

## پاسخ سوالات تمرین چهارم

- ۱- با استفاده از تابع درهم سازی  $h(k) = k \bmod 8$ ، اعداد 5, 27, 16, 37, 75, 33, 24, 60, 10, 42, 56، را به ترتیب در جدولی به اندازه ۸ درج کنید. برای این منظور آرایه ای با ۸ عنصر رسم کنید و هر خانه را با عدد/اعداد مورد نظر پر کنید. برای حل مشکل برخورد ها از روش زنجیره ای استفاده کنید.

	$\bmod 8$	idx
5	5	0
27	3	1
16	0	2
37	5	3
75	3	4
33	1	5
24	0	6
60	4	7
10	2	
42	2	
56	0	
65	1	

شکل بالا در حالتی است که عنصر جدید را به انتهای لینک لیست اضافه کنیم. در حالت دیگر می توان عنصر جدید را به ابتدای لینک لیست اضافه کرد که در این صورت لینک لیست ها برعکس می شوند.

۲- اگر بخواهیم یک درخت red/black را با ویژگی‌های زیر augment کنیم، آیا اردر عملیات (search و insert, delete) تغییری می‌کند؟ اگر تغییر می‌کند توضیح دهید چگونه تغییر می‌کند و اگر نه، توضیح دهید چرا تغییر نمی‌کند.

از آنجایی که ارتفاع سیاه یک گره فقط به ارتفاع سیاه و رنگ فرزندان بستگی دارد، قضیه 14.1 نشان می‌دهد که ما می‌توانیم این ویژگی را بدون تأثیر بر اردر سایر عملیات درخت ردبلک حفظ کنیم.

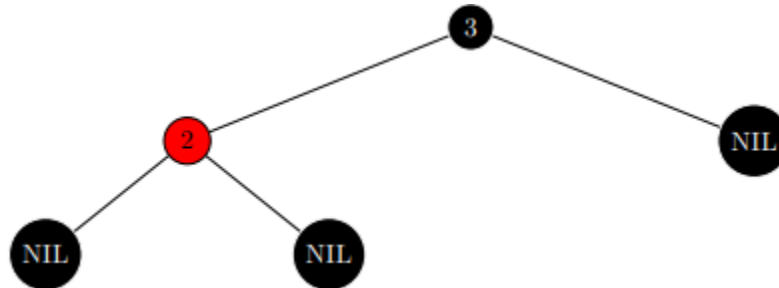
- **Theorem 14.1** (Augmenting a red-black tree)

Let  $f$  be a field that augments a red-black tree  $T$  of  $n$  nodes, and suppose that the contents of  $f$  for a node  $x$  can be computed using only the information in nodes  $x$ ,  $left[x]$ , and  $right[x]$ , including  $f[left[x]]$  and  $f[right[x]]$ . Then, we can maintain the values of  $f$  in all nodes of  $T$  during insertion and deletion without asymptotically affecting the  $O(\lg n)$  performance of these operations.

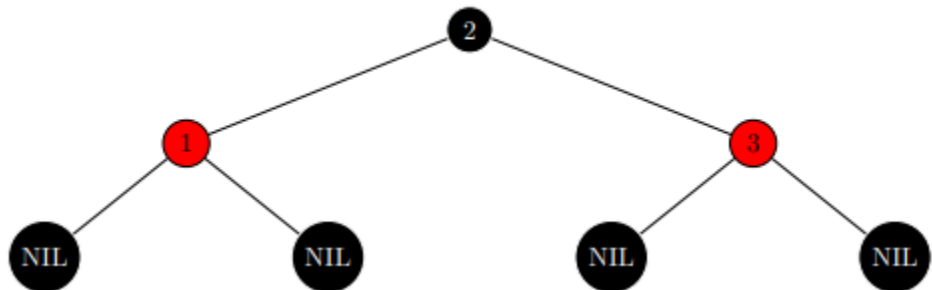
ولی این قضیه برای عمق درخت برقرار نیست. مثلاً برای عملیات delete اگر ریشه‌ی درخت را حذف کنیم، ممکن است مجبور شویم تا  $O(n)$  تا از گره‌ها را آپدیت کنیم.

۳- تصور کنید گره‌ای مشخص به یک درخت red/black اضافه می‌شود. سپس بلافاصله آن گره از درخت حذف می‌شود. آیا درخت red/black حاصل با درخت قبلی یکسان است؟ اگر بله اثبات کنید در غیر اینصورت مثال نقض بیاورید.

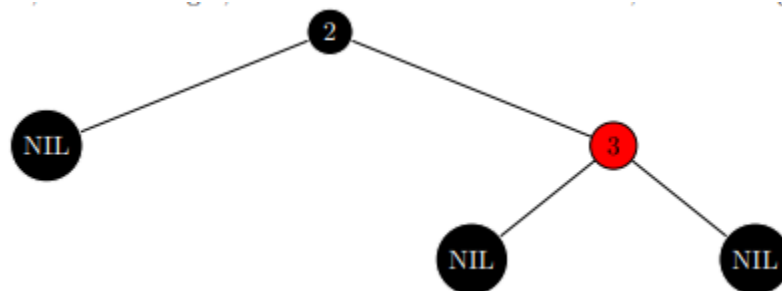
خیر درخت حاصل با درخت قبلی یکسان نیست. برای بررسی این امر میتوان مثال نقض زیر را بررسی کرد:  
ابتدا دو عدد ۳ و ۲ را به همین ترتیب به درخت اضافه میکنیم تا درخت زیر حاصل شود:



حال عدد یک را به درخت اضافه کرده تا به درخت زیر برسیم:



سپس بلافاصله عدد یک را حذف میکنیم. مشاهده می‌شود که درخت حاصل به شکل زیر است:





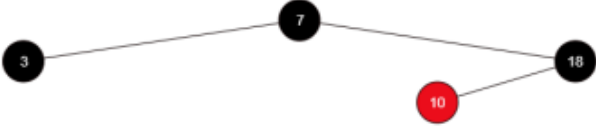
به شکل واضحی مشخص است که این درخت با درخت پیش از اضافه کردن عدد ۱ یکسان نیست.


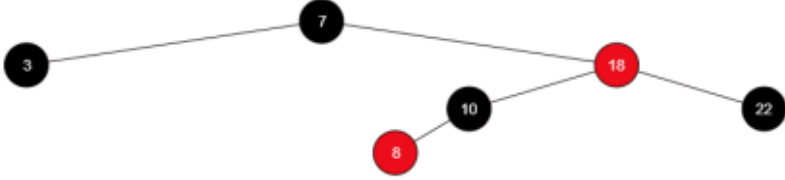
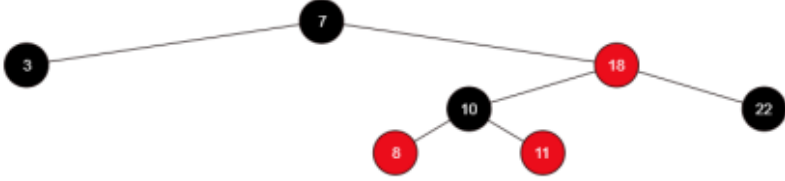
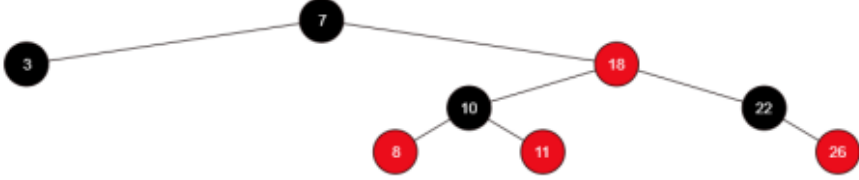
۳- موارد زیر را به ترتیب به یک درخت red/black اضافه کنید. در هر مورد مراحل و کیس‌ها را توضیح دهید. (از چپ به راست شروع می‌شود)

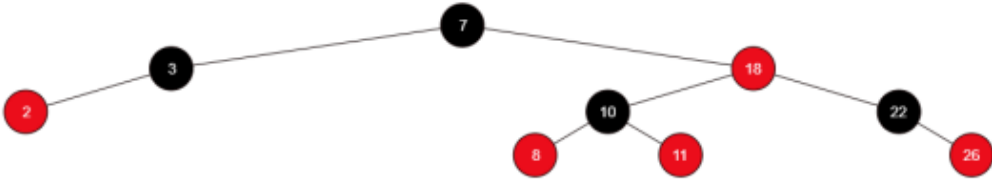
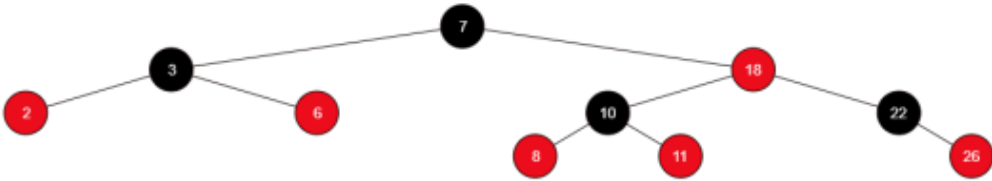
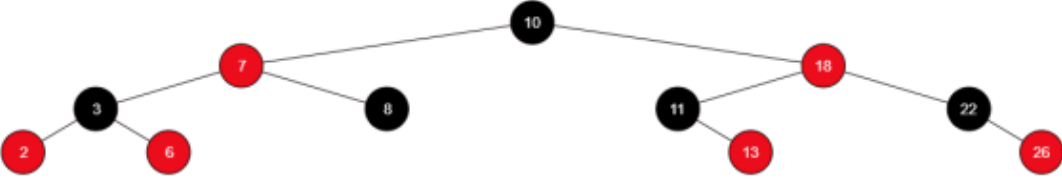
7, 3, 18, 10, 22, 8, 11, 26, 2, 6, 13

پس از اضافه کردن مقادیر بالا، مقادیر زیر را به ترتیب از چپ به راست از درخت حاصل حذف نمایید و مجدداً در هر مورد مراحل و کیس‌ها را توضیح دهید.

18, 11, 3, 10, 22,

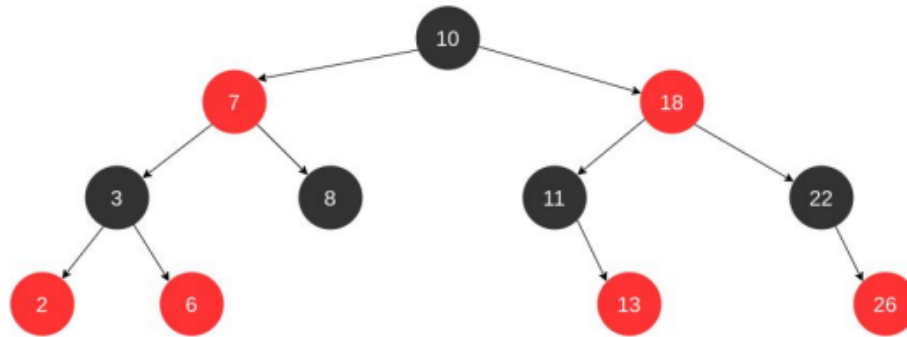
 <p>درج 7: کیس 0 (نود 7 ریشه است) درج 3: - (بابا مشکلی است)</p>	7, 3
 <p>درج 18: - (بابا مشکلی است)</p>	18
 <p>درج 10: کیس 1 (بابا و عموی نود 10 قرمز است) + کیس 0 (نود 7 ریشه است)</p>	10

 <p>درج 22: - (بابا مشکى است)</p>	22
 <p>درج 8: کيس 1 (بابا و عموى نود 8 قرمز است)</p>	8
 <p>درج 11: - (بابا مشکى است)</p>	11
 <p>درج 26: - (بابا مشکى است)</p>	26

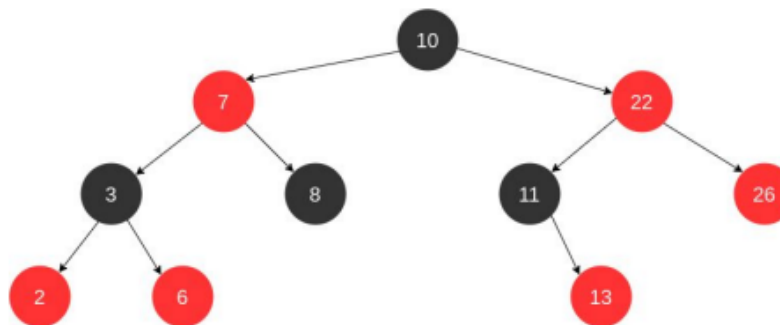
 <p>درج 2: - (بابا مشکی است)</p>	2
 <p>درج 6: - (بابا مشکی است)</p>	6
 <p>درج 13: کیس 1 (بابا و عموی نود 13 قرمز است) + کیس 2 (بابای نود 10، قرمز و عموی مشکی است). حالت مثلث</p>	13

پاک کردن:

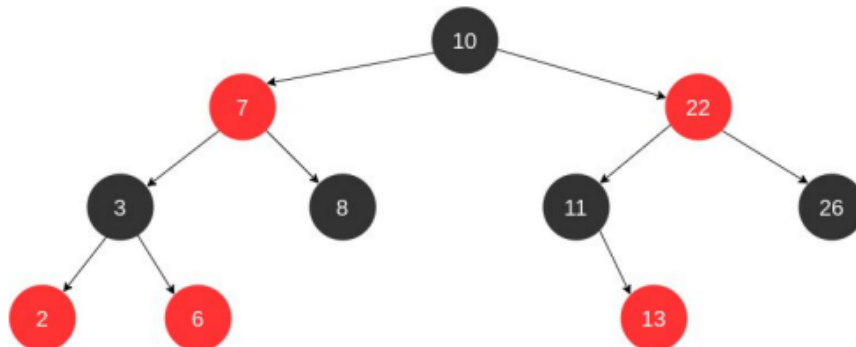
درخت اولیه:



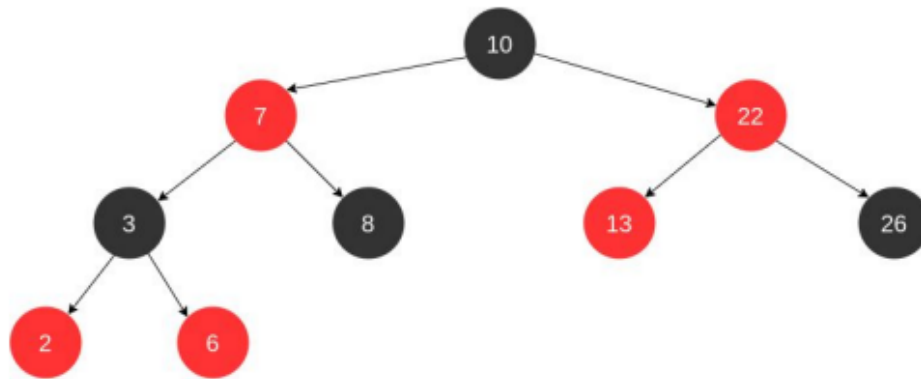
18 پارت ۱ (حذف نود ۱۸): دوتا بچه دارد، پس کیس ۳ است. محتوای ساکسسور آن یعنی ۲۲ در آن کپی می‌شود و سپس باید آن ۲۲ اضافی را پاک کنیم که کیس ۲ است و به جایش ۲۶ می‌نشیند. حاصل:



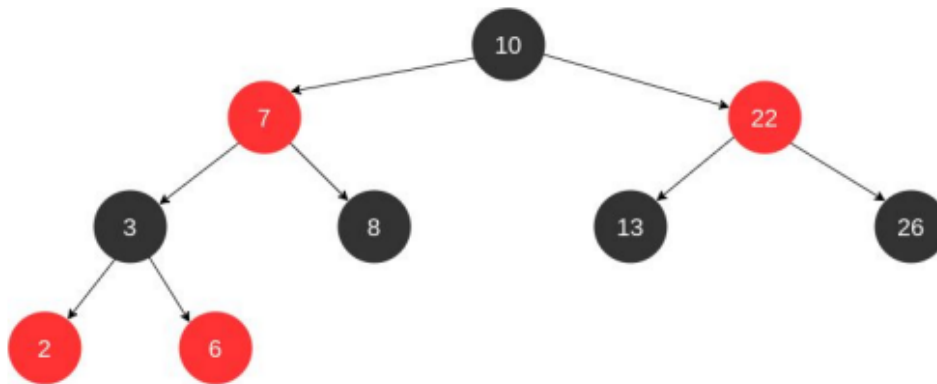
نودی که پاک شد مشکلی بود در نتیجه violation داریم. x، نود ۲۶ است و فیکس آپ را با آن صدا می‌زنیم. پارت ۲ (فیکس آپ): کیس صفر (نقیض شرط حلقه‌ی while) حاکم است، در نتیجه کافیسیت x را مشکلی کنیم. حاصل:



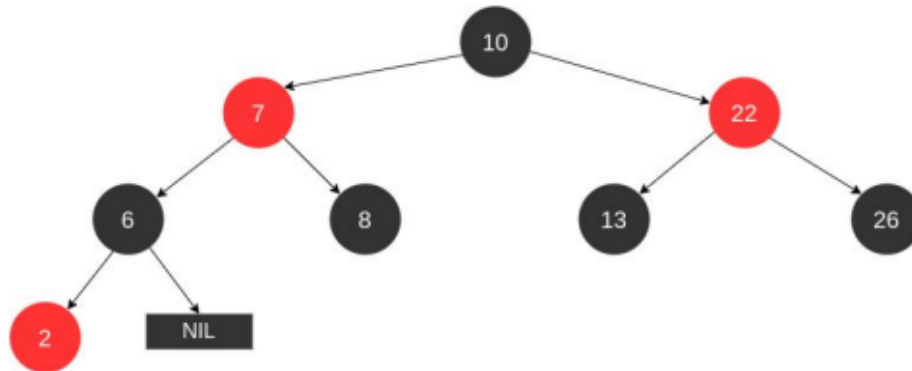
پارت ۱ (حذف نود ۱۱): یک فرزند دارد در نتیجه کیس ۲ است. ۱۳ جایگزینش می‌شود. حاصل:



نودی که پاک شد مشکلی بود، در نتیجه violation داریم. x، نود ۱۳ است و فیکس‌آپ را با آن صدا می‌زنیم. پارت ۲ (فیکس‌آپ): کیس صفر (نقض شرط حلقه‌ی while) حاکم است، در نتیجه کافیست x را مشکلی کنیم. حاصل:



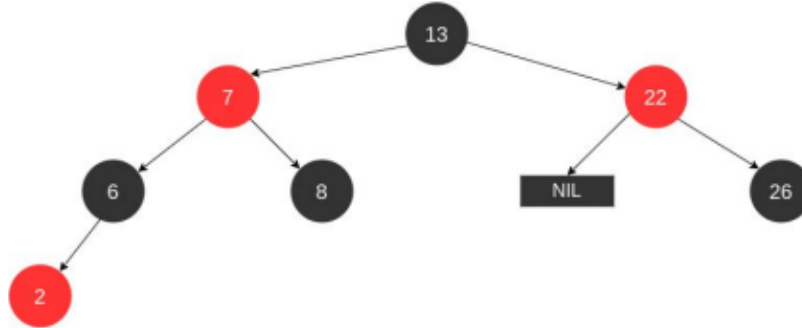
پارت ۱ (حذف نود ۳): دوتا بچه دارد، پس کیس ۳ است. محتوای ساکسور آن یعنی ۶ در آن کپی می‌شود و سپس باید آن ۶ اضافی را پاک کنیم که کیس ۱ است و به جایش NIL می‌نشیند. حاصل:



چون نودی که پاک شد قرمز رنگ بود، در نتیجه violation ای رخ نمی‌دهد و نیاز به پارت ۲ نیست.

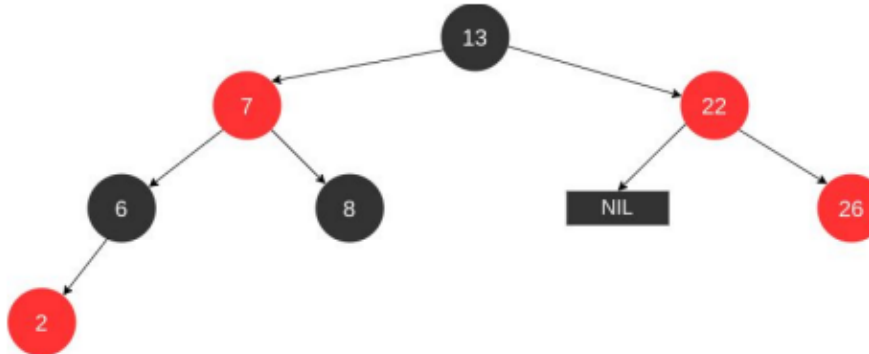


پارت ۱ (حذف نود ۱۰): دوتا بچه دارد، پس کیس ۳ است. محتوای ساکسور آن یعنی ۱۳ در آن کپی می‌شود و سپس باید آن ۱۳ اضافی را پاک کنیم که کیس ۱ است و به جایش NIL می‌نشیند. حاصل:

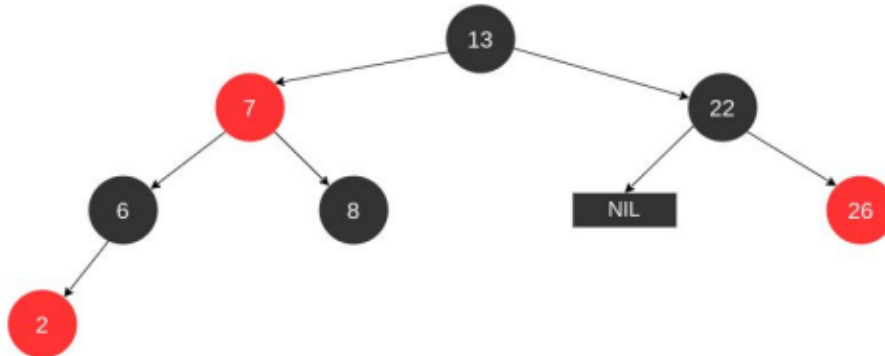


چون نود ۱۳ ای که پاک کردیم مشکلی بود، پس violation رخ می‌دهد. x در اینجا NIL است و فیکس‌آپ را با آن صدا می‌زنیم.

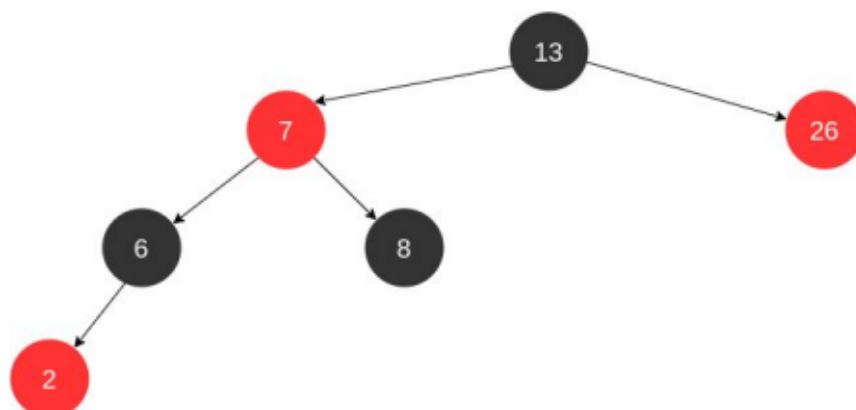
پارت ۲ (فیکس‌آپ): کیس ۲ رخ می‌دهد، زیرا sibling نود x ما (که می‌شود نود ۲۶) مشکلی است و دو فرزند مشکلی نیز دارد. رنگ sibling را قرمز می‌کنیم. حاصل:



حال x جدید می‌شود پدر x فعلی؛ یعنی x جدید می‌شود نود ۲۲. حال فیکس‌آپ را بر روی ۲۲ انجام می‌دهیم. کیس صفر (نقیض شرط حلقه‌ی while) حاکم است، در نتیجه کافیسست x را مشکلی کنیم. حاصل:



پارت ۱ (حذف نود ۲۲): فقط فرزند راست دارد در نتیجه کیس ۱ است. ۲۲ پاک شده و به جایش ۲۶ می‌نشیند. حاصل:



چون نودی که پاک شد مشکلی بود، پس violation داریم. x، نود ۲۶ است و فیکس‌آپ را با آن صدا می‌زنیم. پارت ۲ (فیکس‌آپ): کیس صفر (نقض شرط حلقه‌ی while) رخ می‌دهد چون که ۲۶ قرمز است. کافیت رنگش را مشکلی کنیم. حاصل:

