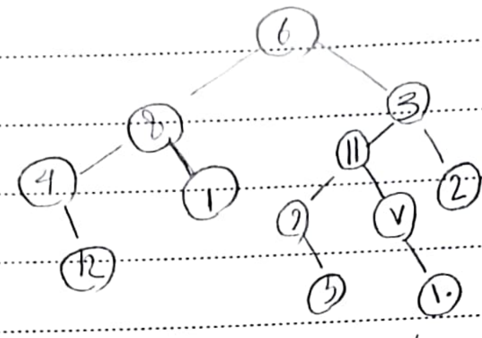
Subject

Date ۲۰ / آذر / ۱۴۰۱

شهره ادمیز، ۸۱۰۱۰۰۲۷۲

تکلیف HW، ساختمان داده و الگوریتم

سوال ۱.

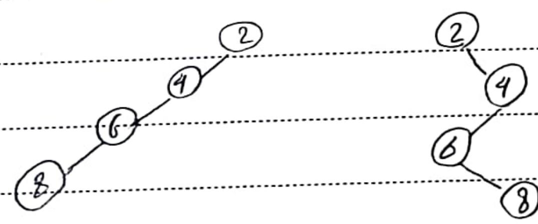


الف) در پیمایش postOrder ابتدا عنصر آخر را به عنوان ریشه انتخاب کرده سپس در پیمایش Inorder، دور ریشه خط می‌کشیم. عددهای سمت راست آن در زیر درخت راست و عددهای سمت چپ در زیر درخت چپ آن قرار می‌گیرد.

ب) نمی‌توان. در درس گفته (آزمون تیل) دیدیم که با داشتن هر دو تا از این سه پیمایش درخت تبدیل یک درخت واحد است. برای مثال با داشتن postorder و preorder اگر بدانیم درخت دودویی کامل است، عنصر اول preorder را به عنوان ریشه قرار می‌دهیم. عنصر بعدی را فرزند چپ ریشه قرار می‌دهیم. حال در postOrder تمام عناصر قبل از این عنصر، زیر درخت های آن هستند. به صورت بازگشتی تا پایان همین کار را انجام می‌دهیم.

مثال: preorder : 2, 4, 6, 8

postorder : 8, 6, 4, 2



قسمت بعدی با داشتن Inorder و preorder است که مانند قسمت الف است. فقط چون ما preorder داریم، عنصر اول را ریشه قرار می‌دهیم و زیر درخت چپ و راست را پیدا می‌کنیم.

Nevis

Subject

Date

سوال ۲. را در لیست پیوندی دوطرفه ذخیره می‌کنیم. (در ابتدا یک Node می‌سازیم و مقدار آن را برابر صفر قرار می‌دهیم.

از ریشه این درخت کار را آغاز می‌کنیم و مقدار آن را با مقدار داخل node جمع می‌کنیم و در داخل آن قرار می‌دهیم. اگر فرزند چپ داشتیم؛ چک می‌کنیم که prev.node آن به چیزی وصل است یا خیر، اگر نبود، برای فرزند چپ ریشه یک node.prev (عنصر قبلی node انتخابی) عملیات گفته شده را به صورت بازگشتی تکرار می‌کنیم. ولی اگر عنصر قبلی (node.prev) خالی یا نال بود، یک نود جدید به node.prev وصل می‌کنیم و مقدار آن را ۰ قرار می‌دهیم. و عملیات را برای فرزند چپ ریشه در node.prev انجام می‌دهیم.

درصورتی که ریشه فرزند راست داشته باشد، تمام عملیات بالا را برایش انجام می‌دهیم فقط به جای node.prev از node.next استفاده می‌کنیم و به جای اینکه روی عناصر و فرزندان چپ پیمایش کنیم روی فرزند راست پیمایش می‌کنیم.

در نهایت به تعداد ستون‌های درخت node داریم که در هر یک مجموع اعداد آن ستون قرار دارد. max را از بین آنها انتخاب می‌کنیم و بازگشت می‌دهیم. پیچیدگی زمانی از مرتبه $O(n)$ می‌باشد زیرا هر عنصر درخت تنها یک بار پیمایش می‌شود.

سوال ۳. در ابتدا برای آرایه مقدار مجموع پیشوندی را حساب می‌کنیم. به این ترتیب که هر یک از خانه‌های آرایه، حاصل جمع تمامی خانه‌های ماقبل آن و خودش است.

برای یافتن مقدار مجموع عناصر از index = a تا index = b به شرطی که $b > a$ باشد، مقدار Array[b] - Array[a] را حساب می‌کنیم.

در مرحله بعد مجموع هر آرایه‌های کشف را در یک min-heap وارد می‌کنیم. اگر مقدار عناصر کشف درخت از m کمتر بود عنصر جدید را در آرایه پوش می‌کنیم که این عملیات از مرتبه پیچیدگی $O(\log m)$ برخوردار است.

Subject

Date

(ادامه ۳)

در غیر این حالت، در صورتی که تعداد عناصر stack به تعداد m رسید، پس از خروج هر عنصر از آرایه، اگر عنصر ادامه دهنده دنباله نزولی بود، آن را pop کرده و عنصر را push می‌کنیم.

اگر حذف نشود، برای زیر آرایه بعدی می‌رویم. در هر مرحله عدد Array[0] - Array[n] را الگوریتم برای m دیگر این مرحله پوشش می‌کنیم.

حال اثبات می‌کنیم که این الگوریتم از مرتبه $O(n^2 \log(m))$ است: برای پیدا کردن تمام Array[b] - Array[a] ها ۲ حلقه تو در تو داریم. و در هر بار پیمایش عناصر، $O(\log m)$ عنصر را پوشش می‌دهیم.

$\hookrightarrow n^2 \times \log m = n^2 \log(m) \Rightarrow O(n^2 \log m)$

سوال ۴.

دو عدد x و y را از ورودی می‌گیریم. در حالتی که $y > x$ باشد، مقدار max و min را تعریف می‌کنیم به حالتی که $min = x$ و $max = b$ باشد.

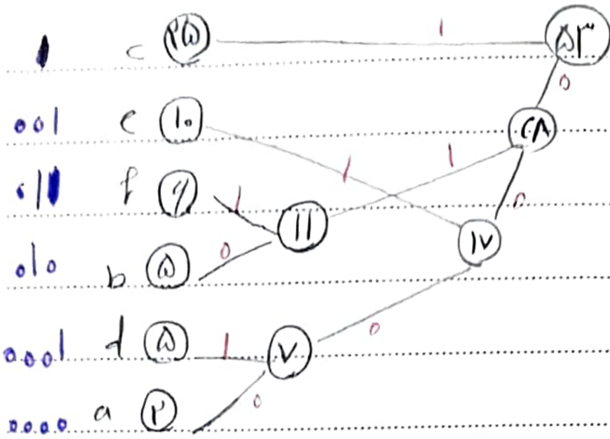
از ابتدای آرایه شروع می‌کنیم و عناصر را یکی یکی پیمایش می‌کنیم. اگر مقدار عنصر بعد از عنصر قبلی بیشتر بود، max را آپدیت می‌کنیم و در صورتی که عنصر از عنصر بعدی خود کمتر بود، min را با عدد داده شده آپدیت می‌کنیم.

در صورتی که مقدار min کاهش یا max افزایش پیدا کرد، به این معنا است که درخت جداگانه‌ای ایجاد شده، پس دنباله داده شده قابل ایجاد در BST نمی‌باشد.

الگوریتم از مرتبه $O(n)$ است زیرا هر عنصر را فقط یک بار پیمایش می‌کنیم.

Nevis

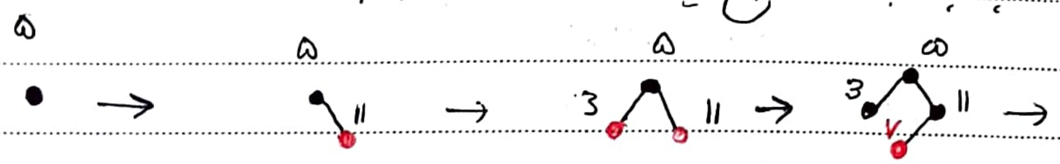
سوال ۵.

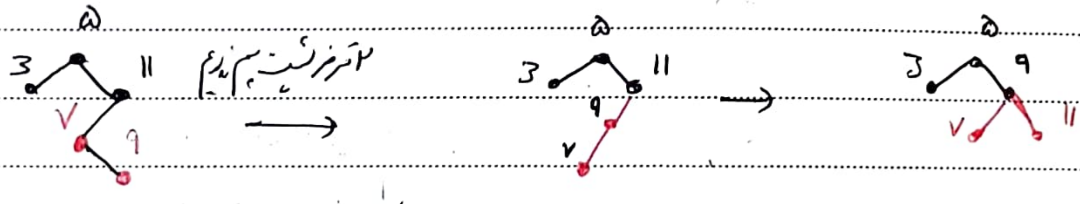


پاسخ: bdfce → 010 0001 011 1 100

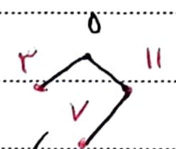
سوال ۶.

الف) از چپ به راست درج کنید: ۵، ۱۱، ۳، ۷ و ۹

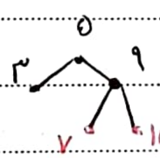




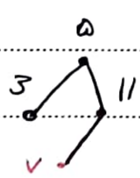
ب) یک مثال نقض. درخت



را داشتیم که با اضافه کردن ۹ تبدیل به



شد حال اگر ۹ را حذف کنیم، به



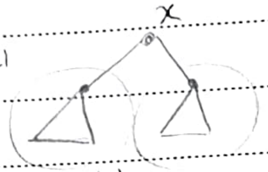
تبدیل می‌شود که با درخت اولیه متفاوت است.

ج) تعریف: هر زیر درخت دلخواه ریشه x حداقل دارای $2^{bh(x)}-1$ گره داخلی است.

اثبات: فرض استقرا: x_0 $2^{bh(x=0)}-1=0$ ✓

(ادامه ۶)

گام استقرا: اگر این قضیه برای همه زیردرخت ها با $bh = bh(x) - 1$ برقرار باشد، داریم:



حکم: $2(2^{bh(x)-1}-1)+1 = 2^{bh(x)}-1$

حال چون نمی‌توانیم ۲ گره قرمز پشت سر هم داشته باشیم، پس تعداد نصف یا بیشتر گره ها سیاه هستند

یعنی $h/2 \le bh(x)$

$\Rightarrow 2^{h/2} \le 2^{bh(x)} \Rightarrow 2^{h/2}-1 \le 2^{bh(x)}-1 \le n$

$\Rightarrow 2^{h/2} \le n+1 \Rightarrow h/2 \le \log(n+1) \rightarrow h \le 2\log(n+1) \rightarrow h = O(\log n)$

سوال ۷.

گفتیم درخت AVL اختلاف ارتفاع زیردرخت چپ و راست حداکثر ۱ است

حداقل تعداد نودها؟

تعداد نود ها = گره های زیردرخت چپ + نودهای زیردرخت راست + ۱ (ریشه)

می‌دانیم $n(h) = n(h-1) + n(h-2) + 1 \rightarrow n(h-2) < n(h-1)$

$n(h) > 2n(h-2) > 4n(h-4) > \dots \rightarrow 2^i n(h-2i)$

$\Rightarrow n(h) > 2^{h/2}$

Subject

Date

سوال ۶ فرین دوم :

```
preOrder(L:list, i:int){
  if(i >= L.size)
      return
  else if(L[i] = = Null)
      return
  print(L[i])
  preorder(L, 2i+1)
  preOrder(L, 2i+2)
}
```

```
postOrder(L:list, i:int){
  if(i>l.size)or(i<0) or(L[i]==Null)
      return
  postOrder(L, 2i+1)
  postorder(L, 2i+2)
  print(L[i])
```

فرض آن است که فرزند چپ گره در $2i+1$ و فرزند راست آن در $2i+2$ است و در صورت عدم وجود فرزند Null داریم