پاسخنامه تمرین شماره ۳

دانشکده مهندسی برق و کامپیوتر | درخت | ساختمان های داده و الگوریتم - پاییز 1400

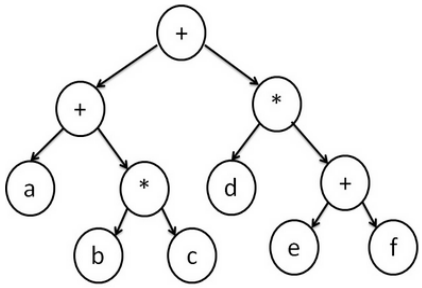
استاد: دکتر هشام فیلی | طراح تمرین : فاطمه سید دباغی

سوال اول

با توجه به درخت عبارت داده شده به سوالات پاسخ دهید.

الف) نمایش prefix درخت زیر را بنویسید. 3

ب) یک الگوریتم با پیچیدگی زمانی O(n) برای تبدیل عبارت prefix به infix ارائه دهید و با استفاده از پاسخ قسمت الف عبارت نمایش infix درخت را بنویسید. 13



پاسخ

الف)

+ + a * b c * d + e f

ب) ابتدا عبارت prefix را از راست به چپ خوانده و اگر کاراکتر خوانده شده یک operand بود آن را در یک استک پوش میکنیم. اگر کاراکتر خوانده شده یک operator بود دو operand اول استک را پاپ کرده و سپس ترکیب دو operand و operator خوانده

شده به صورت operand1+operator+operand2 را در استک پوش کرده و تمامی مراحل را تا به پایان رسیدن عبارت prefix ادامه میدهیم.

مراحل تبدیل:

stack : -

string: + + a * b c * d + e f

---

stack :

string: + + a * b c * d +

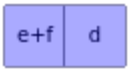
---

stack :

string: + + a * b c * d

---

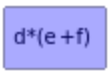
stack :



string: + + a * b c *

---

stack :



string: + + a * b c

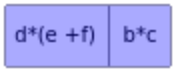
---

stack :

string: + + a *

---

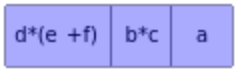
stack :



string: + + a

---

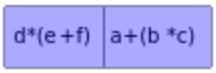
stack :



string: + +

---

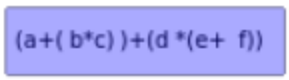
stack :



string: +

---

stack :



سوال دوم

فرض کنید یک آرایه مرتب شده به طول n داریم الگوریتمی با پیچیدگی زمانی **بهینه** ارائه دهید که:

الف) عناصر این آرایه را به گونه‌ای در یک درخت جست‌وجوی دودویی (BST) خالی اضافه کند که این درخت بیشترین ارتفاع ممکن را داشته باشد. 6

ب) عناصر این آرایه را به گونه‌ای در یک درخت جست‌وجوی دودویی (BST) خالی اضافه کند که این درخت کمترین ارتفاع ممکن را داشته باشد. 8

پاسخ

الف) آرایه را به صورت خطی اسکن میکنیم و اولین عنصر را به‌عنوان ریشه قرار داده و بقیه عناصر را در درخت وارد میکنی. در این صورت درخت کج می‌شود، یعنی تمام گره‌های درخت در یک طرف ریشه قرار می‌گیرند. بنابراین ارتفاع درخت برابر با تعداد عناصر آرایه خواهد بود که بیشترین حالت ممکن است.

ب) عنصر وسط آرایه را درنظر میگیریم. آن را به عنوان ریشه قرار میدهیم. با انجام این کار اطمینان حاصل می کنیم که نیمی از عناصر آرایه در سمت چپ ریشه و نیمی در سمت راست قرار دارند.

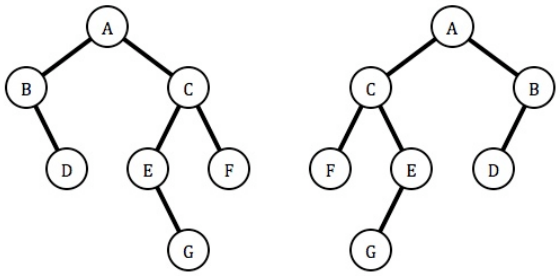
نیمه سمت چپ آرایه را درنظر گرفته و الگوریتم بالا را به صورت بازگشتی فراخوانی میکنیم و آن را به root.left اضافه میکنیم.

نیمه سمت راست آرایه را درنظر گرفته و الگوریتم بالا را به صورت بازگشتی فراخوانی میکنیم و آن را به root.right اضافه میکنیم.

سوال سوم

الگوریتمی با پیچیدگی زمانی **بهینه** ارائه دهید که یک درخت دودویی را به عنوان ورودی دریافت میکند و درختی تولید میکند که یک تصویر آینه از درخت اصلی باشد. به عنوان مثال، اگر از این روش در درختی که در سمت چپ نشان داده شده استفاده کنیم، در نهایت با درختی که در سمت راست نشان داده شده‌است استفاده می‌کنیم. (دقت کنید که از نظر تئوری این دو درخت تفاوتی ندارند اما مثلا بعد از اعمال عملیات راس B به عنوان فرزند راست راس A ذخیره شده است)

شبه کد (pseudocode) الگوریتم خود را نوشته و همچنین پیچیدگی آن را محاسبه کنید. 8



پاسخ

به صورت بازگشتی عمل میکنیم. تابع mirror را روی زیردرخت چپ و راست صدا زده و نتیجه‌ی بازگشتی را با یکدیگر جابجا میکنیم. در بدترین حالت پیچیدگی زمانی O(n) و بهترین حالت O(logn) که logn ارتفاع درخت در حالت متوازن است خواهد بود.

```
mirror(root):
    if (root == None):
        return
    else:
        temp = root
        mirror(root.left)
        mirror(root.right)

        temp = root.left
        root.left = root.right
        root.right = temp
```

سوال چهارم

n نفر دانشجو در یک ردیف از کلاس درس کنار هم نشسته اند. تعدادی از این افراد تلاشگر هستند و در کار های گروهی به اندازه‌ی a[k] کار میکنند. تعدادی از این افراد نیز تنبل هستند و در صورت اضافه شدن به هر گروهی باعث می‌شوند تا توان کاری آن گروه به اندازه a[k] کاهش یابد. دانشجو ۱ تا n به ترتیب داخل یک صف ایستاده اند.

برای انجام یک کار می خواهیم تعدادی دانشجو **متوالی** از این صف را انتخاب کنیم، به دلیل نامعلومی می خواهیم m-امین بزرگترین توان را از بین همه گروه های ممکن (زیردنباله‌های متوالی در آرایه) داشته باشند! (توان هر گروه مجموعه توان اعضای آن است) الگوریتمی با پیچیدگی زمانی **بهینه** ارائه دهید که با داشتن m، m-امین بزرگترین توان گروه را پیدا کند. 14

پاسخ

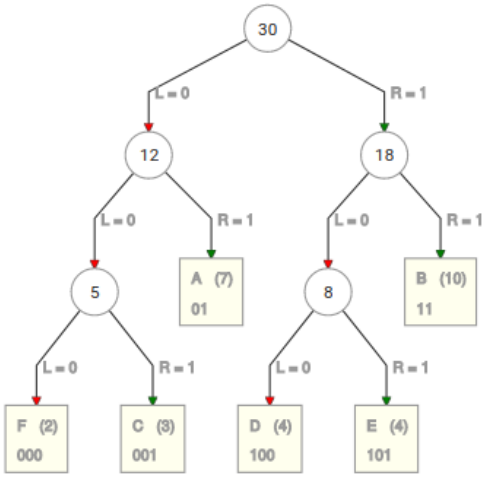
در ابتدا برای ساده‌تر شدن کار آرایه‌ی جدیدی با نام sum تعریف میکنیم. ایندکس i این آرایه نشان دهنده ی جمع 0 تا i آرایه‌ی ابتدایی است. با این کار می‌توانیم مجموع زیرآرایه‌های پیوسته را از ایندکس i تا j به صورت sum[j]-sum[i-1] پیدا کنیم. حال برای یافتن پاسخ خود از min heap استفاده میکنیم. در ابتدا m عنصر اول را که به کمک حلقه‌ی تو در تو بدست میاوریم در min-heap اضافه میکنیم. پس از آن برای اضافه کردن هر عنصر بررسی میکنیم که آیا آن مقدار از مقدار min_heap.top() بزرگتر است یا نه. اگر بزرگتر بود min_heap.top() را pop کرده و عنصر جدید را اضافه میکنیم و عمل heapify را انجام میدهیم. اینکار را تا آخر برای همه حالات انجام میدهیم. درنهایت min_heap.top() پاسخ مورد نظر ما خواهد بود. پیچیدگی زمانی این الگوریتم O(n ∧ 2 log (m)) خواهد بود. پیچیدگی حافظه نیست با توجه به اینکه سایز heap ثابت و m است O(m) است.

سوال پنجم

کد بهینه‌ی هافمن برای کاراکترهای زیر با توجه به جدول احتمال پیشامد آن‌ها چیست؟ درخت هافمن خواسته شده را بسازید و کلمه‌ی "decaf" را در کد باینری متناظر نمایش دهید. 11

Letter	*Probability*	*Letter*	*Probability*	*Letter*	*Probability*
a	*7/30*	*b*	*10/30*	*c*	*3/30*
d	*4/30*	*e*	*4/30*	*f*	*2/30*

پاسخ



Letter	*Encoding*	*Letter*	*Encoding*	*Letter*	*Encoding*
a	*01*	*b*	*11*	*c*	*001*
d	*100*	*e*	*101*	*f*	*000*

decaf -> 10010100101000

سوال ششم

الف) اعداد 80, 40, 25, 17, 15, 5, 18, 8 را به ترتیب از چپ به راست در یک درخت قرمز-سیاه درج کنید. 13

ب) حداکثر ارتفاع یک درخت قرمز-سیاه با ۸ راس چقدر است؟ 8

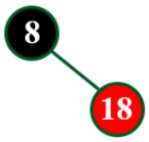
ج) نشان دهید در یک درخت قرمز-سیاه ارتفاع هر راس حداکثر ۳ برابر کوتاه‌ترین فاصله از این راس به یکی از برگ‌هاست. 8

پاسخ

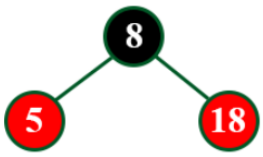
الف)

8:

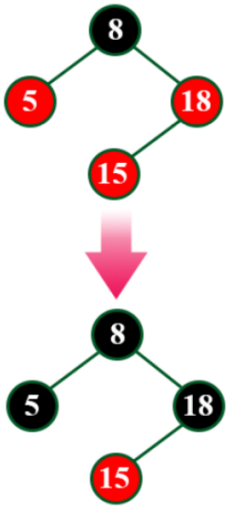
18:



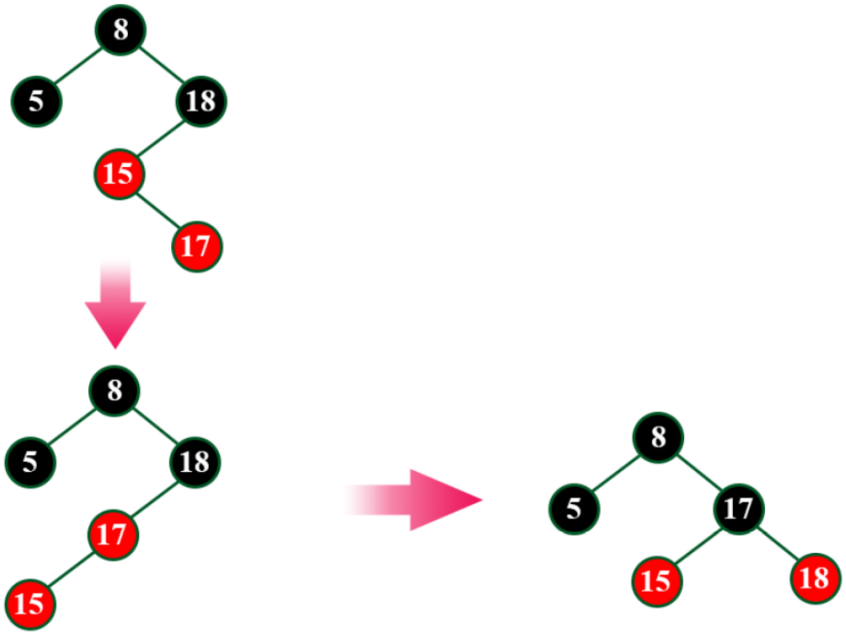
5:



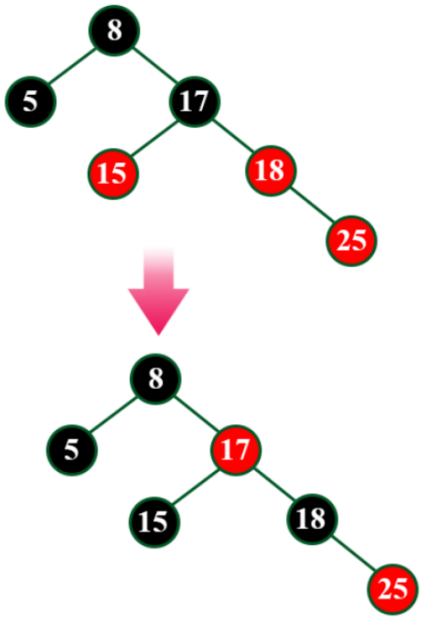
15:



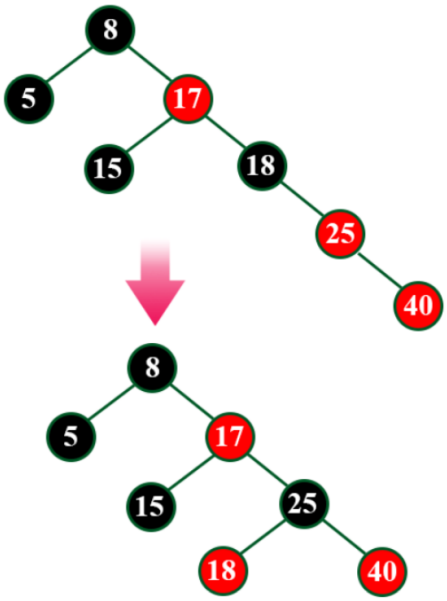
17:



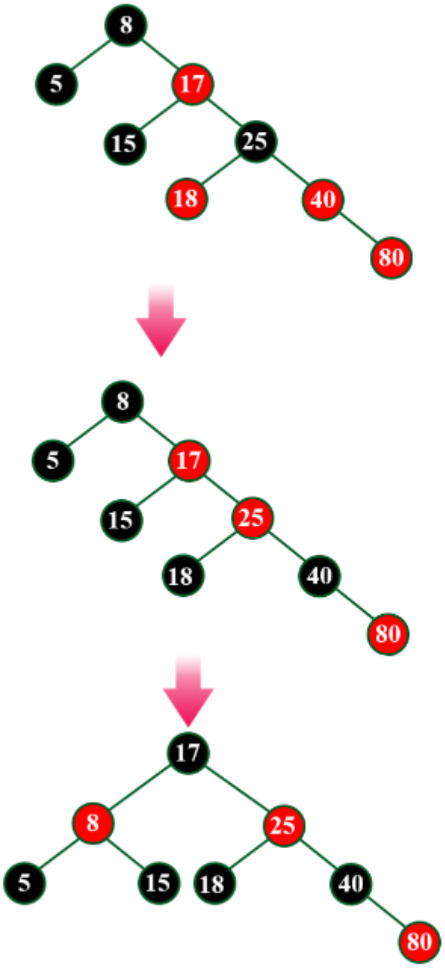
25:



40:


8
5
17
15
18
25
40
8
5
17
15
25
18
40

80:


8
5
17
15
25
18
40
80
8
5
17
15
25
18
40
80
17
8
25
5
15
18
40
80

ب) برای به دست آوردن حداکثر ارتفاع باید طولانی‌ترین مسیر را پیدا کنیم. میدانیم طولانی‌ترین مسیر از ریشه تا برگ‌ها با توجه به محدودیت‌های درخت دارای حداکثر $\log2(n+1)$ گره با رنگ سیاه است. همچنین یک گره قرمز باید دارای فرزندان سیاه باشد که با توجه به محدودیت گره‌های سیاه تعداد آن‌ها حداکثر $\log2(n+1)$ خواهد بود. درنتیجه:

Maximum height = max black nodes + max red nodes

$= \log2(n+1) + \log2(n+1)$

$= 2.\log2(n+1) = 2.\log2(9) = 6.34 = 6$

ج) ارتفاع راس x را h(x) و کوتاهترین فاصله از این راس تا یکی از برگ‌ها را d(x) مینامیم. bh(x) تعداد گره های سیاه رنگ در مسیری از ریشه تا یک برگ است. گره های برگ نیز گره سیاه شمرده می شوند. با توجه به ویژگی‌های درخت قرمز-سیاه میتوان نتیجه گرفت که $h(x)/2 =< bh(x)$است. میدانیم $bh(x) =< d(x)$است پس میتوان نتیجه گرفت $h(x) =< 2d(x)$.

سوال هفتم

نشان دهید پیچیدگی زمانی یک avl-insert که بر روی یک درخت AVL با n نود انجام میشود O(lgn) است و همچنین تعداد rotate ها نیز O(1) است. 8

پاسخ

میدانیم عملیات insertion در avl tree همان عملیات insert در BST به همراه چند rotation است. در عملیات insert در یک درخت avl میتوانیم از زمان rotation صرف نظر کنیم چون تنها چند پوینتر عوض میشوند و این عملیات از مرتبه O(1) خواهد بود. بنابراین پیچیدگی زمانی O(h) و با توجه به اینکه درخت avl یک balanced-tree است (h=logn) پیچیدگی زمانی یک avl-insert برابر O(logn) است.