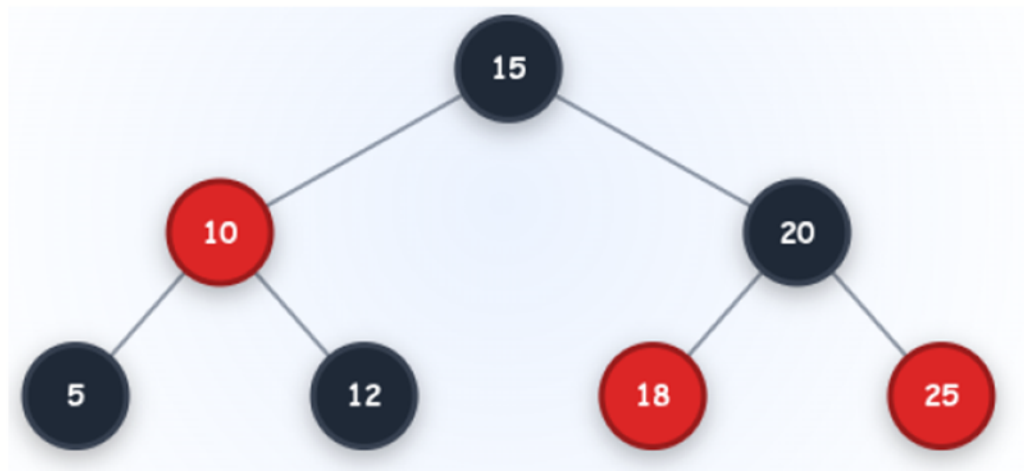
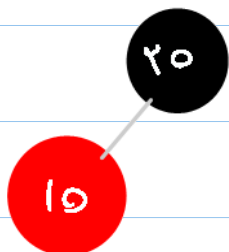
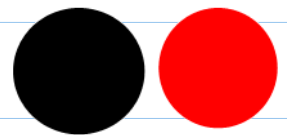


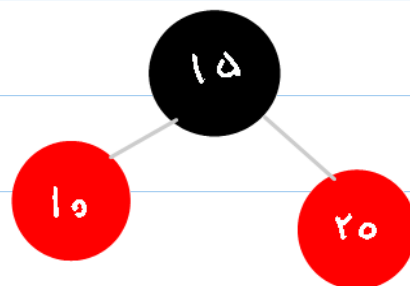
الف) درخت قرمز-سیاه زیر را در نظر بگیرید. یک ترتیب از insertion، یا شروع از نود 20 ارائه دهید که این درخت را خروجی می دهد. (راهنمایی: می توانید یک بار یک نود دلخواه را insert و سپس delete کنید، به جز این مورد اجازه ی استفاده از delete را ندارید)



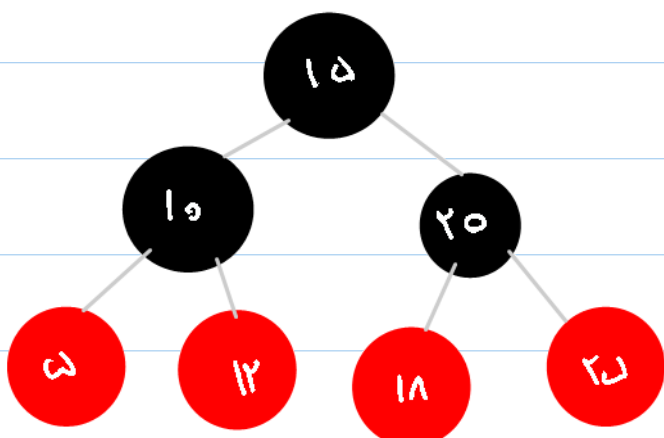
درج ۲۰ :



درج ۱۰ :

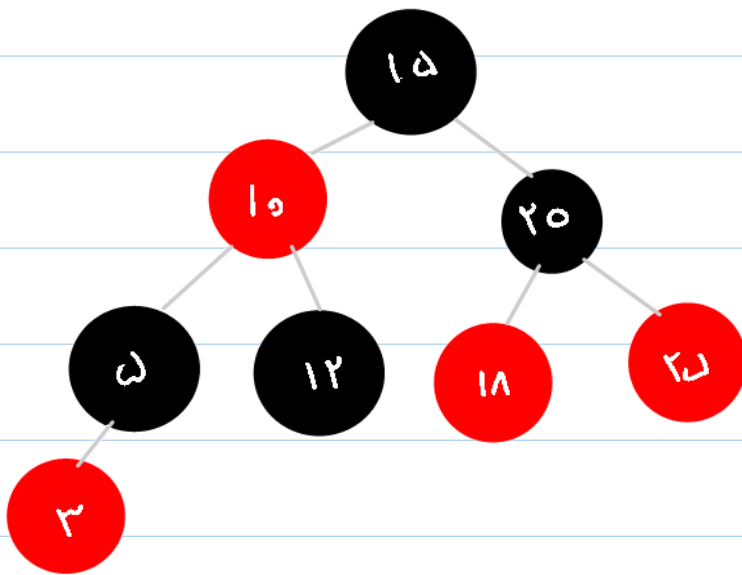


درج ۱۵ :

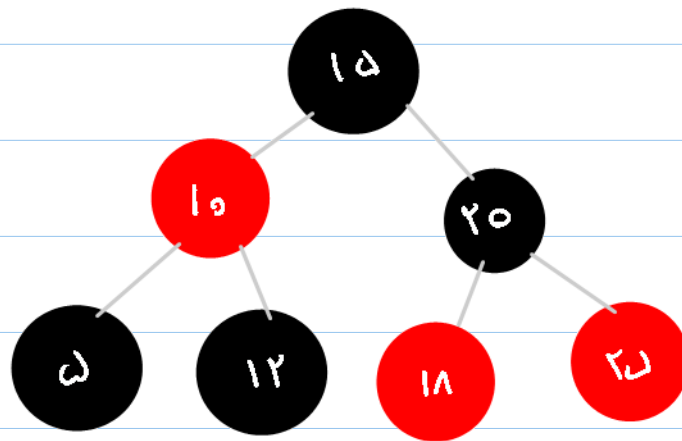


درج ۱۸ و ۲۵ و ۱۲ و ۵ :

درج ۳ :



حالت ۳ :



ب) در کد مربوط به insert در درخت قرمز-سیاه، تابعی به نام `fix_violation()` تعریف می شود. در صورتی که امکان نقض قوانین درخت قرمز-سیاه، بعد از انجام insertion وجود داشته باشد، این تابع با انجام دنباله ای از اعمال، این violation را اصلاح می کند.

- باگ (یا نقص منطقی) موجود در پیاده سازی این تابع را مشخص کنید.
- درختی ارائه دهید که به دلیل این نقص، پس از اجرای کامل عملیات insert، درخت نهایی همچنان ناقض یکی از قوانین درخت قرمز-سیاه باشد.
- مشخص کنید کدام قانون درخت قرمز-سیاه نقض شده است و چرا این تابع قادر به اصلاح آن نبوده است.
- توضیح دهید که پیاده سازی درست این تابع به طور کلی چگونه حالات مختلف نقض قوانین را تشخیص داده و اصلاح می کند.

```
while node != root and node.parent.color == RED:
```

```
parent = node.parent
grandparent = parent.parent
uncle = getUncle(node)
```

```
if uncle != null and uncle.color == RED:
```

```
parent.color = BLACK
```

```
uncle.color = BLACK
```

```
grandparent.color = RED
```

→ node = grandparent

```
else:
```

```
if parent == grandparent.left:
```

```
if node == parent.left:
```

```
rightRotate(grandparent)
```

```
swapColors(parent, grandparent)
```

```
else:
```

```
leftRotate(parent)
```

```
rightRotate(grandparent)
```

```
swapColors(node, grandparent)
```

```
else:
```

```
if node == parent.left:
```

```
rightRotate(parent)
```

```
leftRotate(grandparent)
```

```
swapColors(node, grandparent)
```

```
else:
```

```
leftRotate(grandparent)
```

```
swapColors(parent, grandparent)
```

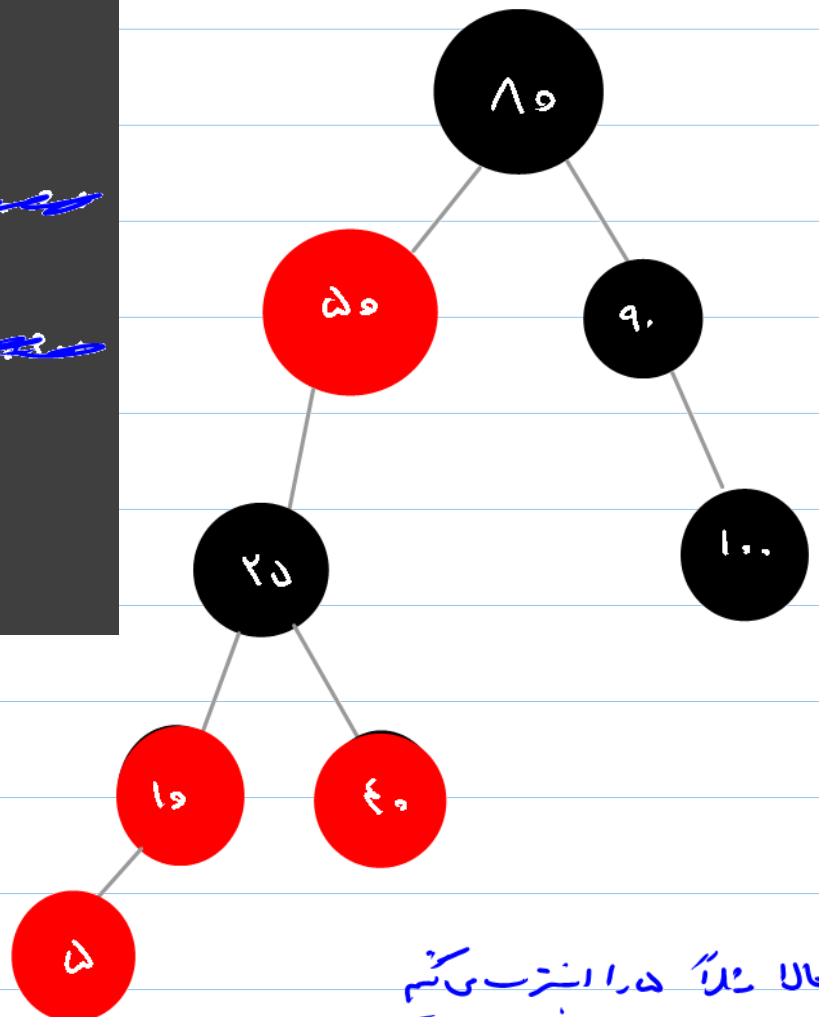
```
root.color = BLACK
```

باگ: در صورتی که محو، قرمز باشد و پدر بزرگ

قرمز شده، باید تا ۱۰۰ رفت و پدرش را یک کر

در واقع یک آپدیت داریم با مانده است

node = grandparent



حالا مثلاً ۵۰ را اینتر می‌کنیم

۲۰ قرمز می‌شود و ۵۰ را ۴۰ می‌شود، ولی می‌بینیم که ۵ هم قرمز است پس این رده که نباید قرمز

فرزند قرمز داشته باشد، منتفی می‌شود

این تابع اینگونه عمل می‌کند: تا وقتی که به ۱۰۰ نرسیم و پدر قرمز نداشته باشد:

اگر محو و پدر قرمز بوده به آما مارا می‌کنی، پدر بزرگ را قرمز و با خود پدر بزرگ کار میکن

اگر محو سیاه بود و پدر قرمز بود :

① گره مُرند چپ و پدر مُرند یپ : چرخش به راست مدل پدر بزرگ + تقویم رنگ پدر و پدر بزرگ

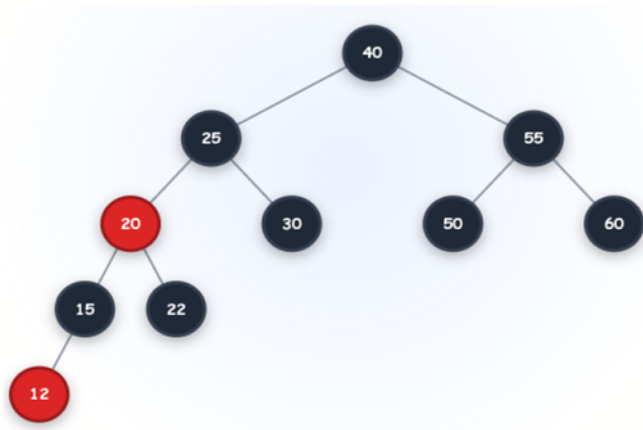
② گره مُرند راست و پدر مُرند راست : چرخش به چپ مدل پدر بزرگ + تقویم رنگ پدر و پدر بزرگ

③ گره مُرند راست و پدر مُرند یپ : چرخش به چپ مدل پدر ، چرخش راست مدل پدر بزرگ
+ تقویم رنگ گره با پدر بزرگ

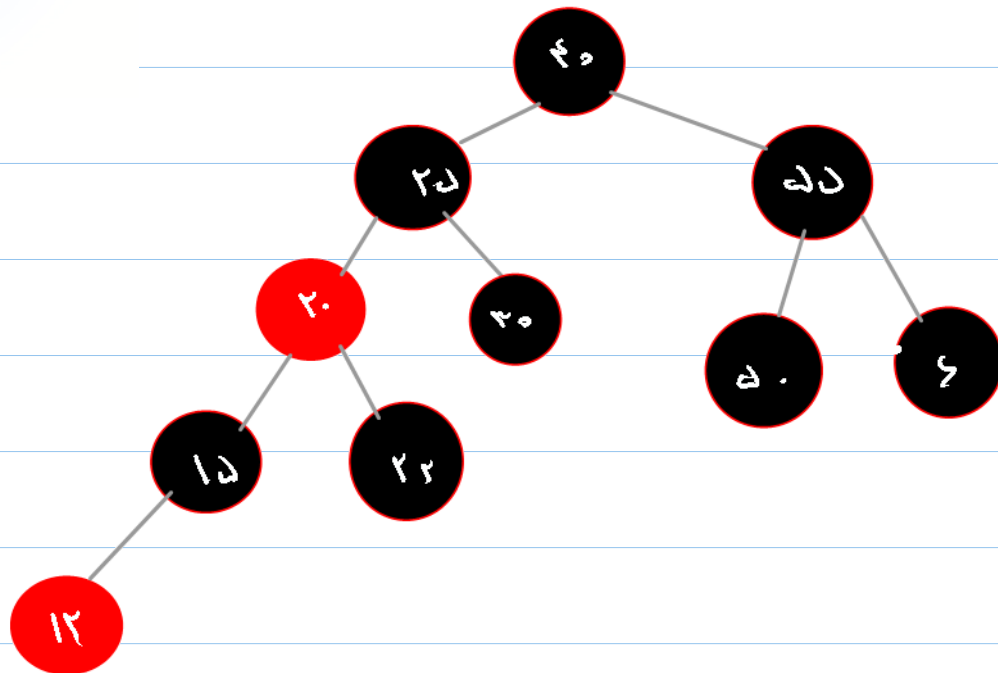
④ گره مُرند چپ و پدر مُرند راست : چرخش به راست مدل پدر ، چرخش به چپ مدل پدر بزرگ
+ تقویم رنگ گره با پدر بزرگ

(پ) مقادیر زیر را به ترتیب از چپ به راست از درخت زیر حذف کنید.
در هر مرحله، مراحل انجام حذف را توضیح دهید و درخت حاصل را رسم کنید؛ سپس حذف بعدی را روی درخت به دست آمده انجام دهید.
(هنگام حذف نود با دو فرزند، مقدار آن را الزاماً با Inorder Predecessor جایگزین کنید.)

30, 50, 15, 12, 40, 25, 20

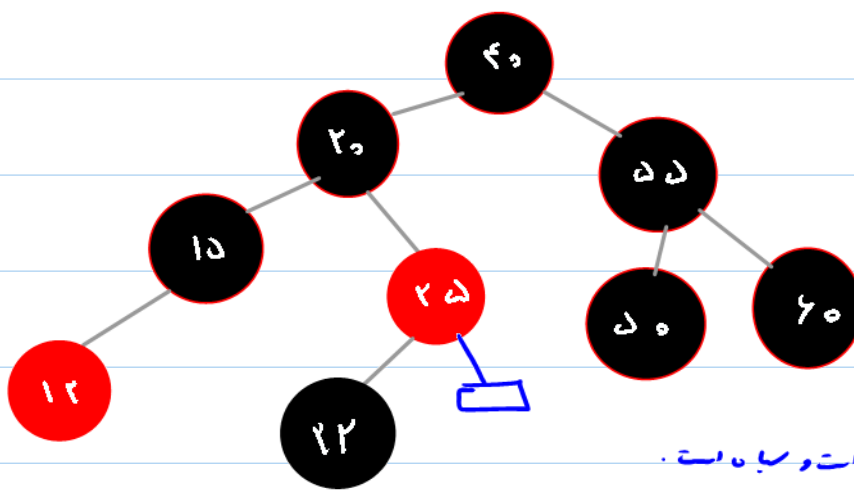


حذف ۳۰: چون گره هفتی است، حذف آن مشکلی نیست



۳۰ حذف می شود، چون سیاه است و برادر، مشکلی ایجاد نمی کند

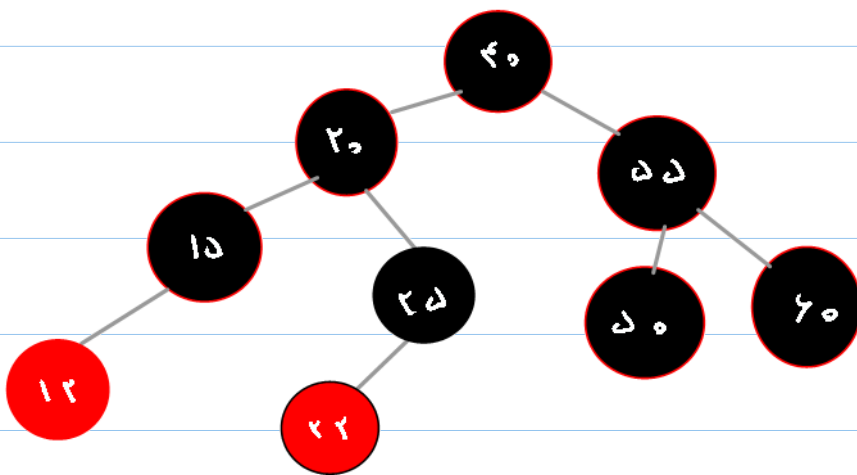
برادر ۳۰ به ۲۰ که قرمز است، رنگ برادر و پدر را عوض می کنیم به هفتی : ۲۰ و فرزند : ۲۵
فرزند راست پدر



الان یک برگه که برای ۲۲ است و سیاه است.

فرزند ۲۲ برگه‌ای که سیاه شده

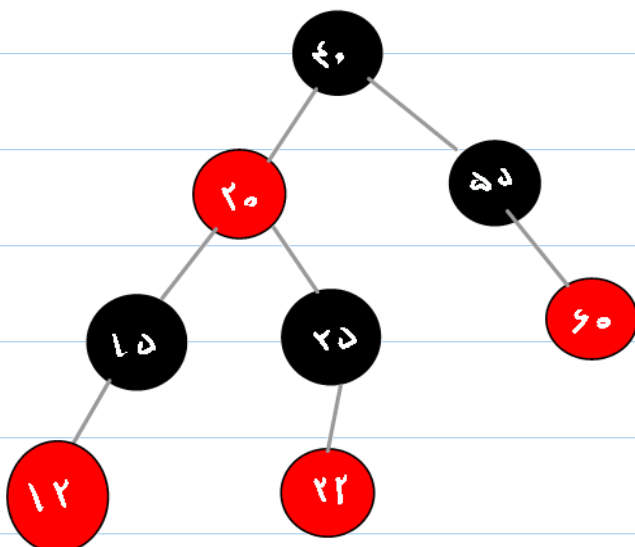
حالا برای برگه ۲۲ فرزند پیدا می‌کنیم و چون پدرش قرمز است آن را سیاه می‌کنیم



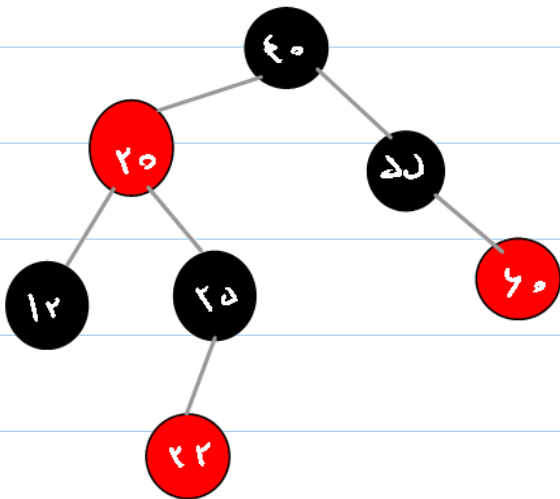
حذف ۵۰ : برای ۵۰ که سیاه است و خودش سیاه است و فرزندانی هم سیاه است.

۶۰ فرزند ۵۰ به پدر منتقل می‌شود و پدر بزرگ ۴۰ سیاه است و ۲۰ نیز سیاه است.

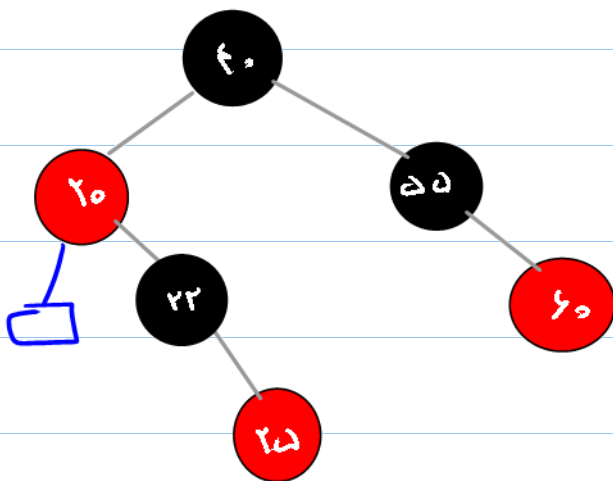
فرزند ۲۰ ۱۵ و ۲۵ که سیاه شده به ۲۰ منتقل می‌شود



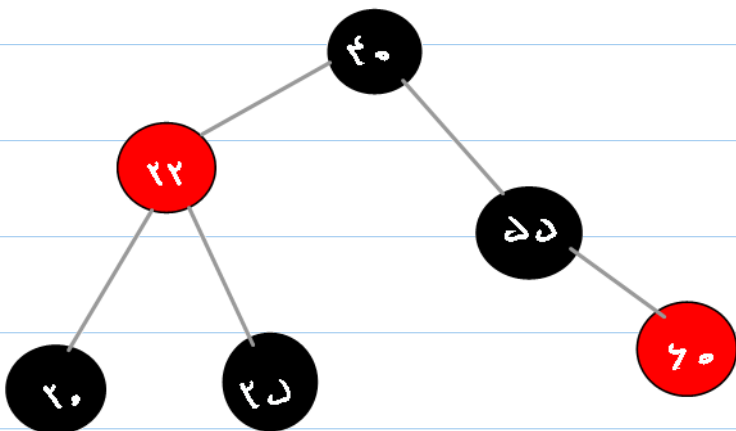
حذف ۱۵ : سیاه با ورنه فرز ۱۲ / ۱۲ جاگزین ۱۵ می شود و رنگ آن مشکی با سیاه می شود



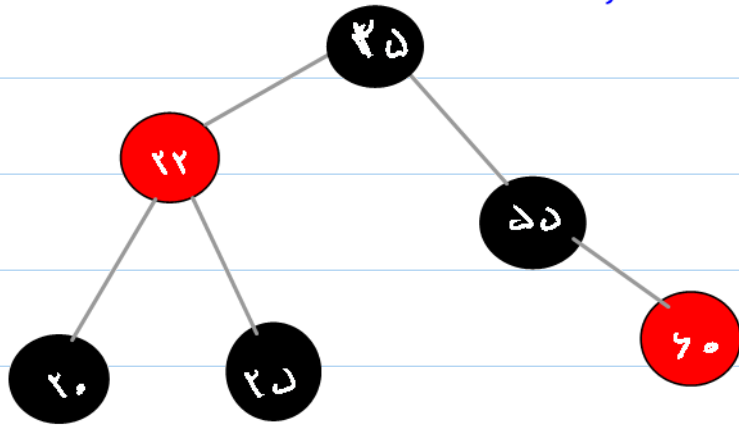
حذف ۱۳ : برادر به سیاه ، ورنه رنگ برادر به ۲۲ که فرزند ۲۵
رنگ ۲۲ در ۲۵ را معنی می دهد فرز ۲۵ و ۲۲ مشکی



فرزند دور برادر (۲۲) قرمز است
رنگ ۲۲ را به رنگ والد ۲۰ تغییر می دهیم
۲۰ سیاه و ۲۲ قرمز
فرزند دور را که ۲۵ است سیاه می کنیم
چون به پ دور ۴۰ می زنیم



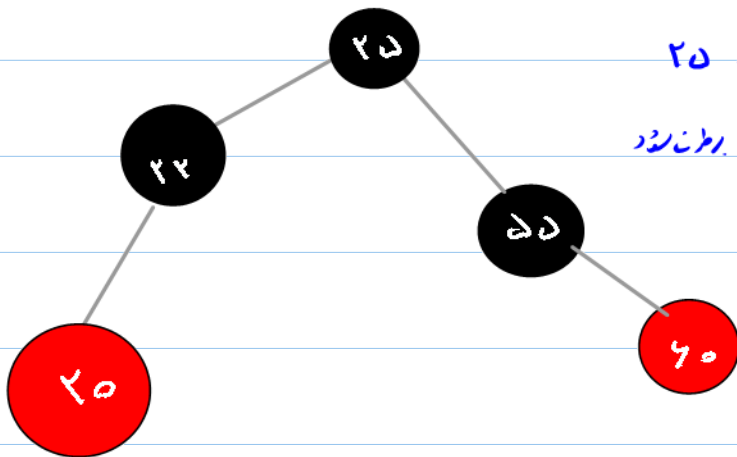
حذف ۴۵ : ۲ فرزند دارد $IP = 25$ ، ۲۵ باینین ۴۰ سطر



حالا باید ۲۵ را از پایین حذف کرد :

۲۵ سیاه ، برادر ۲۰ با ۲ فرزند سیاه

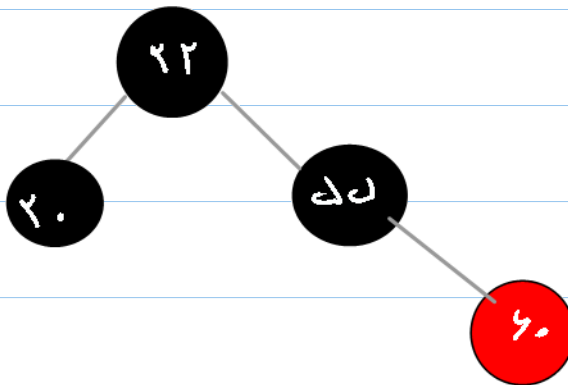
۲۰ را فرزینی نیم و رنگش را باید رعایت کنیم



حذف ۲۵ : ۲ فرزند دارد $IP = 22$ ، ۲۲ باینین ۲۵

درستی که ۲۲ با ۲۵ دور ، باید ۲۰ سیاه شود تا شکل برقرار شود

حذف ۲۰ : برادرش ۲۲ سیاه با یک فرزند قرمز ۲۵ است



فرزند دور برادر ۲۵ است

رنگ ۴۵ را به پدر پدر تغییر می دهیم

فرزند دور ۲۵ را سیاه می کنیم

پرنسپ به سبب دور ۲۲

