

-2 حین داشتن جودنداز

درست داشته باشید درست binary search tree باشید باز هم اینکه تغییر مساله والدراست
نه حالکن نیز نیم، متوجه این اینکه هر bst قابل تبدیل به bst دیگر باشان عنصرهاست
و این کار با تبدیل درست از rotation ممکن است.

ایشان با استفاده از استقرای:

base step: برای $n = 2$ bst نهاده کرد و نشان داده شد که هر دو تابع پیلی با هم
همتنی دارند سپس سطر طبله بجزئی از استقرای

inductive step: $n \times n \dots n$ درست بود پس $n+1$ نیز درست باشد
نیز این مفهوم برای الاست.

این اثبات را نیم با درنظر گرفتن هر دو تابع bst، چنین می توانیم با اعمال شادی \rightarrow
rotation را بایس نماییم، با استفاده از الگوریتم زیره

IDEA

(1)

while ($\text{node} \neq \text{root}$)
 if (node is a left child)
 apply a right-rotation to node and node.p
 else
 apply a left-rotation to node and node.p
 proof : با مر بار rotate چرخه می‌شود، اتفاق نیز زمانی ممکن است نا اعمال شناسناس است من تراجم
 برای شرح می‌بریم

حال آن‌گه ابتدا root دستورات چهار یعنی روت اد کن اینست، آن‌را با همین الگوریتم به اینجا
 برخانیم، bst ای خواهیم داشت که دستورات همان یعنی روت اد است، زیرا وقت سنت راست
 دستورات ای همان عکس نیز راست دستورات دوم را داده و علی‌با شکل تغییرات اینجا طوری بیان نیز نمی‌شود.

حال برای نیزه‌چرت (الست و دستورات ای) الگوریتم را به صورت بازگشت اجرا
 می‌کنیم کاربردیت بالاتر دوم بررسی، ازان‌گاهی نیزه‌چرت همی داشت چهار بیشتر از ۱۰
 گره خواهد بود، درینجیه با توجه به فرض مانند step inductive می‌دانیم اینجا
 این الگوریتم حمل می‌کند، نتیجه می‌توانیم بـ «الست دیگر بررسی»

نتیجتاً جوب مسند انسا باشد که برای درخت خاله‌های دوستی داشتی بازگشت شده بیا
 من سود

۲-

Inorder - without recursion (r , curr, T) - ۳
 1. if ($r \neq \text{null}$) then ↑ root node
↓ node
 2. (O1) stack = new stack
 3. curr $\leftarrow r$
 4. while ($\text{curr} \neq \text{null}$ or stack.size > 0)
 5. (O1) { white ($\text{curr} \neq \text{null}$)
stack.push (curr)
curr $\leftarrow \text{curr}.left$ } نام‌گذاری
با این‌قدر
 6. (O1) { curr $\leftarrow \text{stack.pop}$
print (curr.key) } push c stack
نام‌گذاری
با این‌قدر
 7. curr = null
 8. (O1) { curr $\leftarrow \text{curr.right}$ } ذم رفت چب بررسی شده طبق
ذم رفت راست را بررسی می‌کنم.



Subject:

Time complexity = $O(n)$ - proof =

هر کوہ یہ بار بروز من شود یعنی در push ، stack هر کوہ کو شود که هر کوہ کو شود print ، pop هر کوہ کو شود آن کو شود باید اسی سے متناسب باشد. سب سے توان گشت پڑا گھر کوہ کی قدرانسان تابع هر کوہ کو شود ($O(n)$)

$T(n) \in O(n)$

* درست است آن دو توالی while داری که از طرف دیگر تغیر نمی کند و دو توالی داری دارد. دو توالی حالت بین اندیزه هی کو کو شود دو توالی حالت بین اندیزه هی کو کو شود دارند. دو توالی حالت بین اندیزه هی کو کو شود دارند. دو توالی حالت بین اندیزه هی کو کو شود دارند. زمان برنامه $O(n)$ است.

→ Root node

PostOrder - without Recursion (T , n , curr)

1. if($n \neq \text{null}$) then \downarrow , node
2. $\text{stack} = \text{new Stack};$
3. do
4. while ($r \neq \text{null}$)
5. if ($r.\text{right} \neq \text{null}$) $\left\{ \begin{array}{l} \text{push } r.\text{right} \\ \text{push } r \end{array} \right.$ اولی خوبی ایسا کو شود
6. $\text{O}(1)$ دو سبب روشنی را دارد
7. $r \leftarrow r.\text{left}$ کو stack
8. $r \leftarrow \text{stack.pop}$
9. $\text{O}(1)$ آخر کوچک کو pop خوبی
10. if ($r.\text{right} \neq \text{null}$ & $\text{stack.top} == r.\text{right}$) \downarrow آخر کوچک کو pop خوبی
11. $\text{O}(1)$ دوسرا دارکن صفحه بروز
12. stack.pop آخر کوچک کو pop خوبی
13. $\text{stack.push}(r)$ دوسرا دارکن صفحه بروز
14. $r \leftarrow r.\text{right}$ شروع
15. else $\left[\begin{array}{l} \text{print}(r.\text{key}) \\ r \leftarrow \text{null} \end{array} \right]$
16. while ($\text{stack.top} \neq \text{null}$)

IDEA

(3)

Subject:

Time Complexity: $\in O(n)$ - proof:

است الـ traversal هم الاست زیر الـ recursive algorithm با يار تکمیل تریو هار برسی اینم جتنا
تریسی برسی درون مقطاعات الاست که مکن است overheard ایجاد شد، نه یا ثابت است،
تائیدی نهاد رهان (An) سالند

در این قسم خطوط مخفی شده هزینه‌ی ثابت داشته و بینه ب این‌جهی تعداد طبقه
من ۱ بار تکرار مسحود while داشته و چونی cur بینه داشته مایل‌به‌جستجوی
دره ساده بیس دیگرینی حالت ب این‌جهی دره که رهان شود

$| T(n) \in O(n) |$

PreOrder - Without - Recursion (T, r) $\xrightarrow{\text{root node}}$

1. if ($r \neq \text{null}$)

2. stack = new stack } $O(1)$

3. stack.push(r)

4. while (stack.top != null)

5. curr = stack.top

6. print (curr) } $O(1)$

7. stack.pop

8. if (curr.right != null) } $O(1)$

9. stack.push(curr.right)

10. if (curr.left != null) } $O(1)$

11. stack.push(curr.left)

برای هر کدام pop می‌شود
ایدها print می‌شوند
فرزند را استاد سمسز زنند
آنرا push می‌کنیم تا آنرا خروجی
زنند برسی می‌کنیم
جستجو شود.

Time Complexity: $O(n)$ proof:

ترمیمات و چیزی می‌توانیم جستجو کنیم. هرگره یک بار برسی می‌شود، با این معنی

هزینه‌ی ثابت دارد. اینجا هم کاملاً با یک درست‌حالت خود push شویز و www

تا برسی می‌شود، while می‌باشد، یک بار تکرار مسحود

$| T(n) \in O(n) |$



(4)

۴- الف) درخت جستجوی متعادن درخت T_1, T_2 : r_1, r_2 برای کدام از دو خانه درخت $T_1 = T_2$ درخت T_1 بود؟
 T_1 را با r_1' نیز search . T_2 را با r_2' دوخت . $r_1' \neq r_2'$ درخت T_1 را درخت T_2 نمایش دهد .
 tree 1 root . tree 2 root .

Check - version1(T_1, T_2, r_1, r_2 , curr) inorder \leftarrow ترتیب تابعی
 L, R root

1. if ($r_1 \neq \text{null}$ & $r_2 \neq \text{null}$)

2. stack = new Stack

3. curr $\leftarrow r_1$

4. while (curr $\neq \text{null}$ or stack.size() > 0)

5. while (curr $\neq \text{null}$)

6. stack.push(curr)

7. curr \leftarrow curr.left

8. curr \leftarrow stack.pop

9. if (Tree-search($r_1, curr, key$), $\neq \text{null}$)
 print(yes)

10. else print(no)

11. curr \leftarrow curr.right

root

Tree - search (x, k)

1. if $x == \text{null}$ or $k == x.\text{key}$ then

2. return x

3. if $k < x.\text{key}$ then

4. return Tree-search($x.\text{left}, k$)

5. else return Tree-search($x.\text{right}, k$)

Time Complexity : \leftarrow جستجوی $O(n)$ ، \leftarrow جستجوی $O(1)$ جستجوی $O(n)$

$$h = h_x = \lg n_x \quad \leftarrow \text{جستجوی } O(n)$$

$$n = n_1$$

IDEA

(5)

Subject:

ب) ابتدا مرتب tree را می‌شیس inorder تا کلمه در درام لا را تک آرایه خصوصی کنیم.

پس نیاز به $O(n+r)$ حافظه اضافی داریم. تک آرایه برای اعشاری وقت آدمی برای

اعشاری وقت T_r . پس از آنکه در اینجا همان خصوصی کردیم، با استفاده از حکم درون اینکه آیا تک آرایه

تک آرایه هست یا نه $S_1 \subseteq S_2$ می‌شود \Rightarrow subset

Time complexity :

با توجه به هزینه زمان هایی که در هم می‌گذاریم و باسته باید تردد $O(nr) = O(n+r)$ زمان

می‌برند و حکم درون اینکه آیا $S_1 \subseteq S_2$ هست بالسته ای $O(n+r)$ زمان بعد

$\Rightarrow \text{Time} = O(n+r)$

extra space = $O(n+r)$

محصول دو آرایه با این ارزشها بترتیب n_1, n_2 داریم:

check - version2 (T_1, Tr, r_1, r_T)

int i, j = 0

Array $S_1[n]$

Array $S_T[n_T]$

(\leftarrow if $r_1 \neq null$)

if ($r_1 \neq null$) then

Stack stack1 = new Stack

curr1 $\leftarrow r_1$

while ($curr1 \neq null$ or $stack1.size() > 0$)

while ($curr1 \neq null$)

[$stack1.push(curr1)$

[$curr1 \leftarrow curr1.left$

$curr1 \leftarrow stack1.pop$

$S_1[i] \leftarrow curr1$

$curr1 \leftarrow curr1.right$

$i++$

(Iterative).

\checkmark inorder traversal

T_1 tree intact is by

S_1 ذخیره داده شد

$O(n_1)$

if ($r_T \neq null$) then

Stack stack2 = new Stack

curr2 $\leftarrow r_T$

while ($curr2 \neq null$ or $stack2.size() > 0$)

while ($curr2 \neq null$)

[$stack2.push(curr2)$

[$curr2 \leftarrow curr2.left$

$curr2 \leftarrow stack2.pop$

$S_T[j] \leftarrow curr2$

$curr2 \leftarrow curr2.right$

$j++$

(Iterative)

\checkmark inorder traversal

T_T tree intact is by

S_T ذخیره داده شد

$O(n_T)$

map<int, int> map;

for $i = 0$ to n_1

[$map[S_1[i]]++$

for $i = 0$ to n_T

[if $map[S_T[i]] > 0$

[[$map[S_T[i]]--$

else

[return false

return true

$\forall i \in S_1$ $C_{S_1}^i \leq n_1$

$\forall i \in S_T$ $C_{S_T}^i \leq n_T$

$O(n_1 + n_T)$

IDEA

(6')

- a) الگوریتم الائچه نماین و در مرتبه زمانی $O(n)$ برسی کرده آیا دوست دوستی حیثیت
همسی پایینی.

با استفاده از سیسی morder میتوانیم (هر چیزی که از ترتیب جعلی خود است
یا نه؛ از قدرت بودن) دست نی باشد.

check-BST (Tree, r)

↳ root node

1. if ($r \neq \text{null}$)
2. int prev = Integer-Min-Value → لکتیون - عبارتی که را در ابتدا برای prev
3. if (check-BST (Tree, r.left) \neq true)
4. return false
5. if (r.key \leq prev)
6. return false
7. prev = r.key
8. return (check-BST (Tree, r.right))
9. return true → آن دوست خالی است

IDEA

(6)

Time Complexity:

worst case زمان زیست سکن وقت BST با شرط زیر ایام تابعی کروها

که سودن ب عواین شکل در وقت اینجا از ترقی ۱۰ شروع ب

حد درین سایم، یکبارهای نیز وقت حب به قدر ترسید چک

س گنم / داگر برای ایندو هر تراستام میتوود، دلخون بزیر

حالت است ادایم در چشم داین میلیات بازیش ادامه دارد

تابن بازیش حد درین شرط root.key < prev برسی نهفته دار، تابسته دارد

$$\Rightarrow T(n) = O(1) \times n \in O(n)$$

- ۴ دیالیکیش \leftarrow post order \rightarrow inorder \rightarrow root دویس را میان

Inorder traversal = D, H, B, E, A, F, C, G
root = A

Postorder traversal = H, D, E, B, F, G, C, A
root = A

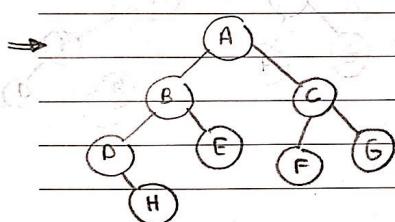
inorder: left, root, right postorder: left, right, root

اصلی دیگر : درینه درانه وجود دار

اصلی دیگر : عناصر نیز وقت حب، (استخدم اینده زنسته شده از درجه همیش

F, G, C بازیجنه را اصلی دیگر : عناصری وقت حب

D, H, B, E : عناصری وقت راست



از آن جای پستوردر آن آنده، در در inorder آن دیگر ؟ من D / اولینین عین اس

و H باین فرزند راست آن باشند پستوردر آن در اول خالص شد. (با عجم به ترسید چک

IDEA

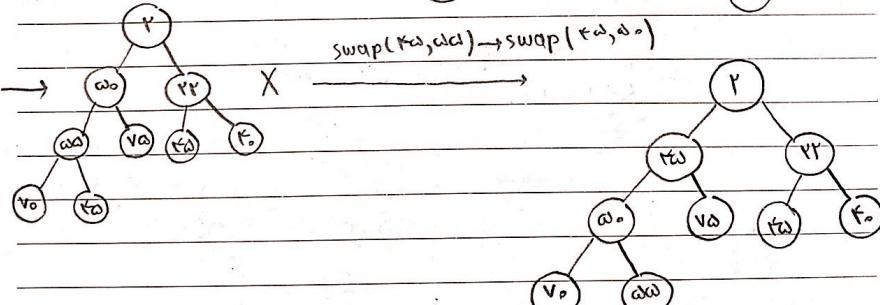
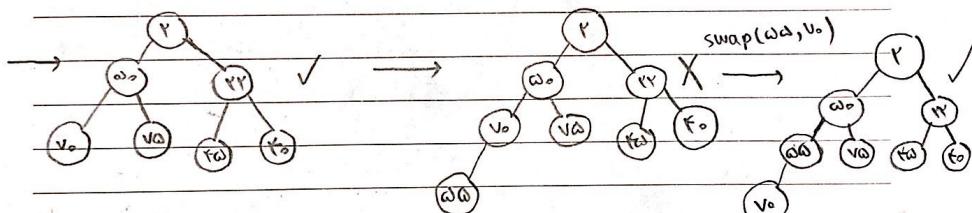
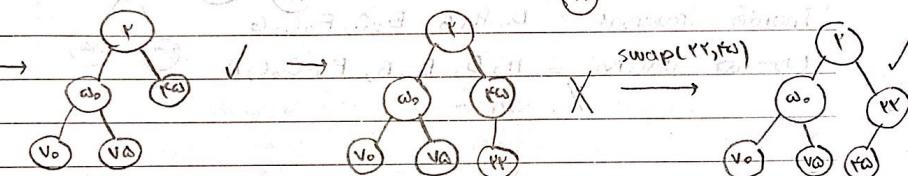
Subject:

۷ با درج تعداد زیر ب ترتیب یک min Heap بسازید.

2, 50, 45, 70, 75, 22, 40, 55, 45

تجزیه شد اعداد ب ترتیب والاس شرند و فقط یک دوست است این در ترتیب بسازید.

Array: [2 | 50 | 45 | 70 | 75 | 22 | 40 | 55 | 45]



IDEA

(8)

Subject:

- بحث زیر را به ترتیب دارا داشت case های را در مرحله ترقیح (های) (ادن بار در ۷ دارم شود)

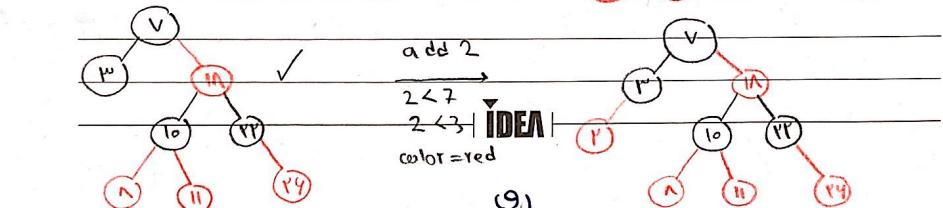
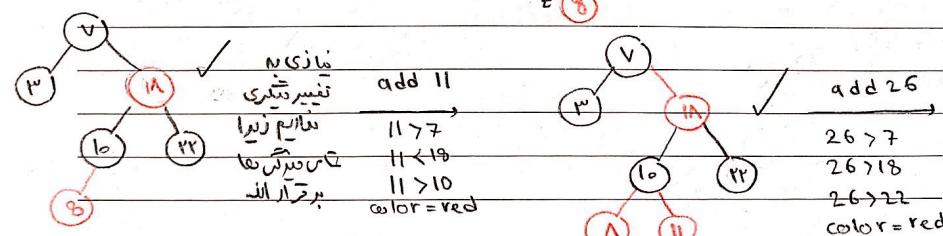
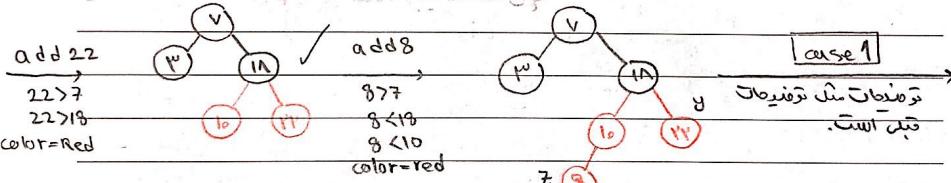
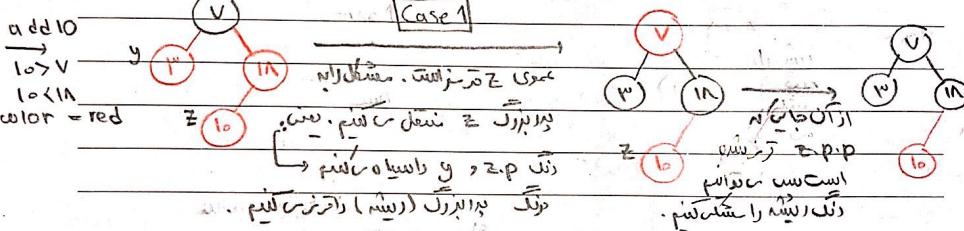
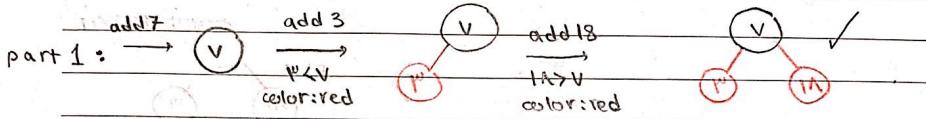
part 1:

$7, 3, 18, 10, 22, 8, 11, 26, 2, 6, 13$

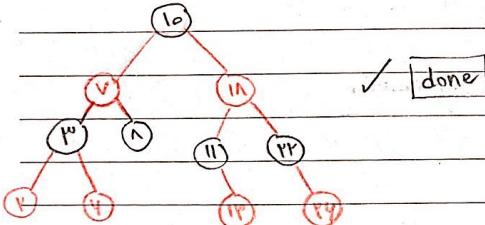
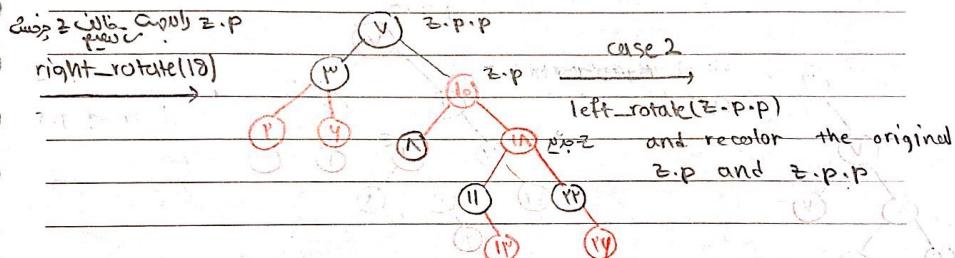
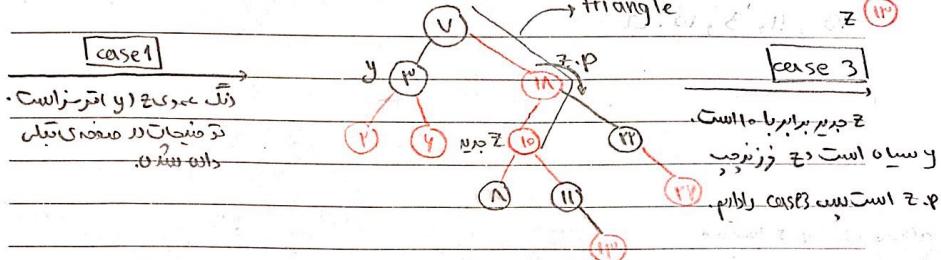
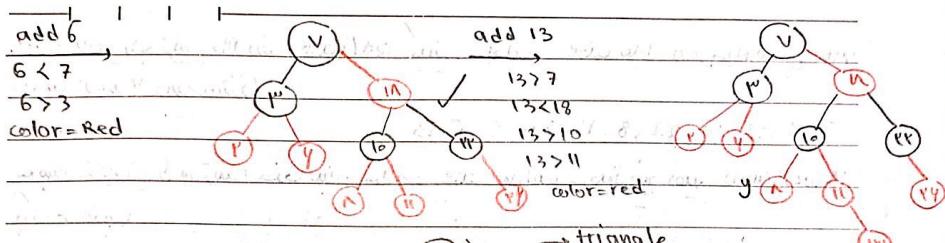
سپس بحث زیر را به ترتیب حذف کنید. مراده های را در مرحله ترقیح دهید. اولین بار در ۱۸ پاک شود

part 2:

$18, 11, 3, 10, 22$



Subject:



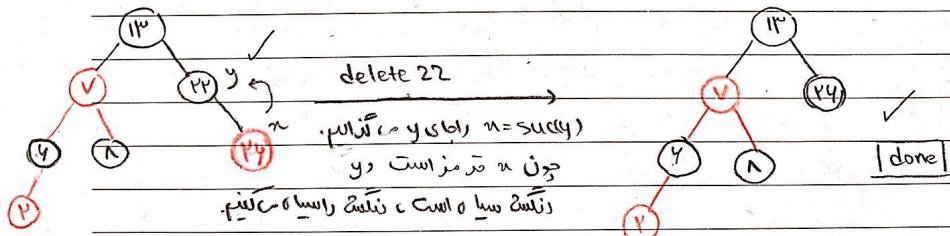
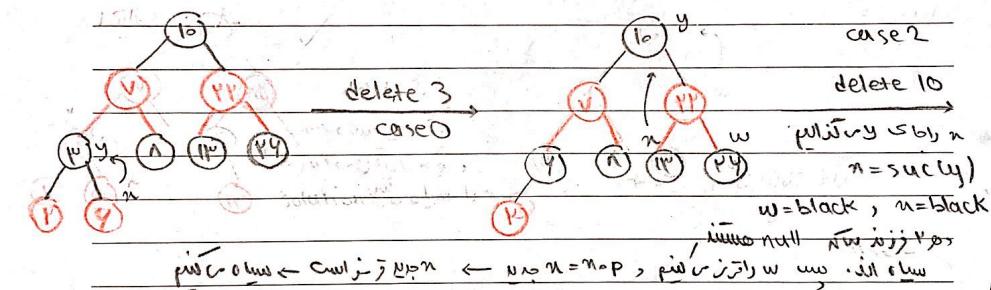
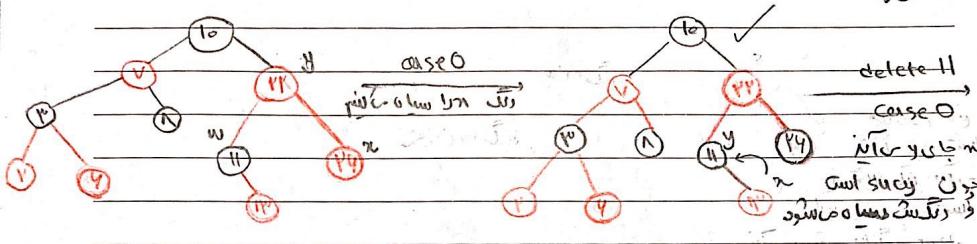
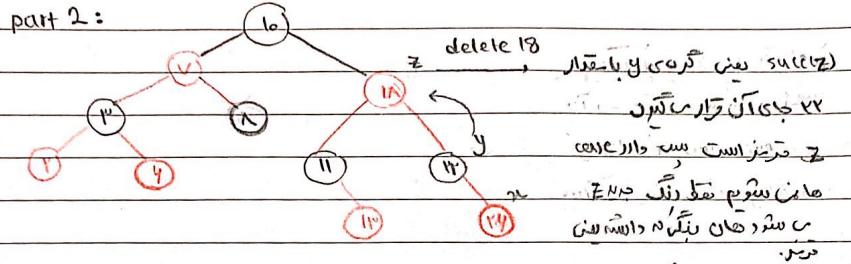
IDEA

(10)

Subject:

case 4: حسابات جمع بدلی کارهای ساده از.

part 2:



IDEA

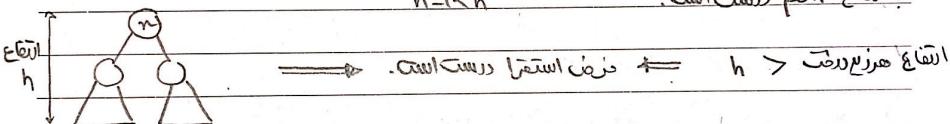
(11)

برگ‌ها

امتحانی : ۹ - $bh(n) = \sqrt{n}$ سیاه بجز خود را از $bh(n-1)$ در مدخل طارد
اول مخاطبین ثابت نمایند : " هر زیردرخت باریش و حداقل $bh(n)$ نمایند " استواری تردید)

بالستفاده از استقرآیی : آنکه $= h(n) = 0$ null است. \leftarrow $bh(n) = 1 = 1 - 1 = 0$ \checkmark \leftarrow
با اینی استقرآیی ثابت شد که $bh(n) = 0$ \leftarrow

حال خونه سیاه این ادعای برای هر زیردرخت با ارتفاع $n-1$ درست است. با این ثابت نیم باید نیز درست با ارتفاع هم درست است.



آخر ارتفاع سیاه n $bh(n)$ ارتفاع سیاه نمی‌تواند بزرگتر از $bh(n)-1$ باشد. (ب زنگ زید درخت هاستگه طالع)

آخر ارتفاع نمایند $bh(n-1)$ $bh(n)-1$ \leftarrow
 $bh(n) = bh(n-1) + 1$ \leftarrow می‌باشد
 $bh(n) = bh(n)$ \leftarrow می‌باشد " "

| IDEA |

(12)

بنابر این اتفاقاً سیاه و خردلی است. بنابراین سوارگوهای داخل زیر درخت با لیستی دارند. $bh(n) - 1$ است.

و زیر درخت داریوس سوارگوهای مغلوب نیز درخت با لیستی n است.

$$\begin{aligned} & \text{سوارگوهای داخل } \gg (v^{bh(n)-1}) + (v^{bh(n)-1}-1) + 1 \\ & \text{زیر درخت با لیستی } \gg v^{bh(n)} - 1 \quad \checkmark \quad [\text{ایجاد شده}] \end{aligned}$$

لهم سیم از رسیم تابه که خداوند را بسیار خوبی دارد.
 $bh(T) \geq h(T)$ \Rightarrow سوارگوهای داخل

$$\begin{aligned} & \xrightarrow{\text{طبقه بندی}} n \geq v^{bh(T)} - 1 \geq v^{\frac{h(T)}{v}} - 1 \\ & \rightarrow n+1 \geq v^{\frac{h(T)}{v}} \rightarrow \log_v(n+1) \geq \frac{h(T)}{v} \rightarrow v \log_v(n+1) \geq h(T) \\ & \Rightarrow h(T) \leq v \log_v(n+1) \end{aligned}$$

* حلست برتری:
 درخت rb، زیر درخت نسبتی میانی است با وجود به $h(T)$ بستگی که میانه های مختلف نباشد.
 $v \log_v(n+1)$ در آن اینجا مسوند نه نسبتی زمان حذف است.
 (.Cwl self-balancing tree) Cwl هم این Cwl باقی بودند rb tree میانه ای اعمال می شوند.
 همینکه نگاهش خواهد داشت. نسبتی

IDEA

(13)