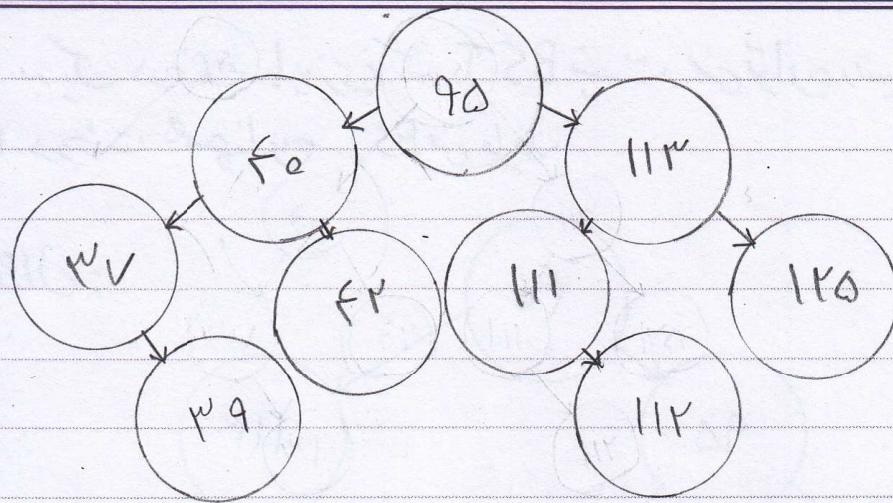


Date: ..... Subject: سری ۳ هایین ساخته ای داده از فرزنده



لای اینکه رأس که اعماقی سود ارتفاع درخت را زیاد کند باید به عنوان فرزنده باشیم. همچنان ارتفاع باید درخت اعماق سود باشد. با براین باید فرزن ۳۹ یا ۱۲ باشد.

فرزن کنیم اگر فرزن ۳۹ باشد با توجه به درخت داریم:

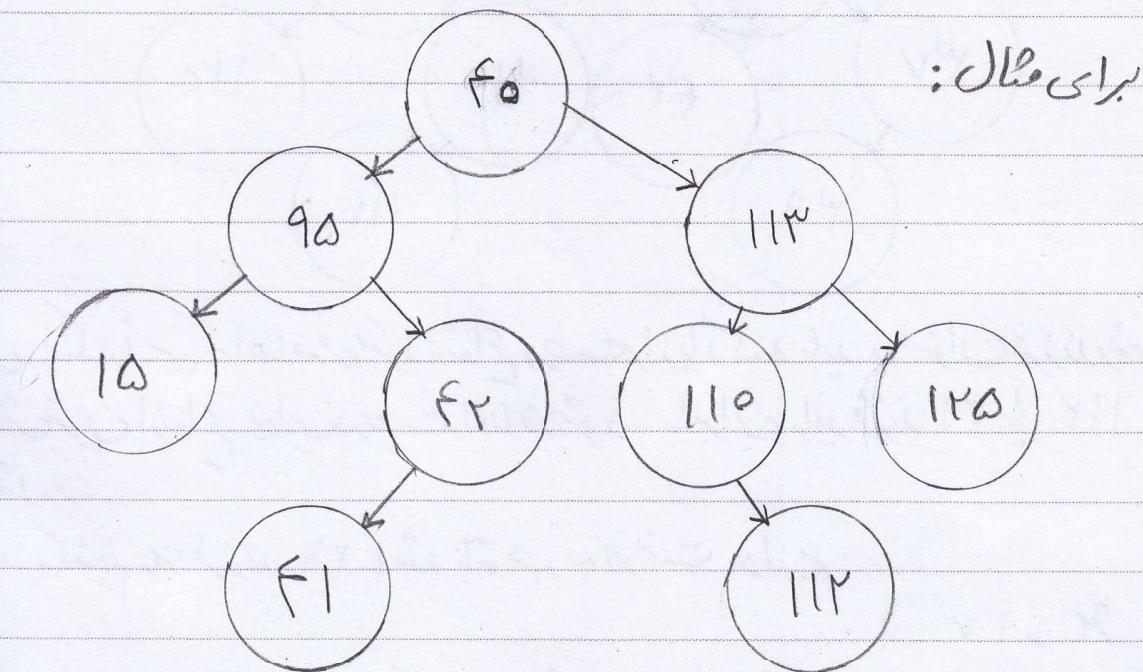
$$\left. \begin{array}{l} n > 39 \\ n < F_0 \\ n \neq 39 \end{array} \right\} \Rightarrow n = 38 \Rightarrow \text{با براین } n=38 \text{ تنها مقدار صحیح است که درخت داریم. این نتیجه طبقه صورت خواهد کرد.}$$

فرزن کنیم اگر فرزن ۱۲ باشد با توجه به درخت داریم:

$$\left. \begin{array}{l} n > 11 \\ n < 11^2 \\ n \neq 11^2 \end{array} \right\} \Rightarrow \text{با براین همچنان مقدار صحیح درایم} \Rightarrow \text{لای فشار} \Rightarrow \text{نتیجه صورت خواهد کرد.}$$

Date: ..... Subject: .....

مسئلہ: ایک درخت بائیزی کر کے BST نیت فرخان رسمی جوں باہم  
دینے والے BST فرخان کو rotate کریں



سک: در هر سه طبقه بجزی استفاده از تابع بازگشت از Stack

st.push(root)

الگوریتم پیاسن : inorder

while (st.size != 0)

از پیش شروع می کنیم و هر بار که

v = st.top()

روی یک رأس هستیم اگر فرزند چپ

if (v.left == NULL)

نمایست یا قبل از فرزند چپ آن میده

{ print. v.val()

شدید بود، نوبت خودش است

st.pop()

که حاصل شود و سپس نوبت زیرخت

if (v.right != NULL)

میگوییم آن می شود.

st.push(v.right())

else

زمان اجرای الگوریتم  $O(n \log n)$

که  $n$  برابر تعداد رؤوس دخت است.

st.push(root)

الگوریتم پیاسن : preorder

while (st.size != 0)

از پیش شروع می کنیم و هر بار که

v = st.top()

روی یک رأس هستیم آن را اخراج

{ st.pop()

می کنیم و فرزند راست را در

while (v != NULL)

الفداری می کنیم و به کجا می بینیم

{ print. v.val()

جهد آن می رویم.

{ st.push(v.right())

v = v.left()

زمان اجرای الگوریتم  $O(n \log n)$  باشد

که  $n$  برابر تعداد رؤوس دخت است.

Date:

Subject:

St.push(root)

الgoritم بیانی پس

while( st.size != 0 )

از شروع فرآیند و هر بارز

{ v = St.top()

یک رأس اگر فرزند خود داشت

{ if ( v.left != NULL )

و دیده نشده بود آن سمت فریم

{ St.push(v.left())

در غیر این صورت اگر فرزند راست

{ { v.left() = NULL }

داشت و دیده نشده بود آن سمت

{ elseif ( v.right != NULL )

فریم در غیر این صورت

{ St.push(v.right())

ذرت خود آن رأس

{ { v.right() = NULL }

محاسبه

{ else

{ print v.val()

زمان اجرای الgoritم  $O(n)$

{ St.pop()

صباخ که  $n$  تعداد رؤوس

درخت است.

Date:

Subject:

سرچ: (الف) یک متغیر  $res = true$  تعریف می‌کنیم که جواب است. درخت را پیاپی می‌کنیم و هر رأس که مقادیر آن را در درخت  $T$  حسبت و جویی کنیم که  $(deg)_r \leq log(n)$  طول کشید و اگر این مقدار در  $T$  بود  $O(n \log n)$  بار این کار را کرده ایم. در مجموع  $n$  بار این کار را کرده ایم.  $res = false$  عملیات انجام داده ایم.

(ب) در آرایه به اندازه  $n$  و  $m$  تعریف می‌کنیم و هر دو درخت را به صورت  $inorder$  می‌کنیم و در این دو آرایه می‌زنیم. حالا دو مجموعه عدد در در آرایه به صورت مرتب شده قرار دارند. تا  $i$ -یم  $O(n+m)$  عملیات انجام داده ایم. فرض کنیم در آرایه را  $a$  و  $b$  بنامیم. حال می‌خواهیم چک کنیم که مقادیر زیر مجموعه  $i$  مقادیر  $b$  است یا نه. دو استاره گرزوخ را بر سر  $a$  و  $b$  قرار می‌دهیم. ناید مقترانه  $a[i:j]$  در  $b$  موجود باشد، پس این تا جایی که  $a[i:j] < b$  می‌باشد خواهد بود. اگر بحالات  $a[i:j] > b$  رسیدیم به این معنی است که  $a[i:j] > b$  موجود نیست و س جواب منفی است.

در غیر این صورت بحالات  $a[i:j] = b$  و رسیدیم که هم‌اونه خواهد بود که  $a[i:j] < b$ . در اینجا نیز  $O(n+m)$  عملیات انجام دادیم. پس در مجموع حافظه ای اضافی  $O(n+m)$  و تعداد  $O(n+m)$  عملیات انجام داده ایم.

Date:

Subject:

سی: دو آرایه س باقی رفته کنیم:  $ma[v]$ ,  $mi[v]$   
 بزرگترین مقدار در زیر دخته ای  $v$ :  $ma[v]$   
 کوچکترین مقدار در زیر دخته ای  $v$ :  $mi[v]$

اگر مقدار باشیم کافی است برای هر ای  $v$  داشت  
 $ma[v.left] < v.val() \leq ma[v]$  : باشیم  
 $mi[v.left] > v.val() \leq mi[v]$

که چک کردن این شرایط  $O(n)$  زمانیست برای حالات کافی است  
 را در  $O(n)$  پیدا کنیم.

بررسی خود را پس از postorder کنیم و ثابت هر راس که  $v$  است  
 بزرگترین مقدار درخت هایش مقدار  $ma$  و کوچکترین مقدار  $mi$  است  
 برای  $v$  داریم:  
 $ma[v] = \max \{v.val(), ma[v.left], ma[v.right]\}$

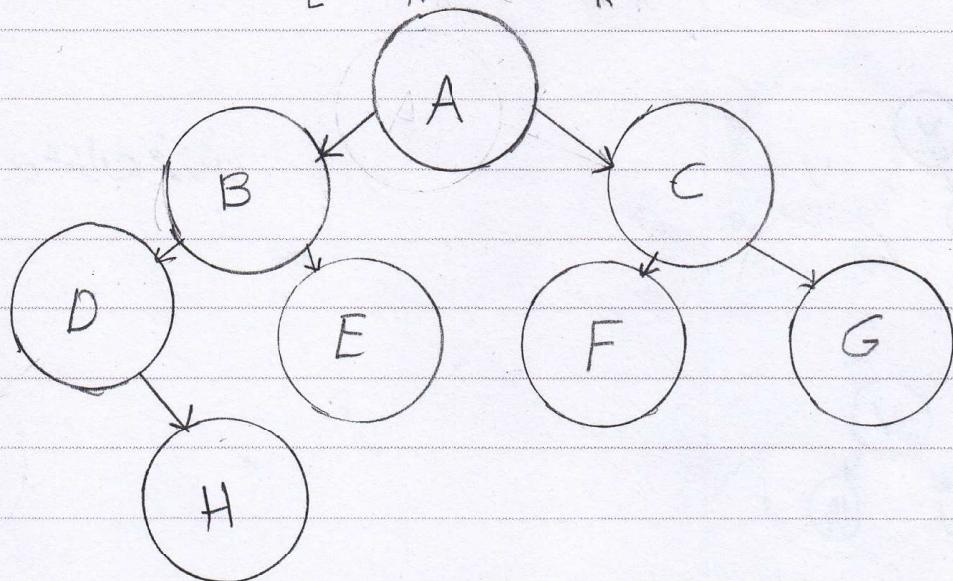
$mi[v] = \min \{v.val(), mi[v.left], mi[v.right]\}$

بنابراین این عملدریز  $O(n)$  برداشت می‌آید.

پس زمان اجرای الگوریتم در مجموع  $O(n)$  است.

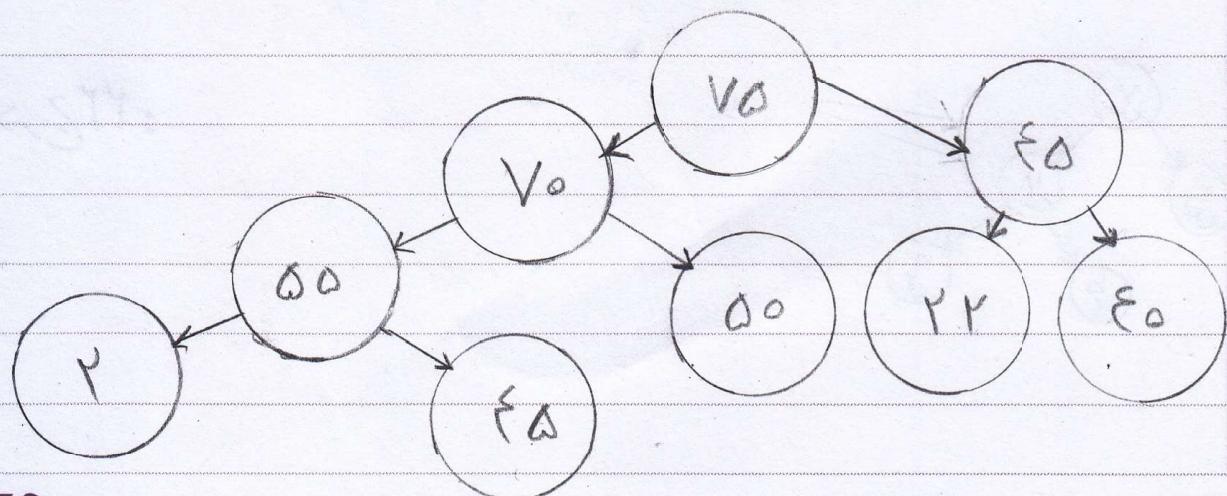
Date: ..... Subject: .....

Inorder traversal = D, H, B, E, A, F, C, G : ↗  
Postorder traversal = H, D, E, B, F, G, C, A



P, D, f, V, V, P, E, Q, Q, E, Q  
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓  
P, V, E, V, D, P, E, Q, Q, E, Q

V, V, E, Q, Q, D, P, V, E, P, E  
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

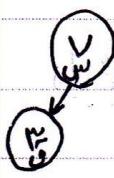


Date:

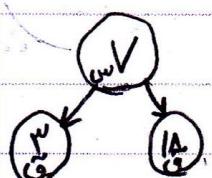
Subject:



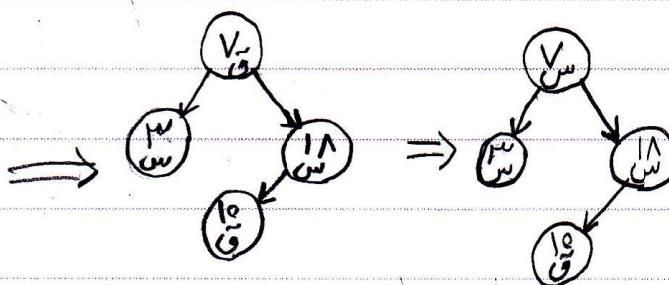
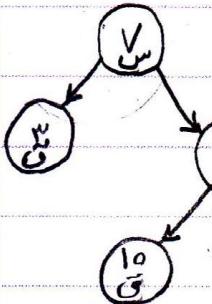
حصہ ۱: الف) درج ۷: مختار ۷ طبق عنوان رسید  
از نگ سیاه درج حق کنیم (Case 1).



درج ۳: ۳ به عنوان فرزند چپ ۷ قرار می‌گیرد (به علت کوچکتر بودن)  
و رنگ سیاه قرفزی می‌سود.

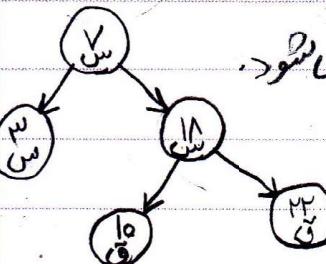


درج ۱۸: ۱۸ به عنوان فرزند راست قرار می‌گیرد (به علت بزرگتر بودن)  
و رنگ سیاه قرفزی می‌سود.



درج ۱۰:

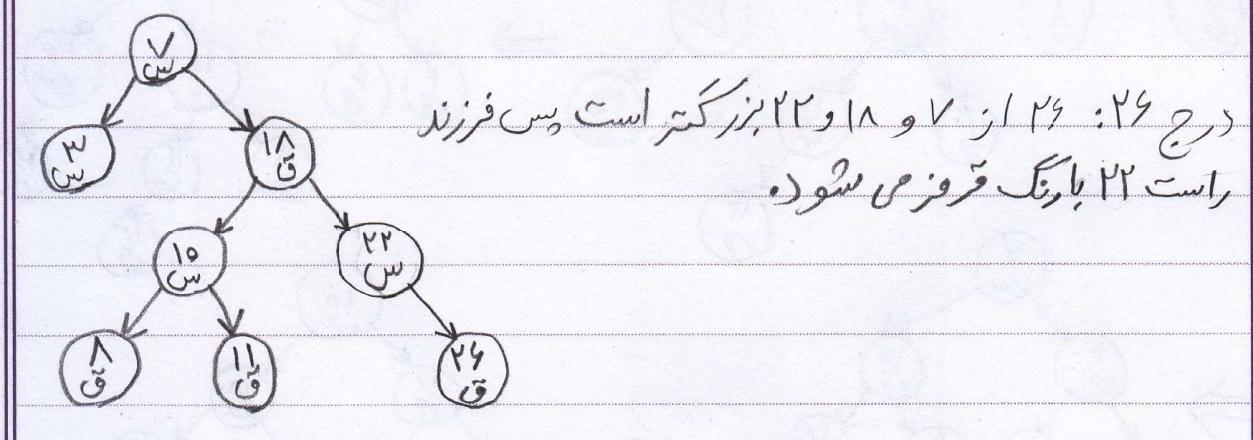
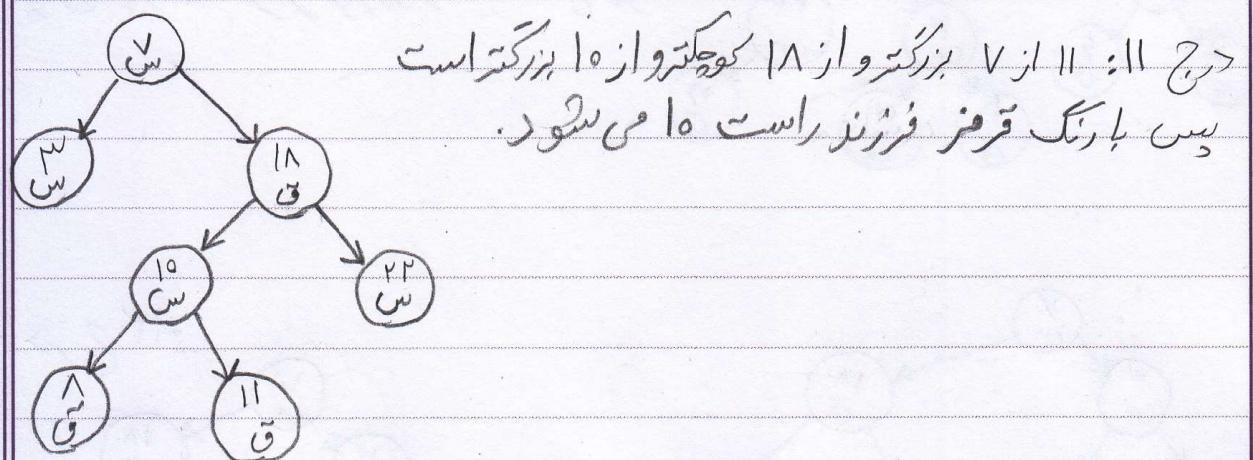
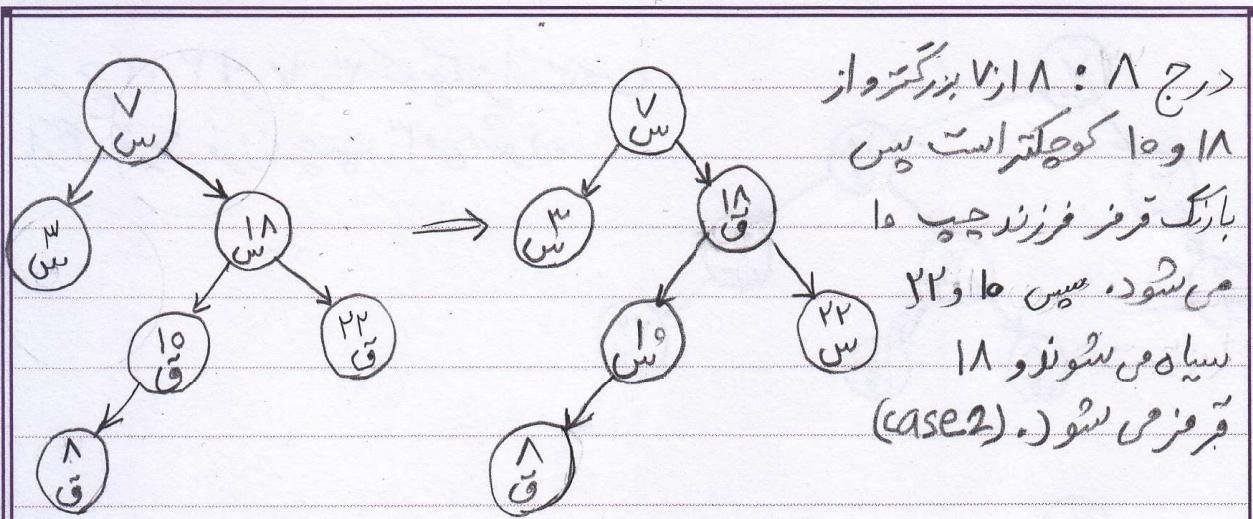
۱۰ از ۷ بزرگتر و از ۱۸ کوچکتر است پس فرزند چپ ۱۸ می‌سود و رنگ سیاه قرفزی می‌سود.  
سپس رنگ ۱۰ و ۳ سیاه می‌سود (Case 2) و سپس ۷ قرفزی می‌سود ولی دوباره سیاه می‌سود (Case 1).



درج ۲۳: ۲۳ از ۷ و ۱۸ بزرگتر است پس فرزند راست ۱۸ می‌سود.  
ورنگ سیاه قرفزی می‌سود.

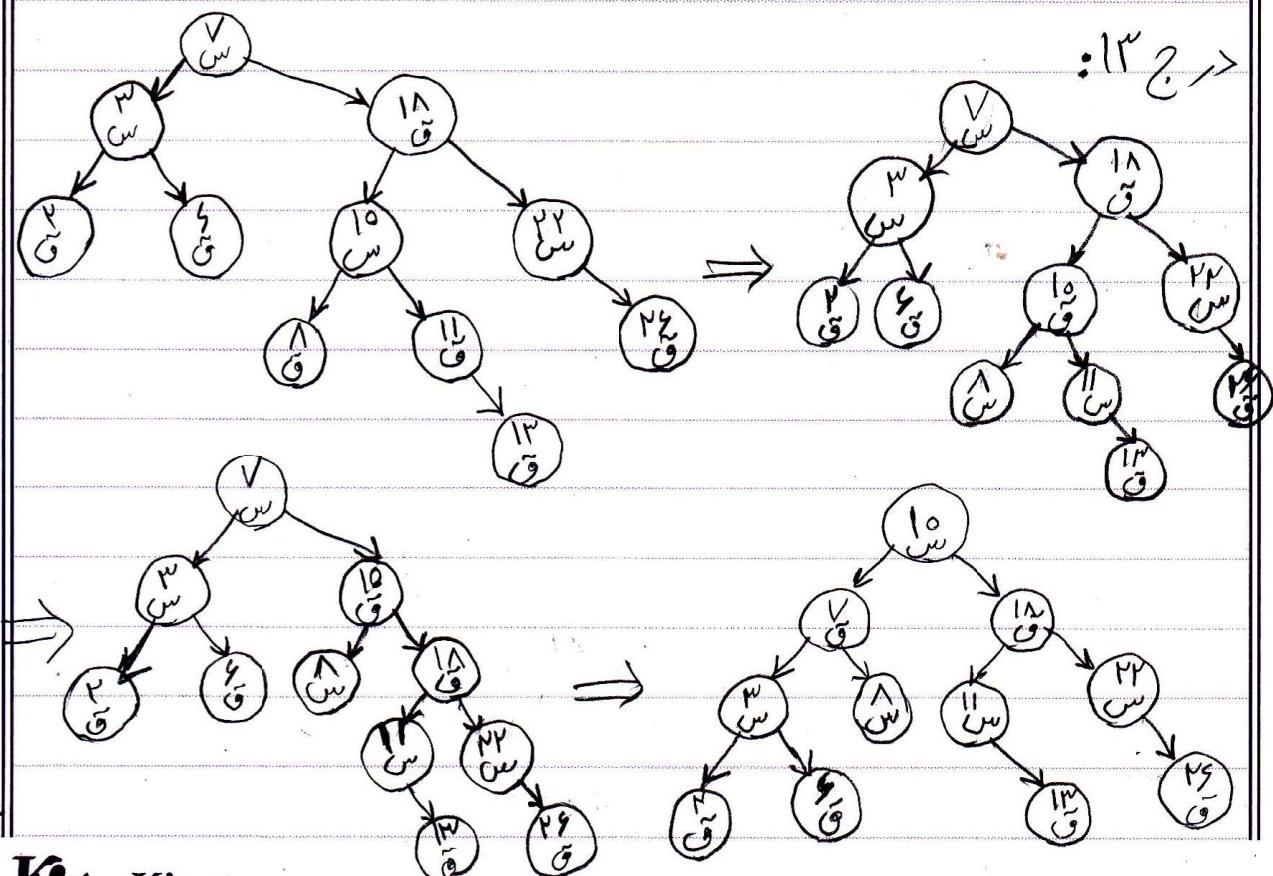
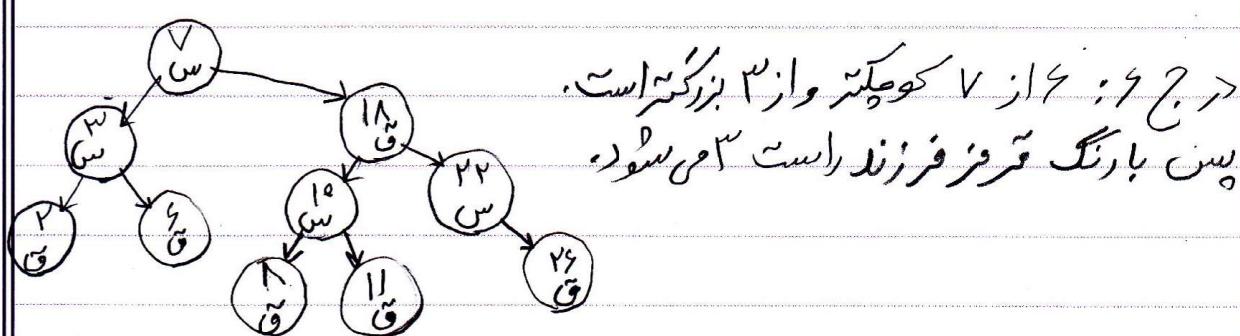
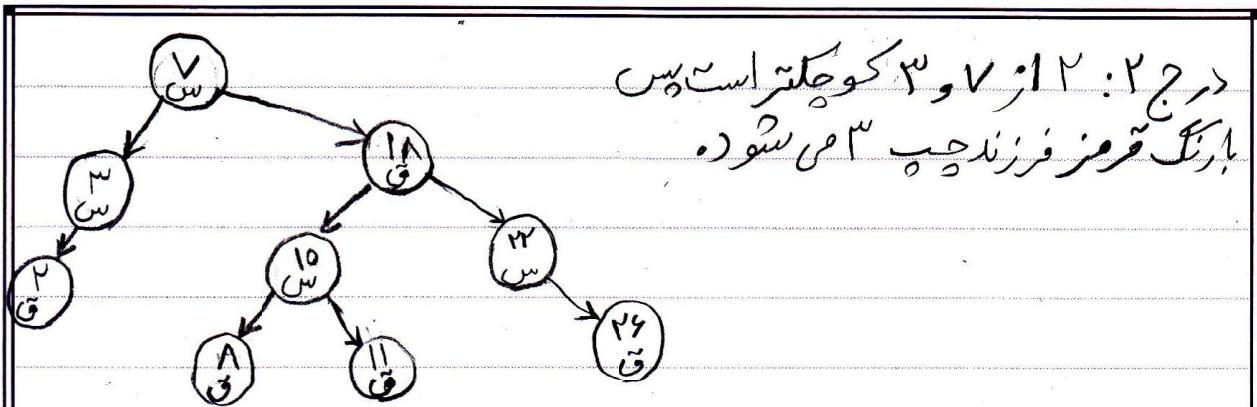
Date:

Subject:



Date:

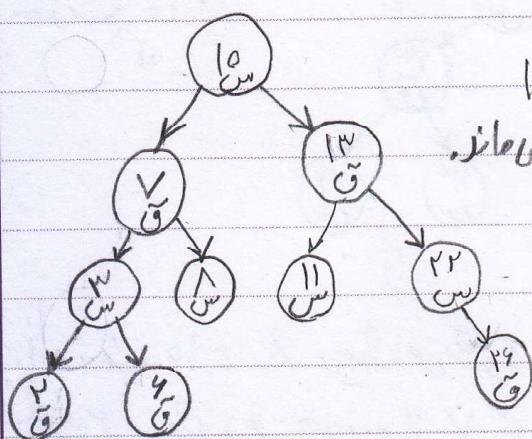
Subject:



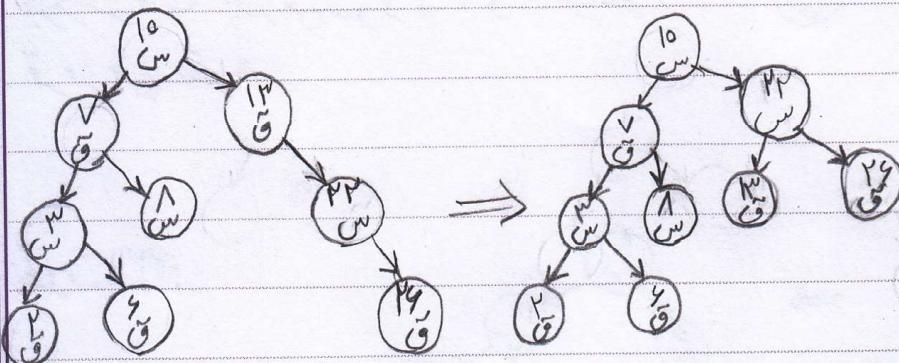
Date:

Subject:

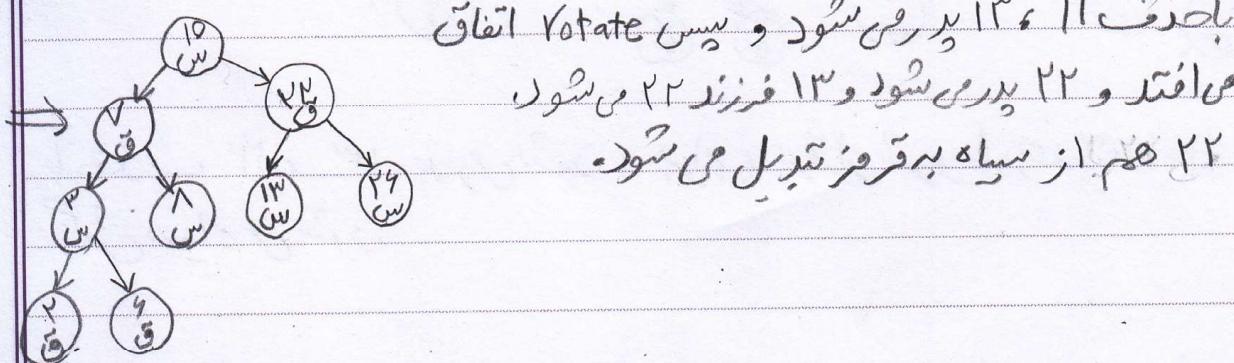
۱۳ از ۷ بزرگتر و از ۱۸ کوچکتر و از ۱۰ و ۱۱ بزرگتر است پس فرزند راست ۱۱  
با نگ قوه‌ی سود پس رنگ ۱۱ او سیاه و نگ ۱۰ قرمز سود (Case 2).  
چون ۱۰ و ۱۱ قوه‌ی آبیارانه rotate انجام می‌شود و پس مارینه و سیاه می‌شوند (Case 2).  
و اینجات کوچکتر بودن ۱۰ و بزرگتر بودن ۱۱، از ۱۰ جدا و فرزند راست آنها  
و ۱۱ قوه‌ی سود



۱۱ حذف: با حذف ۱۱، ۱۳ که از ۱۰  
بزرگتر است پدر ۱۱ و ۲۲ من سود و زنگ قوه‌ی هماند.



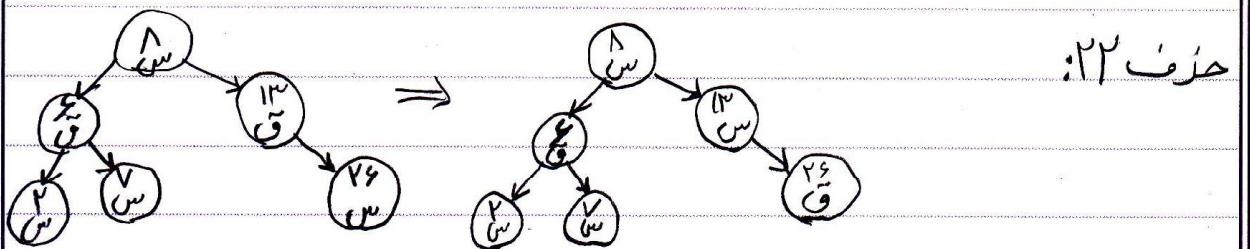
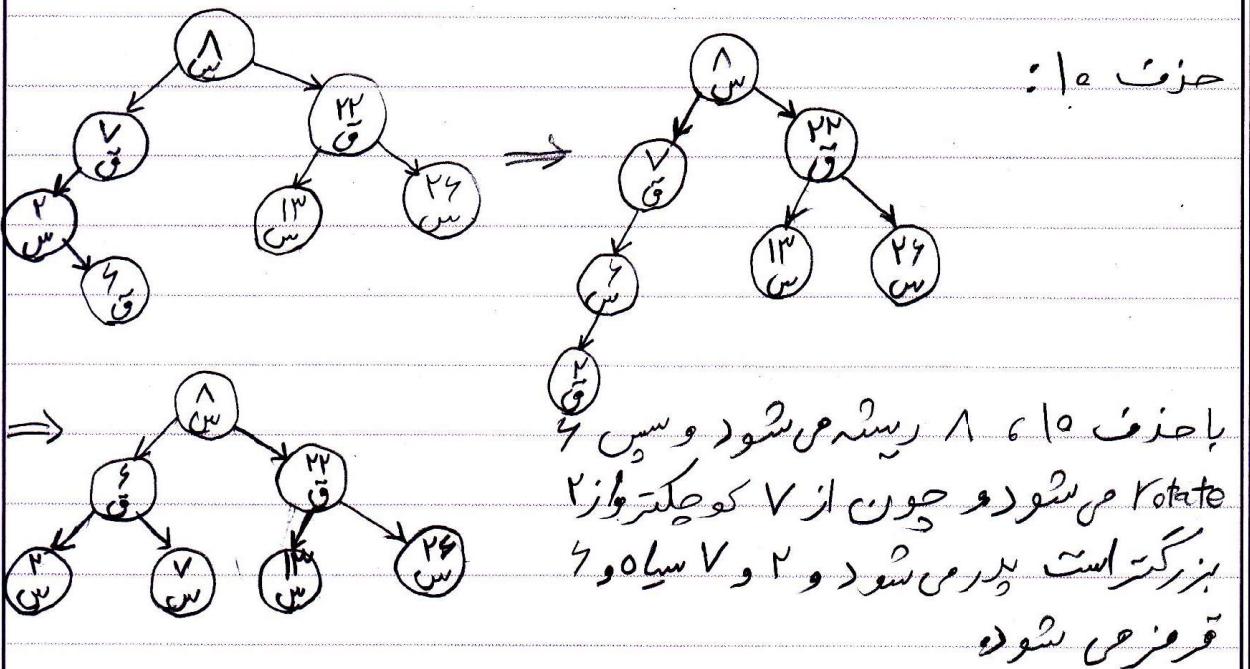
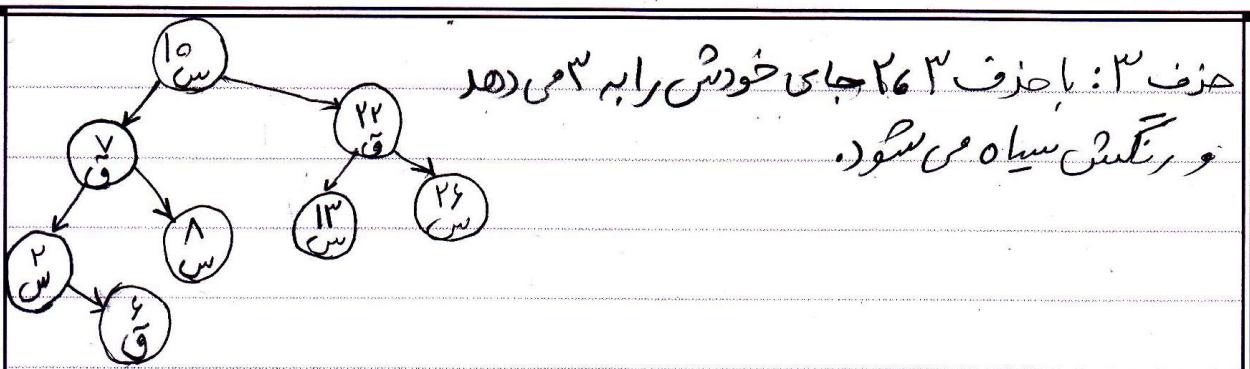
حذف ۱۱:



با حذف ۱۱، ۱۳ پدر سود و پس rotate اتفاق  
می‌افتد و ۲۲ پدر سود و ۱۳ فرزند ۲۲ من سود  
۲۲ هم از سیاه به قوه‌ی تبدیل می‌شود.

Date:

Subject:



س۹: فرض کنید درخت  $n$  رأس دارد و  $K$  کوتاه ترین مسافت بین  
برای رسیدن به  $NULL$  از  $Root$  می باشد. این مسافت است که  
 $K \leq \log_2(n+1) \Rightarrow n \geq 2^K - 1$

می بینیم مسافت بین  $Root$  و  $NULL$  حداقل  $K$  درجه برابر نزدیک ترین  
 $NULL$  می باشد زیرا تعداد رأس های black در صورت آن برابر با  
 تردیک ترین  $NULL$  است و تعداد رأس های red کمتر از تردیک ترین  
 رأس های black می باشد. لذا گزینه قویتر مجاور خواهد بود (داشته)

بنابراین ارتفاع درخت نیز حداقل  $\log_2(n+1)$  خواهد بود.

کندلر برگری RBT بر درخت های BST معمولی هم باشد. چون  
 عملیات های داده ای از  $O(n)$  می باشد و اما در BST معمولی ممکن است  
 $O(n^2)$  باشد.