یکی از پرکاربردترین ساختمان داده ها در بانک های اطلاعاتی مکانی است. این ساختمان داده برای هر بعدی از فضا خوش تعریف است اما برای سادگی پیاده سازی فرض میکنیم فضا به یک صفحه اقلیدسی محدود شده است.

قبل از اینکه range-tree در دو بعد را توضیح دهیم ابتدا در یک بعد را توضیح میدهیم و بعدا بیان خواهیم کرد چگونه می توان آن را به دو بعد (و حتی ابعاد بالاتر) گسترش داد. فرض کنید n نقطه در صفحه داده شده است. ابتدا فقط مقادیر x این نقاط را (که مجموعه ای از اعداد بر روی محور x می باشد) در نظر می گیریم. ما قصد داریم ساختمان داده ای بسازیم که آماده جواب دادن به این سوال باشد: برای یک بازه داده شده مانند I=[a,b] بر روی محور x، مقدار x کدام یک از نقاط اولیه در بازه I قرار دارد. درخت اولیه برای یک بعد (محور x) به این صورت ساخته می شود. ابتدا همه نقاط را بر اساس مقدار x مرتب میکنیم. سپس دو تا دو تا، به ترتیب به یک گره پدر وصل کرده و مقدار گره پدر، برابر خواهد بود با مقدار بزرگترین x در زیردرخت چپ آن. اگر در ابتدا تعداد فرد بود، همین عملیات دودویی را برای گره آخر با گره پدر جفت آخر تکرار میکنیم. همین عملیات را تکرار میکنیم تا به یک گره برسیم. نتیجه درخت بالانسی خواهد بود (مانند AVL) که در زمان O(logn) می توان در آن جای یک مقدار x را (همانند BST) پیدا کرد. به این درخت، درخت rang-tree بر روی محور x میگوییم و با Rx نشان می دهیم.

بعد از ساخت درخت برای محور x به سراغ محور دوم می رویم. برای هر گره میانی مانند v در درخت Rx (گره ای که برگ نباشد)، یک اشاره گر وجود دارد به درخت دومی. این درخت دوم برابر است با درخت range-tree از تمام برگ هایی که در زیردرخت v قرار داشتند، اما بر اساس مقادیر y. به این درخت، درخت rang-tree بر روی محور y برای گره v میگوییم و با Ruv نشان می دهیم.

در مثال شکل زیر ۷ نقطه وجود دارد: (1,1),(9,3),(2,7),(7,2),(3,5),(5,8),(6,6).



برای انجام جستجو در این درخت برای فضای دو بعدی (که در زمان O(log2n+k) انجام می شود و k تعداد نقاط داخل جواب نهایی هستند) یک مستطیل مانند R=[(x1,y1),(x2,x2)]داده شده است و ما قصد داریم تمام نقاط داخل یا روی این مستطیل را گزارش کنیم (نقطه اول گوشه چپ پایین و نقطه دوم گوشه راست بالا است). برای جواب دادن به این سوال ابتدا روی Rx شروع به جستو می کنیم. گره ای که مقدارش از x1 بزرگتر باشد و از x2 کوچکتر باشد را گره u می نامیم (اگر در هنگام جستجو مقدار گره میانی با مقدار x2 برابر شد برای ادامه جستجو به سمت چپ می رویم). تمام گره هایی که بین دو مسیر جستجو برای x1 و x2 ، از ریشه تا برگ، قرار میگیرند دارای اهمیت هستند و تمام نقاط در زیردرخت ها امکان حضور در جواب را دارند و باید مقدار y آنها مورد بررسی قرار بگیرد. پس به سراغ هر کدام از این درخت ها مانند Ruv می رویم. در این درخت ها شروع به جستجو کرده و گره ای را که مقدارش از y1 بزرگتر باشد و از y2 کوچکتر باشد را گره t می نامیم. تمام برگ هایی که بین دو مسیر جستجو برای y1 و y2، از ریشه تا برگ، قرار می گیرند، نقاطی هستند که در جواب نهایی وجود دارند. در انتها فقط یک مرتب سازی بر اساس مقادیر y نیاز هست تا با تست های کویرا منطبق شود (وگرنه ضرورتی برای مرتب بودن نتیجه وجود نداشت).

برای مطالعه بیشتر می توانید منابع زیر را مطالعه کنید، اما دقت کنید کد کپی نکنید و همچنین به دنبال ساختمان داده های دیگر نروید. در ضمن توصیه می شود از زبان های مناسب این کار مثل Cpp بهره ببرید (مسولیت استفاده از زبان های دیگر با دانشجو است):

- https://www.youtube.com/watch?v=xVka6z1hu-I
- http://www.cs.umd.edu/~meesh/cmsc420/ContentBook/FormalNotes/MountNotes/lecture20-rangetrees.pdf

راستی ویکیپدیا در مورد پیچیدگی زمانی اشکالاتی دارد (دقت کنید). زمان ساخت شما باید بیشتر از O(nlogn) زمان و حافظه مصرف نکند و جواب هر سوال را در O(log2n+k) بدهد (زمان مرتب سازی آخر در نظر گرفته نشده است).

ورودی

خط اول تعداد نقاط ورودی را نشان می دهد. در خط دوم مقادیر x نقاط ورودی و در خط سوم مقادیر y متناظر با مقادیر خط قبلی داده شده است. پس اولین ایکس و اولین وای با یکدیگر اولین نقطه را نمایش می دهد. خط چهارم تعداد عملیات های بعدی را مشخص می کند. از خط ۵ به بعد هر خط چهار مقدار را مشخص می کند که به ترتیب مختصات x1,y1,x2,y2 برای یک مستطیل است. فرض کنید تعداد ورودی و تعداد عملیات ها از 104 بیشتر نخواهد بود.

خروجی

برای هر عملیات از خط ۵ به بعد در یک خط مقادیر x و در خط بعدی مقادیر y نقاطی که در داخل مستطیل می افتند را چاپ کنید. **دقت کنید نقاط خروجی به ترتیب مقدار y از کوچک به بزرگ مرتب شده باشند.** اگر جواب خالی بود None را چاپ کند.

مثال

ورودی نمونه ۱

```
3
1 2 3
1.2 2.1 3.3
2
0 0 2.1 1.4
1.5 2.2 2.8 5
```

خروجی نمونه ۱

```
1
1.2
None
```

ورودی نمونه ۲

```
7
1 9 2 7 3 5 6
1 3 7 2 5 8 6
3
0 0 5 5
8 1 10 5
0 0 10 10
```

خروجی نمونه ۲

```
1 3
1 5
9
3
1 7 9 3 6 2 5
1 2 3 5 6 7 8
```

Segment Tree

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

یک آرایه به نام a و طول n به شما داده شده است.

هم‌چنین q درخواست به شما داده می‌شود. هر درخواست یکی از دو نوع زیر است:

- `add l r` : به عدد lام آرایه یک واحد اضافه کن، به عدد $l+1$م آرایه دو واحد اضافه کن به عدد $l+2$م آرایه سه واحد اضافه کن، ... و به عدد rم $r-l+1$ واحد اضافه کن.
- `ask l r` : باقی‌مانده‌ی تقسیم جمع اعداد بازه‌ی l تا r آرایه بر عدد 10^9+7 چیست؟(یعنی اگر حاصل جمع $a[l]+a[l+1]+...+a[r]$ را حساب کنیم و آن را بر عدد 10^9+7 تقسیم کنیم حاصل چه می‌شود)

شما باید جواب درخواست‌های نوع دوم را بدهید.

ورودی

در خط اول ورودی به شما به ترتیب دو عدد n طول آرایه و q تعداد درخواست‌ها داده می‌شود.

در خط دوم ورودی n عدد شما داده می‌شود که iامین آن‌ها نشان دهنده‌ی مقدار اولیه $a[i]$ است.

در q خط بعدی در هر خط یک اتفاق داده شده است.

$$1 \leq n, q \leq 2 * 10^5$$

$$1 \leq l \leq r \leq n$$

$$0 \leq a[i] \leq 10^9$$

خروجی

شما باید به ازای هر درخواست نوع دوم جواب درخواست را در یک خط جداگانه چاپ کنید.

مثال

ورودی نمونه ۱

```
6 10
3 5 2 6 8 2
ask 1 1
add 1 5
ask 1 3
ask 2 6
add 4 6
ask 2 5
add 2 5
add 2 5
add 5 6
ask 1 6
```

خروجی نمونه ۱

```
3
16
37
38
70
```

ورودی نمونه ۲

```
5 5
0 0 0 0 0
add 1 1
```

```
add 2 2
add 3 3
add 4 4
ask 1 5
```

خروجی نمونه ۲

```
4
```