پاسخ تمرین شماره ۳
Tree

دانشکده مهندسی برق و کامپیوتر	**مهلت تحویل:** **۱۹ آذر**	**ساختمان های داده و الگوریتم - پاییز ۱۴۰۱**
استاد: دکتر فتحیه فقیه، دکتر هشام فیلی	**۱۹/۹/۱۴۰۱، ساعت ۲۳:۵۹**	**طراحان تمرین : علی هدائی و حامد میرامیرخانی**

---

۱. به سوالات زیر که درخصوص یک درخت دودویی است پاسخ دهید.

الف) درخت مورد نظر را با استفاده از دو پیمایش داده شده رسم کنید.

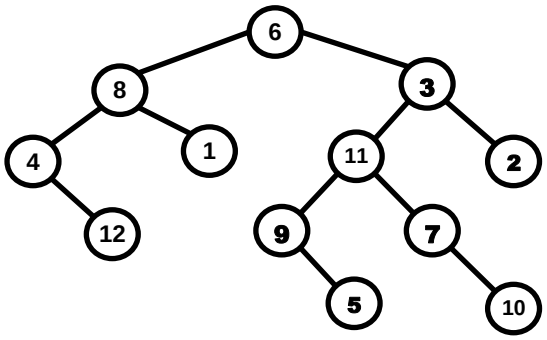
postorder: 12,4,1,8,5,9,10,7,11,2,3,6

inorder: 4,12,8,1,6,9,5,11,7,10,3,2

ب) در قسمت قبل دیدید با در اختیار داشتن نمایش postorder و inorder میتوان به یک درختِ واحد رسید. بنظرتان با داشتن دو پیمایش preorder وinorder و یا جفت پیمایش preorder و postorder نیز میتوان به یک درخت واحد رسید؟ برای هر حالت اگر میتوان این کار را کرد مختصرا روش کار را توضیح بدهید و اگر نمیتوان مثال نقض بیاوردید.

پاسخ.

الف)



ب)

Inorder و Preorder)

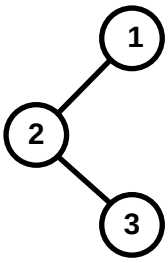
بله میتوان شکل درخت را رسم کرد. روش کار هم اینگونه خواهد بود که ابتدا به پیمایش پیش ترتیب نگاه میکنیم، اولین راس در این پیمایش نشان دهنده ریشه درخت است. در این صورت در پیمایش میان ترتیب همه راس های موجود در سمت چپ این راس در زیردرخت چپ ریشه قرار دارند و بالعکس. این عمل را تکرار میکنیم(راس دوم در نمایش پیش ترتیب را نگاه میکنیم، این راس ریشه ی زیردرخت چپ خواهد بود و...) تا درنهایت به شکل کامل درخت دست پیدا کنیم.

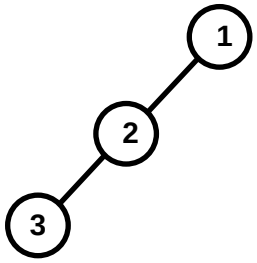
Preorder و Postorder)

خیر نمیتوان دقیقا مشخص کرد شکل درخت ما چگونه است. مثال زیر را ببینید:

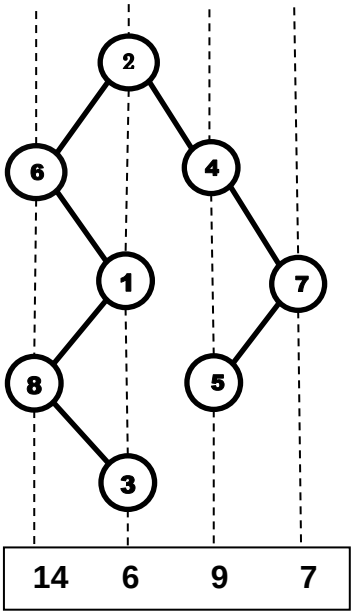
Postorder = 3,2,1

Preorder = 1,2,3





۲. درخت دودویی T با n رأس در اختیار داریم که راس های آن ها از 1 تا n شماره گذاری شده اند. تابع چگالی ستون در این درخت معادل است با مجموع شماره راس هایی که روی یک خط عمودی قرار گرفته اند. الگوریتمی با مرتبه زمانی O(n) پیدا کنید که بیشینه مقدار این تابع را در یک درخت پیدا کند.(به عنوان مثال در شکل زیر، مقدار این تابع برای ستون های مختلف نوشته شده است و ماکسیمم آن ۱۴ است)



پاسخ.

الگوریتم: ابتدا یک لینک لیست دوطرفه در نظر میگیریم که تنها شامل ریشه درخت است. حال روی درخت یک پیمایش پیش ترتیب میزنیم و در هر مرحله از پیمایش درخت، اگر به فرزند چپ رفتیم باید در لینک لیست به نود قبلی برویم و به اندازه شماره راس به مقدار آن نود در لینک لیست اضافه کنیم، بهمین ترتیب اگر به فرزند راست رفتیم به نود بعدی در لینک لیست میرویم و این مقدار را اضافه میکنیم. اگر در طی این فرآیند نود قبلی یا بعدی ای وجود نداشت یک نود میسازیم و مقدار آن را برابر با شماره راس پیمایش شده قرار میدهیم. در پایان پیمایش کافی است روی لینک لیست یک بار حلقه بزنیم و بیشترین مقدار را برگردانیم.

نیازی به نوشتن شبه کد نیست و صرفا برای درک بهتر به آن توجه کنید:

```
func(vertex, node){
  node.value += value(vertex)
// left taken
  if(left(vertex) exists){
    if(pre(node) does not exist)
      pre(node) = new node(0)
      pre(node).set_next(node)
    }
    func(left(vertex), pre(node))
// right taken
  if(right(vertex) exists)
    if(next(node) does not exist){
      next(node) = new node(0)
      next(node).set_pre(node)
    }
    func(right(vertex), next(node))
}
```

۳. آرایه ای از اعداد صحیح (نه لزوما مثبت) در اختیار داریم. از بین زیردنباله های متوالی در این آرایه، میخواهیم **m**اُمین زیردنباله از لحاظ **بزرگی مجموع عناصر** را پیدا کنیم. از **heap** کمک بگیرید و الگوریتمی در زمان **$O(n^2 \log(m))$** ارائه دهید که این زیردنباله را برایمان پیدا کند.

پاسخ.

از روی آرایه ی اولیه آرایه ای به نام **sum** تعریف میکنیم به گونه ای که عنصر **i** اُم آن حاصل جمع **i** عنصر ابتدایی آرایه ما باشد. به این ترتیب با کم کردن عناصر آرایه **sum** از هم، میتوانیم مجموع عناصر هر زیردنباله متوالی را داشته باشیم. حال از یک **min-heap** کمک میگیریم و روی عناصر این آرایه حلقه تو درتو میزنیم و **m** عنصر اول را به **min-heap** اضافه میکنیم. پس از آن تا آخر آرایه پیمایش میکنیم و برای اضافه کردن هر عنصر چک میکنیم که آیا مقدار آن عنصر از **min-heap.top()** بزرگتر است یا خیر، اگر بزرگتر بود **min-heap.top()** را پاپ کرده و عنصر جدید را به هیپ اضافه میکنیم و عملیات **heapify** را انجام میدهیم. در اخر **min-heap.top()** نهایی پاسخ ما خواهد بود.

۴. علی حامد را به چالش کشیده است و از او میخواهد درخت جست و جوی دودویی خاصی را برایش رسم کند. او به حامد دنباله ای از اعداد به طول نامشخص میدهد و حامد هم وظیفه دارد یک درخت جستجوی دودویی رسم کند که درنهایت علی بتواند دنباله موردنظر خود را هنگام جست و جو در آن درخت ببیند. علاوه بر این، او به حامد دو عدد اعلام میکند و قول میدهد اعداد دنباله حتما بین این دو عدد باشند.اما مشکل اینجاست که هر دنباله ای چنین خاصیتی را نمیتواند برآورده کند و این قضیه را نه علی میداند و نه حامد! شما بعنوان نفر سوم به کمک این دو بیایید و الگوریتمی از مرتبه زمانی **$O(n)$** طراحی کنید که تشخیص دهد آیا دنباله ای که علی به حامد میدهد میتواند دنباله ای باشد که در هنگام جستجو در یک درخت جست و جوی دودویی دیده شود یا خیر.(برای مثال اگر اعداد بین ۱ تا ۱۰۰ را در یک درخت دودویی جستجو داشته باشیم دنباله ی $\{۷۰، ۴۸، ۶۱، ۲\}$ نمیتواند دنباله ی درستی باشد)

پاسخ.

به شیوه جست و جو در یک درخت جستجو دودویی دقت کنید، تمام نود هایی که در زیردرخت چپ یک راس قرار میگیرند از آن راس کوچک ترند و بالعکس. پس حین جستجو در این درخت، کران بالا و پایین ما مرحله به مرحله به همدیگر نزدیک تر و محدوده ما کوچکتر میشود. از این ایده میتوانیم برای پیدا کردن الگوریتم استفاده میکنیم.

الگوریتم: ابتدا محدوده مقادیر راس ها را با استفاده از دو عددی که علی میگوید مشخص میکنیم، مثلا بازه ی **(a, b)**. سپس دنباله ی موردنظر را در آرایه **A** ذخیره میکنیم. تک تک عناصر آرایه را بررسی میکنیم و هر عنصر آرایه را با عنصر بعد از خودش مقایسه میکنیم. اگر عنصرفعلی از عنصر بعدی خود کوچکتر بود کران پایین محدوده را آپدیت میکنیم و اگر از عنصر بعدی بزرگتر بود کران بالا را(در واقع این کار معادل با این است که هنگام جست و جو بر اساس مقدار راس ها بین زیردرخت راست یا چپ یکی را انتخاب میکنیم). اگر حین پیمایش آرایه با عددی مواجه شدیم که در محدوده ما قرار نداشت با دنباله ی غلطی رو به رو هستیم در غیر این صورت دنباله دنباله ی صحیحی خواهد بود. بدیهی است که هزینه زمانی الگوریتم هم **O(n)** خواهد بود.

برای درک بهتر الگوریتم شبه کد مقابل را ببینید:

```
for i = 0 to n-1 {
  if(A[i] > max or A[i] < min) {
    return false
  }
  if(A[i+1] > A[i]) {
    min = A[i]
  }
  if(A[i+1] < A[i]) {
    max = A[i]
  }
}
Return true
```

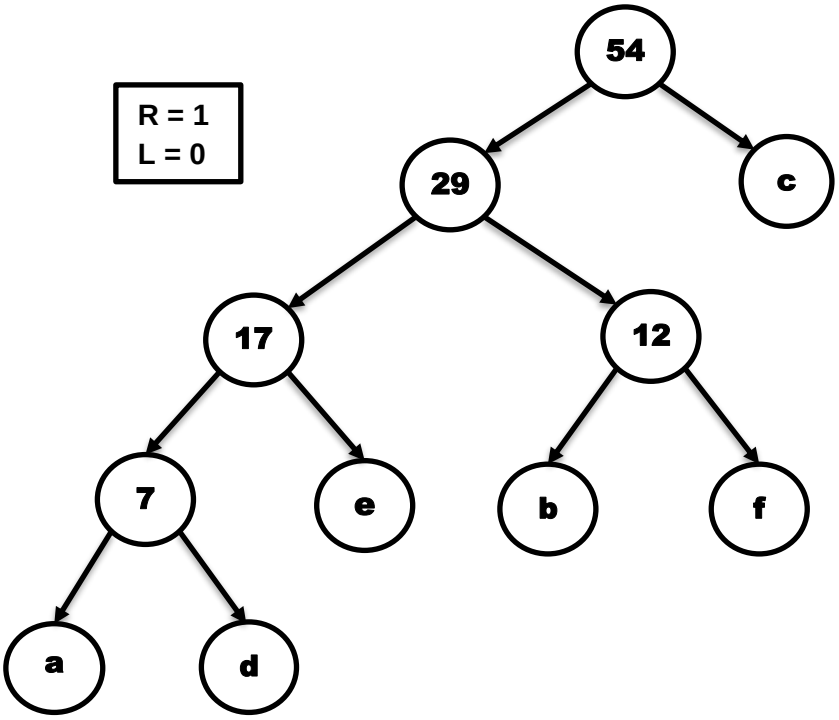
۵. در جدول زیر تعداد دفعات تکرار هر حرف مشخص شده است. درخت هافمن مربوط به آن را بکشید و سپس کلمه "**bdfce**" را در کد باینری متناظر آن نمایش دهید.

تکرار	حرف	تکرار	حرف	تکرار	حرف
۱۰	**e**	۲۵	**c**	۲	**a**
۶	**f**	۵	**d**	۵	**b**

پاسخ.

مطابق درخت رسم شده:

bdfce → 01000010111001



۶. درخصوص درخت قرمز-سیاه به موارد زیر پاسخ دهید.

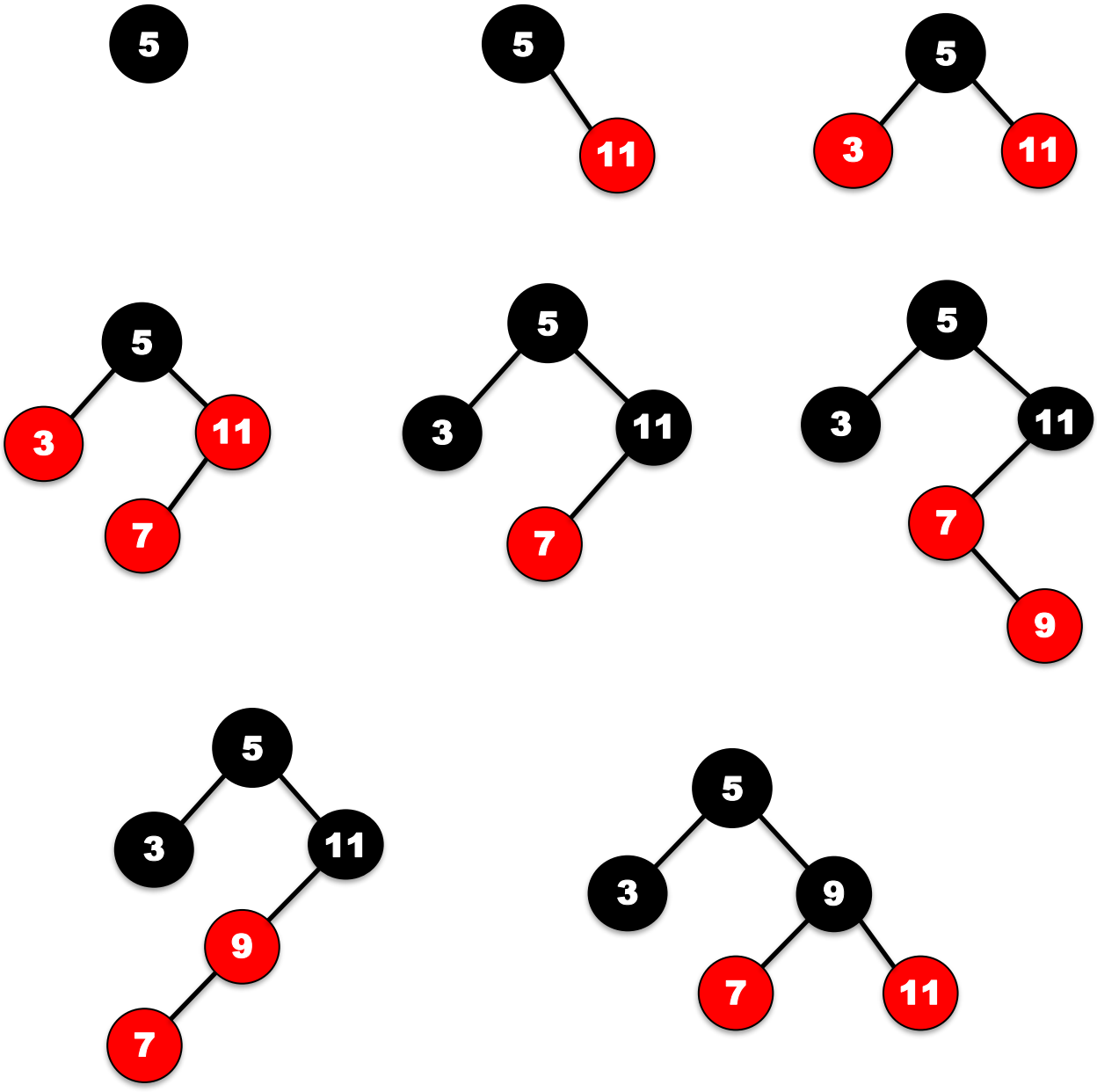
الف) به ترتیب از چپ به راست اعداد **5,11,3,7,9** را در چنین درختی درج کنید.

ب) تصور کنید گره ای مشخص به یک درخت قرمز-سیاه اضافه میکنیم و سپس آن گره را بلافاصله از درخت حذف میکنیم. آیا درخت حاصل با درخت قبلی یکسان است؟ اگر بله اثبات کنید در غیر اینصورت مثال نقض بیاورید.

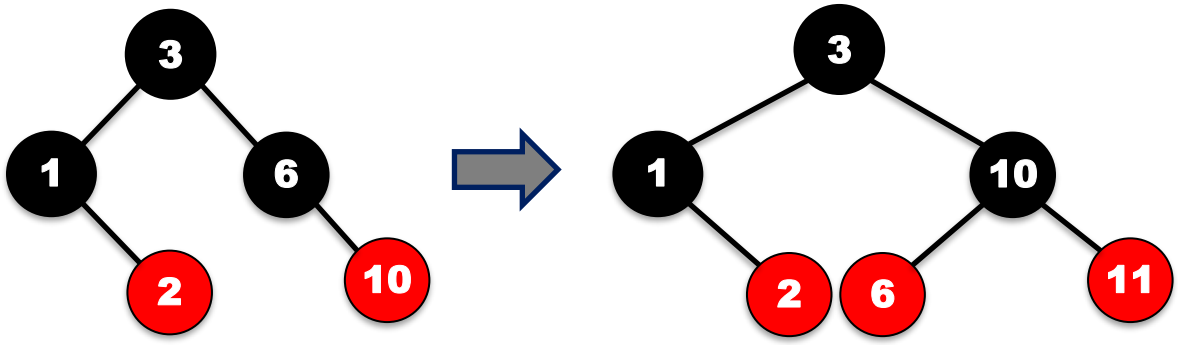
ج) اثبات کنید ارتفاع درخت قرمز و سیاه از مرتبه **O(logn)** است.

پاسخ.

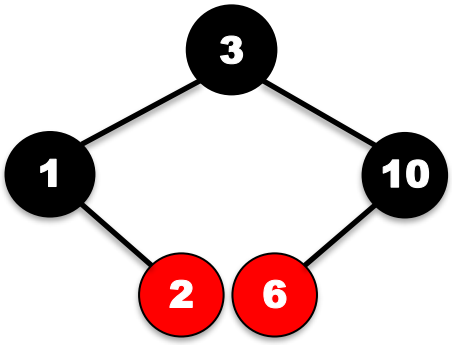
الف)



ب) خیر میتواند یکسان نباشد. مثلا به درخت پایین توجه کنید که به ترتیب اعداد **1,3,6,10,2** به آن اضافه شده اند. سپس به آن گره ای با اندازه ۱۱ را اضافه میکنیم:



حالا بلافاصله همین گره را از درخت حذف میکنیم، درخت زیر بدست می آید که مشخص است با درخت اولیه ما فرق دارد.



ج) میخواهیم برای ارتفاع درخت حد بالایی تعیین کنیم. برای اینکار رئوس قرمز را در رئوس سیاه ادغام میکنیم، درختی حاصل میشود که هر راس غیر برگ در آن ۲ یا ۳ یا ۴ فرزند دارد.

میدانیم در درخت قرمز سیاه تمامی فرزند های یک راس قرمز(در صورت وجود) سیاه هستند پس در هر مسیر از ریشه تا برگ **حداکثر** نصف رئوس قرمز هستند. پس با ادغام کردن رئوس قرمز در سیاه ارتفاع حداکثر ۲ برابر میشود. از طرفی همانطور که گفتیم هر راس غیر برگ در درخت جدید حداقل ۲ فرزند دارد. در نتیجه:

ارتفاع درخت پس از ادغام **h' =** , ارتفاع درخت اولیه **h =** , تعداد رئوس **n =**

$$n \geq 2^{h'} - 1 \Rightarrow \frac{h}{2} \leq h' \leq log(n+1) \Rightarrow h \leq 2log(n+1) \Rightarrow h \in O(logn)$$

۷. یک درخت **avl** داریم که میدانیم ارتفاع آن **h** است. حداقل تعداد رئوسی که این درخت میتواند داشته باشد را برحسب **h** بیان کنید و ادعای خود را ثابت کنید.

پاسخ.

کمترین تعداد رئوس برای ارتفاع **h** طبق رابطه بازگشتی زیر بدست می آید:

Min(h) = Min(h-1) + Min(h-2) + 1

Min(1) = 1, Min(2) = 2

اثبات ادعا: طبق خاصیت درخت**avl** حداکثر اختلاف زیردرخت چپ و راست برابر ۱ واحد است پس حداقل ارتفاع یکی **h-1** و دیگری **h-2** است. پس در نتیجه **min(h)** ما برابر حاصل جمع این تابع برای زیردرخت چپ زیردرخت راست بعلاوه ی خود ریشه خواهد بود. حالا کافی است مقدار این تابع را برحسب **h** بدست آوریم:

$$\boldsymbol{Min(h) = Min(h-1) + Min(h-2) + 1}$$
$$\boldsymbol{F(h) = Min(h) + 1}$$
$$\boldsymbol{F(h) = F(h-1) + F(h-2)}$$
$$\boldsymbol{F(1) = 2, F(2) = 3}$$

که در واقع به دنباله ی فیبوناچی رسیدیم. پس در نهایت خواهیم داشت:

$$\boldsymbol{Min(h) = \frac{\varphi^{h+2} - (-\varphi)^{h+2}}{\sqrt{5}} - 1}$$

نکات تکمیلی

- پاسخ های خود را تا زمان معین شده در سایت آپلود نمایید.
- هدف این تمرین یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق با قوانین درس با آن برخورد خواهد شد.
- دقت فرمایید که پاسخ سوال‌ها یکتا نیست و به دیگر پاسخ‌های صحیح نیز نمره تعلق می‌گیرد.
- در صورت وجود ابهام در مورد سوالات می‌توانید از طریق ایمیل با طراحان تمرین در ارتباط باشید.

شاد باشید.