1. چهارمین کوچکترین عدد می تواند در ردیف دوم، سوم و چهارم باشد.

درایه ها: 2و3و4و5و6و7و8و9و10و11و12و13و14و15

Input = 02;

Input = 50;

Input = 45;

Input = 70;

Input = 75;

Input = 22;

Input = 40;

Input = 55;

Input = 45;

خانه با شماره 1 : همیشه عدد 5 در این مکان قرار میگیرد.

خانه با شماره 2 : یکی از چهار عدد باقی مانده است، پس چهار حالت دارد.

خانه با شماره 3 : سه عدد باقی مانده دارای مقدار ماکسیمم است و مقدار ماکسیمم حتما باید در این مکان باشد. پس یک حالت دارد.

خانه با شماره 4 و 5 : دو عدد باقی مانده باید در این خانه باشند.

حالات کل برابر است با :

1. 2 جابجایی لازم است.

1. الف) حداکثر تعداد rotation در هنگام insert برابر 2 تا است.

زیرا در هر بار اضافه کردن شرط قرمز سیاه بودن (ارتفاع دو طرف درخت چک میشود) و در اضافه کردن بعدی یا با جابه جا کردن رنگ شرط قرمز سیاه بودن درست میشود یا باید درخت را چرخاند که این خود دو حالت میتواند داشته باشد اولی برای خط راست کردن و دومی هم برای اندازه کردن طول درخت.

ب)

ج) هزینه سورت روی این درخت است.



مانند همان red black tree است ولی علاوه بر آن یک فیلد size نیز اضافه میکنیم که برابر است با تعداد فرزندان زیر گره به علاوه یک.

در هنگام insert کردن مانند درخت rb معمولی عمل میکنیم ولی از هر گره که عبور میکنیم یک عدد به size اضافه میکنیم و در هنگام rotation در insert نیز به این صورت عمل میکنیم که :

اگر بخواهیم y به سمت چپ برود.

Delete: در هنگام delete نیز اگر نیاز به rotate بود مانند حالت بالاست و در هنگام delete نیز از هر گره که رد میشویم یک عدد از size کم می کنیم. و اگر مقدار در درخت نبود دوباره از درخت عبور میکنیم و به size هر گره یک عدد اضافه میکنیم.

Find: این عمل مانند همان red black tree معمولی انجام میشود.

Count: در این عمل بعد از find کردن x کافیست size-1 آن را بازگردانیم.

1. برای این کار درخت min-heap ای در نظر میگیریم.

و کارهای زیر را انجام میدهیم:

1. آرایه ای به طول m\*n در نظر میگیریم.
2. یک min-heap با سایز m درست میکنیم و اولین المنت تمام آرایه ها را در آن درج میکنیم.
3. گام های زیر را m\*n بار انجام میدهیم:
   1. کوچکترین عضو را از درخت میگیریم (ریشه درخت) و در آرایه خروجی ذخیره میکنیم.
   2. ریشه درخت را با خانه بعدی آرایه ای که ریشه از آن آمده بود را قرار می دهیم و اگر آرایه خالی بود مقدار بینهایت را قرار می دهیم و درخت را دوباره سورت میکنیم.
4. دو حالت تاثیر یکسانی ندارند مانند مثال زیر:
5. حذف 2 و 1:



1. حذف 1 و 2:

1. درخت متناظر با عبارت داده شده درخت زیر است:

عنصر A حتماً باید در ابتدا اضافه شود، زیرا در جایگشت های دیگه ریشه درخت تغییر میکند.

و بعد از آن نیز عنصر E باید اضافه شود زیرا عنصز بعدی حتما از A بزرگ تر است و در جایگاه آن مینشیند.

در بقیه حالات باید درخت دست چپ E و درخت دست راست آن به همان ترتیب اضافه شود. یعنی 5! تعداد کل حالات است که در نصف آنها b قبل از c آمده است و حالات ترتیب آمدن درخت سمت راست E رعایت شده است. پس جواب کل به فرم زیر میشود:

1. 1. درخت جدیدی ایجاد میکنیم با نام h’ و ریشه آن را با ریشه درخت اول (h) یکی قرار میدهیم و هر سری درخت را آپدیت میکنیم بدین صورت که ریشه را از h’ حذف میکنیم و در درخت h بچه های متناظر با آن درایه را به h’ اد میکنیم. و از k یک واحد کم میکنیم. و درخت h’ را سورت میکنیم. در پایان k مرحله، kامین کوچکترین عدد در ریشه قرار دارد.
   2. در این سوال در درخت بازگشتی پیمایش میکنیم، اگر عنصر از x کوچکتر باشد به شمارش گر یک واحد اضافه میکنیم و سرغ فرزند ها میرویم و اگر بزرگتر از x باشد به مرحله قبلی باز میگردیم. به این دلیل که تمامی فرزندان آن عنصر از خود عنصر بزرگتر است پس از x نیز بزرگتر است.

کد c++ آن در زیر قرار دارد.

