دانشکده‌ی علوم ریاضی

ساختمان داده‌ها ۲۶ آذر ۹۲

جلسه‌ی ۲۱: درخت قرمز سیاه

مدرّس: دکتر شهرام خزائی *نگارنده: سیاوش ریاحی و عرفان خانیکی*

۱ درخت قرمز سیاه

۱.۱ تعریف درخت قرمز سیاه

درخت قرمز سیاه، یک درخت دودویی جست‌وجو با خواص زیر است:

۱. هر گره این درخت، به رنگ قرمز یا سیاه است.

۲. ریشه‌ی درخت، سیاه است.

۳. رنگ هر برگ (NIL)، سیاه است.

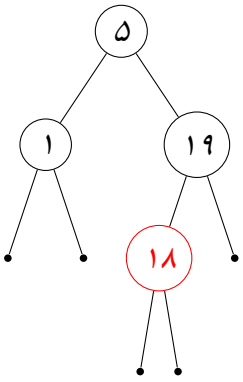
۴. پدر هر گره قرمز، سیاه است.

۵. در هر مسیر از گره x به هر نواده‌ی برگی، تعداد گره‌های سیاه برابر هستند.

۲.۱ درخت قرمز سیاه متمایل به چپ

درخت قرمز سیاه متمایل به چپ [۱]، درخت قرمز سیاهی است که هر گره قرمز، فرزند چپ پدرش باشد. از این پس به جای عبارت درخت قرمز سیاه متمایل به چپ، از عبارت درخت قرمز سیاه استفاده می‌کنیم.

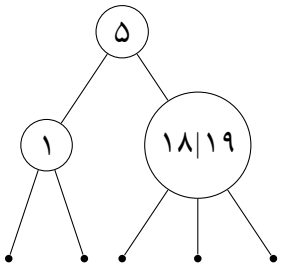
مثال ۱ *درخت زیر، یک درخت قرمز سیاه متمایل به چپ است:*



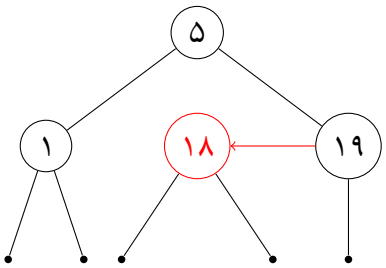
[۱] Left-Leaning-Red-black-tree

۱.۲.۱ نکاتی در باره‌ی درخت قرمز سیاه متمایل به چپ

۱. یک تناظر یک‌به‌یک بین درخت‌های قرمز سیاه متمایل به چپ و درخت‌های ۳ـ۲ وجود دارد. برای مثال، درخت قرمز سیاه بالا با درخت ۳ـ۲ زیر، هم ارز است.

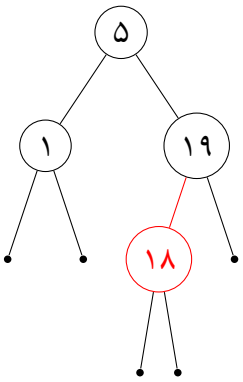


برای تبدیل این درخت ۳ـ۲ به درخت قرمز سیاه، می‌توان آن را به شکل زیر درنظر گرفت:



در واقع در درخت ۲ـ۳ گره قرمز با پدرش، تشکیل یک گره سه فرزندی می‌دهند.

۲. می‌توان علاوه بر رنگ کردن گره‌ها، یال‌ها را نیز به رنگ قرمز یا سیاه رنگ کرد؛ بدین صورت که یال بین گره قرمز و پدرش نیز قرمز است. به بیان دیگر اگر یال‌های قرمز را افقی نشان دهیم، تعداد یال‌های غیر افقی (سیاه) که در هر مسیر از ریشه به برگ‌ها قرار دارد، یکسان است. به عنوان مثال، درخت فوق به صورت زیر نشان داده‌می‌شود:



توجه داشته باشید که رنگ کردن یال‌ها، اطلاعات اضافه‌ای به درخت قرمز سیاه نمی‌افزاید. این کار صرفاً متمایل به چپ بودن درخت را به طور ملموس‌تری به ما نشان می‌دهد و در تعیین عملیاتی که بر روی درخت اعمال می‌کنیم، به ما کمک خواهدکرد.

۲ درج یک عنصر در درخت قرمز سیاه

حال که با تعریف درخت قرمز سیاه آشنا شدیم، باید بتوانیم عنصری را در درخت درج کنیم و در عین حال، درخت ما یک درخت قرمز سیاه متمایل به چپ باقی بماند.

۱.۲ مثال‌های درج عنصر در درخت قرمز سیاه

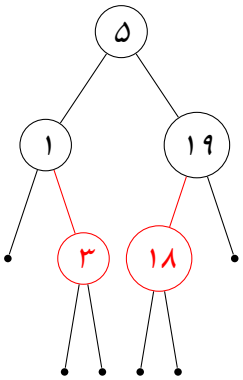
با ذکر چند مثال، حالات درج یک عنصر در درخت قرمز سیاه متمایل به چپ را بررسی می‌کنیم. به طور کلی، برای درج عنصری در درخت، ابتدا آن را با رنگ قرمز در نظر می‌گیریم و همانند درخت جست‌وجوی دودویی، عنصر را درج می‌کنیم. حال با انجام اعمالی، درخت را تغییر می‌دهیم تا ویژگی درخت قرمز سیاه متمایل به چپ حفظ شود. این اعمال عبارتند از:

۱. چرخش به چپ[۲] (حول یک گره)

۲. چرخش به راست[۳] (حول یک گره)

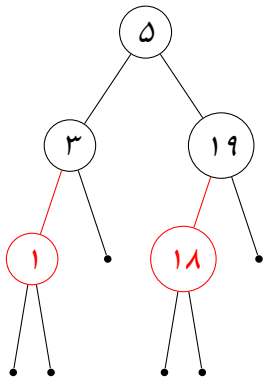
۳. جابجایی رنگ[۴] (یک گره و فرزندانش)

مثال ۲ *می‌خواهیم عدد ۳ را در درخت قرمز سیاهی که در بالا کشیده شده است، درج کنیم.*
پس از درج معمولی ۳ (مانند درخت جست‌وجوی دودویی) در درخت داریم:



درختی که در بالا کشیده‌شده‌است، یک درخت قرمز سیاه متمایل به چپ نیست. پس با متوازن کردن آن، به درخت قرمزسیاه متمایل به چپ زیر می‌رسیم:

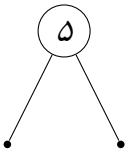
[۲] Left-Rotate(x)
[۳] Right-Rotate(x)
[۴] Flip-Color(x)



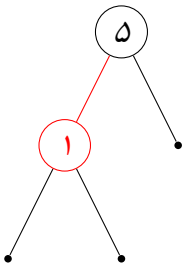
همان‌طور که دیدیم برای حفظ خواص درخت قرمز سیاه، باید اعمالی را روی درخت اجرا کنیم. در دو مثال بعد تمام حالات ممکن برای درج را بررسی می‌کنیم.

مثال ۳ *در این مثال می‌خواهیم درج کردن یک عنصر، در یک درخت تک عضوی را بررسی کنیم. به این منظور، یک درخت قرمز سیاه متمایل به چپ تک‌عضوی که مقدار ریشه‌ی آن ۵ است را در نظر گرفته، در حالت اول ۱ و در حالت دوم ۷ را در آن درج می‌کنیم.*

۱. *پس از درج معمولی ۱ (مانند درخت جست‌وجوی دودویی) در درخت زیر*

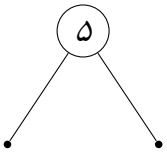


درخت زیر حاصل می‌شود:

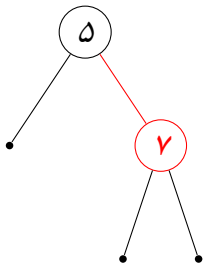


این درخت، یک درخت قرمز سیاه متمایل به چپ است و نیازی به تغییر ندارد.

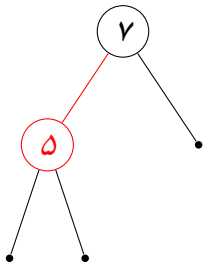
۲. *پس از درج معمولی ۷ در درخت زیر*



درخت زیر حاصل می‌شود:

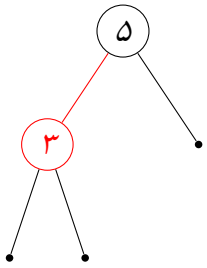


این درخت، درخت قرمز سیاه متمایل به چپ نیست. برای حفظ ویژگی درخت قرمز سیاه متمایل به چپ، باید گره قرمز در سمت چپ باشد، یا معادلاً یال قرمز به سمت چپ متمایل باشد. پس درخت را با چرخش به چپ حول گره ۵، اصلاح می‌کنیم.

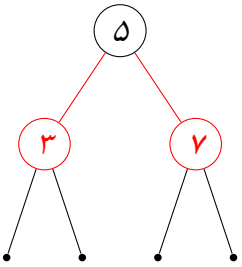


مثال ۴ *در این مثال می‌خواهیم درج کردن یک عنصر در یک درخت قرمز سیاه متمایل به چپ دو عضوی را بررسی کنیم.*

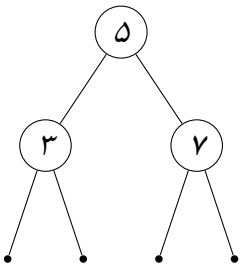
۱. *پس از درج معمولی ۷ در درخت زیر*



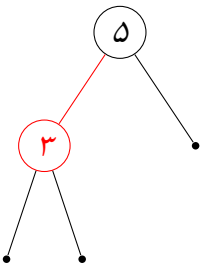
درخت زیر حاصل می‌شود:



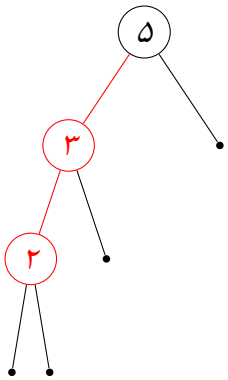
حال رنگ دو گره قرمز را به سیاه و رنگ گره سیاه را به قرمز تغییر می‌دهیم (در واقع عمل جابجایی رنگ عنصر x را فراخوانی می‌کنیم). سپس رنگ ریشه را به سیاه تغییر می‌دهیم.



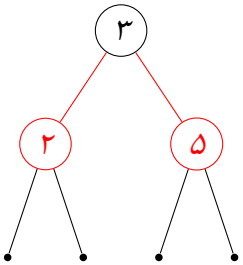
۲. پس از درج معمولی ۲ در درخت زیر



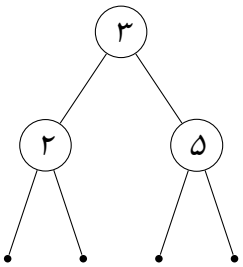
به این درخت می‌رسیم:



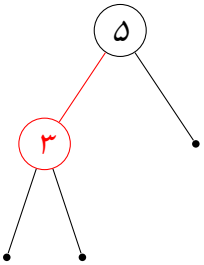
پس از اعمال گردش به راست حول ۵ داریم:



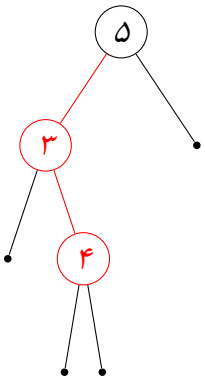
که پس از اعمال جابجایی رنگ بر گره ۳ و تغییر رنگ ریشه به سیاه، به درخت قرمز سیاه متمایل به چپ زیر می‌رسیم:



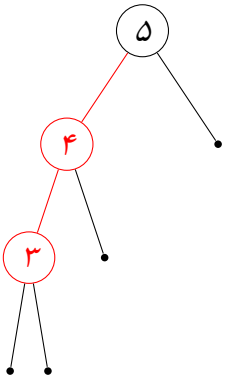
پس از درج معمولی ۴ در درخت زیر



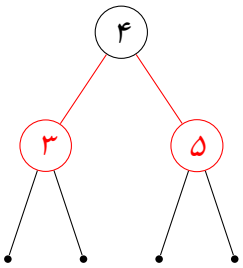
درخت زیر حاصل می‌شود:



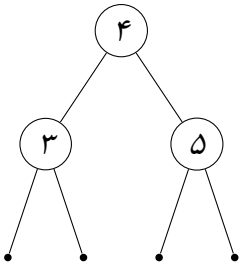
که پس از اعمال گردش به چپ حول گره ۳ به درخت زیر می‌رسیم:



که پس از اعمال گردش به راست حول گره ۵ به درخت زیر تبدیل می‌شود:



پس از اعمال جابجایی رنگ و تغییر رنگ ریشه به سیاه، درخت قرمز سیاه متمایل به چپ زیر حاصل می‌شود.

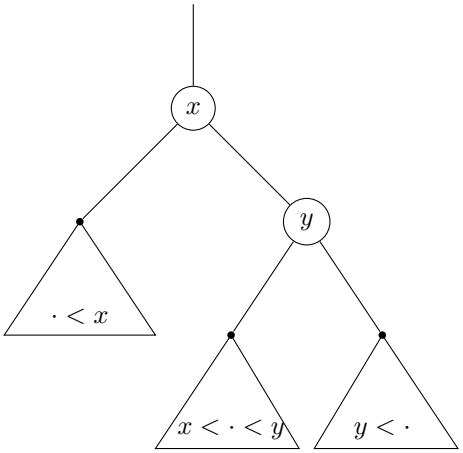


۳ تعریف اعمال چرخش به راست، چرخش به چپ و جابجایی رنگ

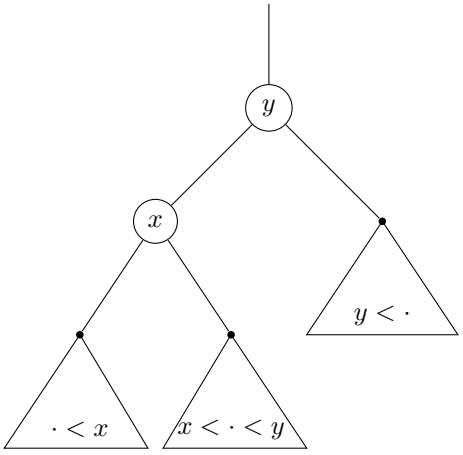
در حالت کلی اعمال LEFT-ROTATE(x) , RIGHT-ROTATE(x) و FLIP-COLOR(x) به صورت زیر تعریف می‌شوند:

۱. LEFT-ROTATE(x)

هدف در عمل LEFT-ROTATE(x) این است که در زیر درخت پایین، گره x فرزند چپ گره y بشود، ولی خاصیت درخت جست‌وجوی دودویی حفظ شود. برای این کار زیر درخت چپ گره y، به زیر درخت راست گره x و گره x، فرزند چپ گره y قرار می‌گیرد.

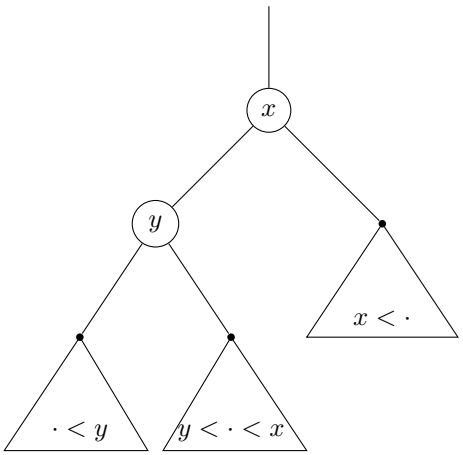


بعد از انجام LEFT-ROTATE(x) این زیر درخت حاصل می‌شود:

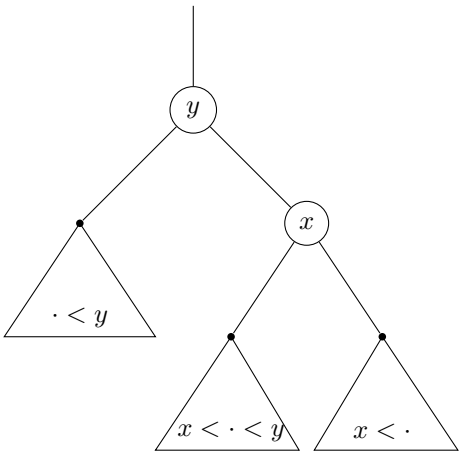


۲. RIGHT-ROTATE(x)

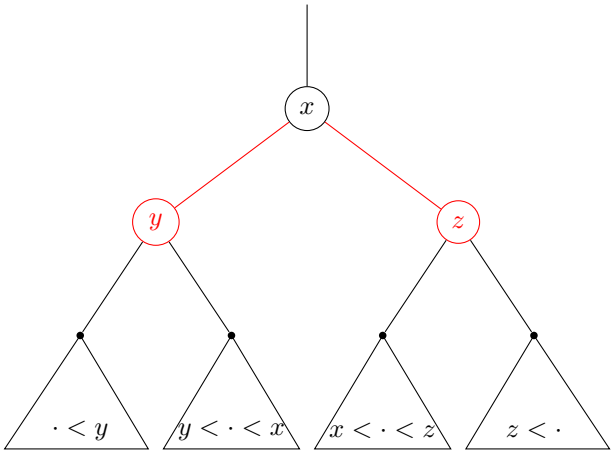
هدف در عمل RIGHT-ROTATE(x) این است که در زیر درخت پایین، گره x، فرزند راست گره y بشود، ولی خاصیت درخت جست‌وجوی دودویی حفظ شود. برای این کار زیر درخت راست گره y، به زیر درخت چپ گره x و گره x، فرزند راست گره y قرار می‌گیرد.



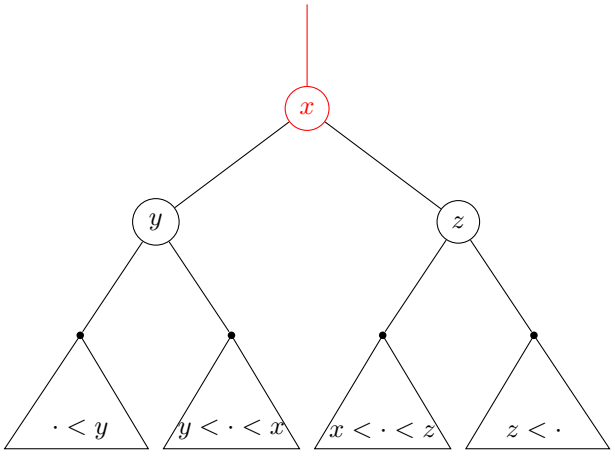
بعد از انجام عملیات درخت زیر حاصل می‌شود:



۳. FLIP-COLOR(x) هدف از انجام عمل FLIP-COLOR(x) این است که رنگ دو فرزند گره x به سیاه و رنگ گره x به قرمز تغییر کند.



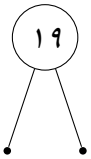
بعد از انجام عملیات، این زیر درخت حاصل می‌شود:



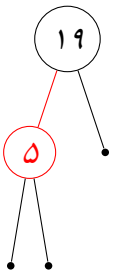
حال با ذکر مثالی، روند درج عناصر در درخت قرمز سیاه مایل به چپ را نشان می‌دهیم.
به طور خلاصه، RIGHT-ROTATE(x) را با R-R(x)، LEFT-ROTATE(x) را با L-R(x)، و FLIP-COLOR(x) را با F-C(x) نشان می‌دهیم.

مثال ۵ *اعداد* $\{۱۳, ۲۴, ۸, ۳, ۱۸, ۱, ۵, ۱۹\}$ *را به ترتیب در درخت قرمز سیاه متمایل به چپ، درج کرده و روند تغییر درخت را نشان می‌دهیم:*

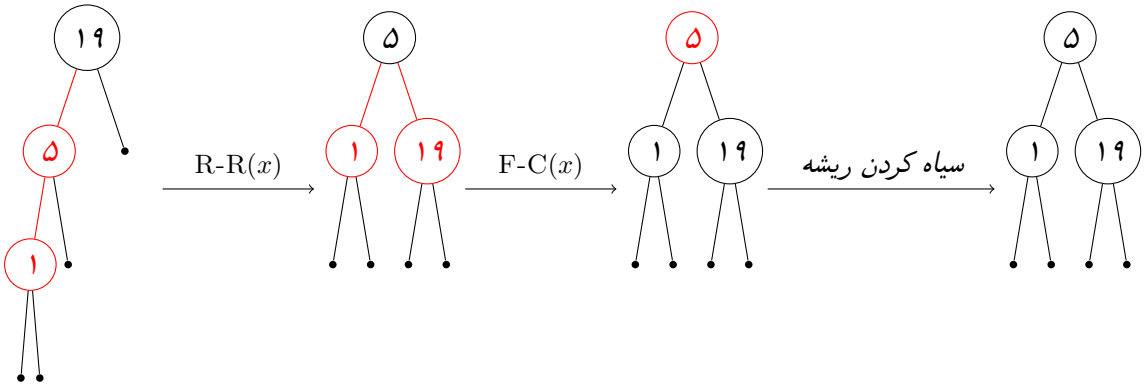
۱. درج ۱۹



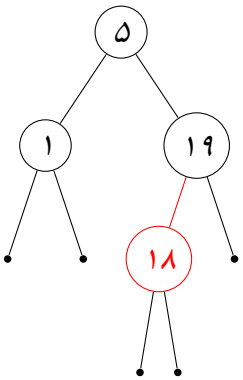
۲. درج ۵



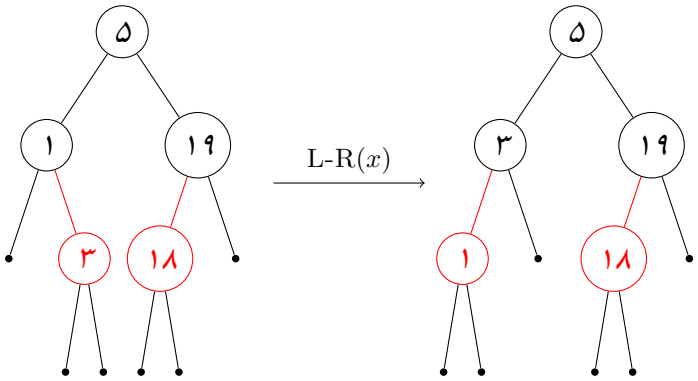
۳. درج ۱



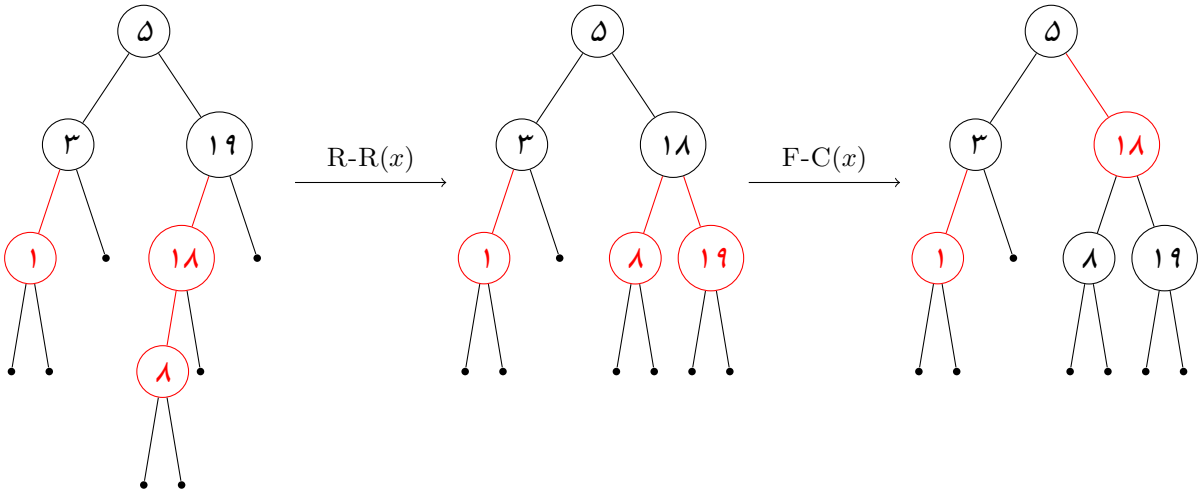
۴. درج ۱۸

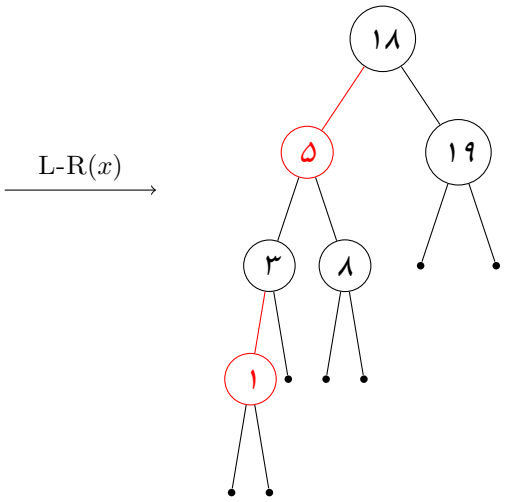

۵
۱
۱۹
۱۸

۵. درج ۳

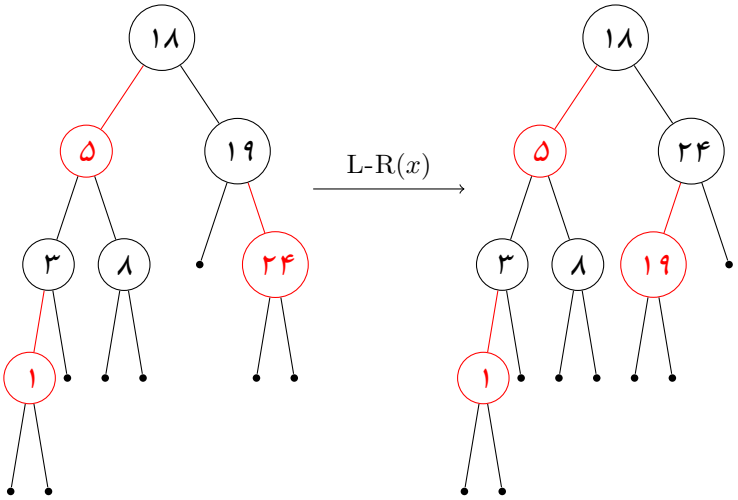

۵
۱
۱۹
۳
۱۸
L-R(x)
۵
۳
۱۹
۱
۱۸

۶. درج ۸

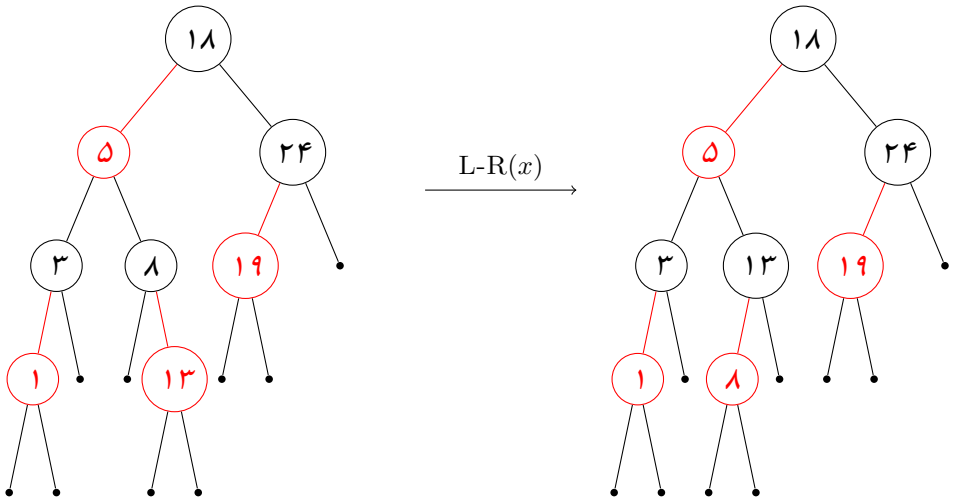

۵
۳
۱۹
۱
۱۸
۸
R-R(x)
۵
۳
۱۸
۱
۸
۱۹
F-C(x)
۵
۳
۱۸
۱
۸
۱۹



۷. درج ۲۴



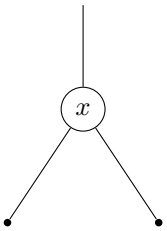
۸. درج ۱۳



۴ حالات درج یک عدد در درخت قرمز سیاه متمایل به چپ

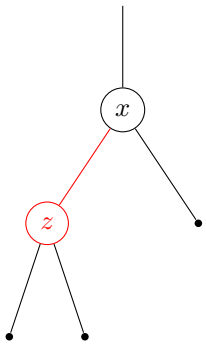
برای درج یک عدد در درخت قرمز سیاه متمایل به چپ، ابتدا گره را با رنگ قرمز در نظر می‌گیریم، حال بدون در نظر گرفتن رنگ گره‌ها، همانند درخت جست‌وجوی دودویی، گره مذکور را در درخت درج می‌کنیم. پنج حالت برای گره درج شده، با نام‌های A1 , A2 ، B1 , B2 , B3 وجود دارد که به طور بازگشتی قابل تبدیل به یک‌دیگر هستند. به عبارت دیگر با انجام ۳ عمل چرخش به چپ، چرخش به راست و جابجایی رنگ، ویژگی درخت قرمز سیاه متمایل به چپ را برای گره درج شده فراهم می‌کنیم. حال ممکن است پس از این اصلاحات، گره دیگری ویژگی موردنظر را نداشته باشد یا گره درج شده تغییر مکان داده و ویژگی مورد نظر را نداشته باشد. این گره (که مکان آن در الگورتم درج، مشخص است) نیز در یکی از این پنج حالت قرار می‌گیرد و قابل اصلاح است.
پس از درج گره z، یکی از حالات زیر را خواهد داشت:

حالت A پدر گره z، پیش از درج گره جدید، فرزندی نداشته و سیاه است. اگر گره پدر را x بنامیم، پس گره جدید، فرزند چپ یا راست آن است.

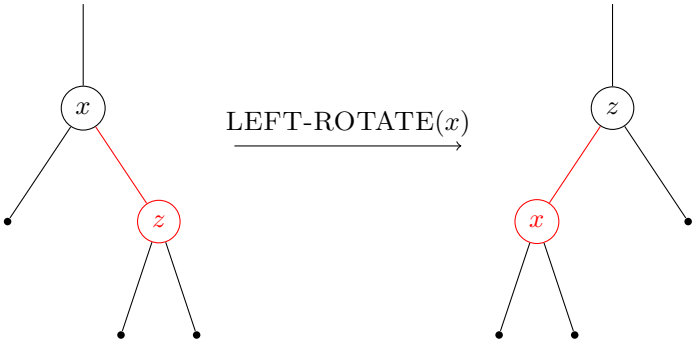


پس از درج گره جدید، یکی از دو حالت زیر پیش می‌آید:

حالت A1 ($z \leq x$)
در این حالت، گره در جای مناسب خود قرار دارد و نیازی به تغییر ندارد.



حالت A2 $(z > x)$
در این حالت، با یک چرخش به چپ برروی گره x (به عبارت دیگر فراخوانی LEFT-ROTATE(x)) درخت اصلاح می‌شود.

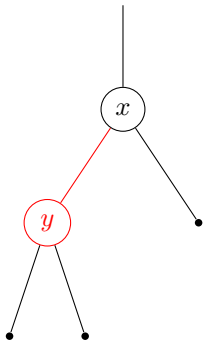


حالت B گره پدر گره z، پیش از درج این گره، یا یک گره قرمز مانند y است یا یک گره سیاه (با یک فرزند چپ قرمز) مانند x است. حال سه حالت ممکن است پیش بیاید:

۱. گره درج شده، فرزند راست گره x باشد.

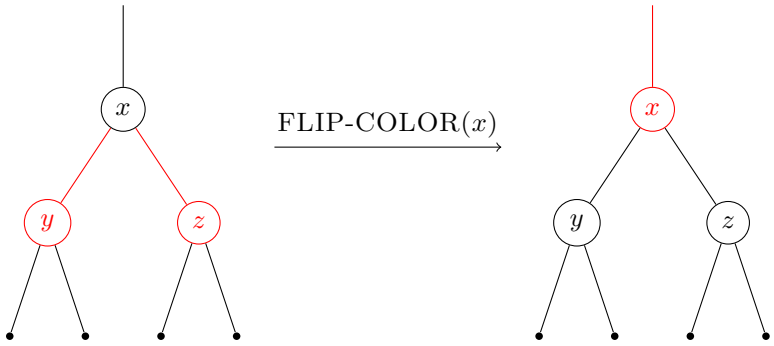
۲. گره درج شده، فرزند چپ گره y باشد.

۳. گره درج شده، فرزند راست گره y باشد .

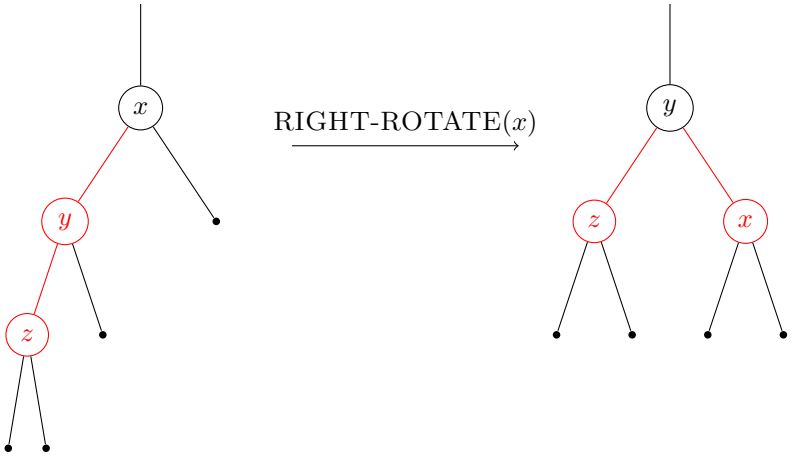


حالت B1 $(z > x)$ در این حالت، با فراخوانی جابجایی رنگ روی گره x (به عبارت دیگر فراخوانی FLIP-COLOR(x)) گره‌های z و y اصلاح می‌شود. اگر گره x ریشه باشد، با تغییر رنگ آن به سیاه اصلاح

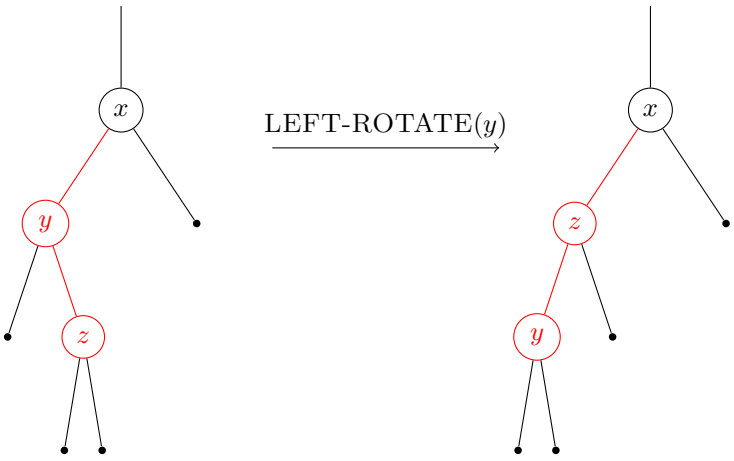
می‌شود. درغیر این صورت، گره x در یکی از حالات A یا B قرار می‌گیرد که به طور بازگشتی قابل اصلاح است.



حالت B2 ($z \leq y$) در این حالت، با یک چرخش به راست برروی z، به حالت B1 می‌رسیم.



حالت B3 ($y < z \leq x$) در این حالت، با یک چرخش به چپ برروی y، به حالت B2 می‌رسیم.



در تقسیم بندی فوق، تمام حالات ممکن برای درج عددی در درخت قرمز سیاه متمایل به چپ بررسی شده و می‌توان به راحتی عددی را در درخت درج کرد.

در حالت کلی می‌توان از نمودار زیر برای درج استفاده نمود:

A
A1
پایان
A2
LEFT-ROTATE
B3
LEFT-ROTATE
B
B2
B1
RIGHT-ROTATE
یکی از ۵ حالت قبل
یا پایان
FLIP-COLOR

۵ الگوریتم چرخش به چپ، چرخش به راست و جابجایی رنگ

برای آن که الگوریتم درج یک عنصر در درخت قرمز سیاه متمایل به چپ را بیان کنیم، ابتدا باید الگوریتم اعمال چرخش به چپ، چرخش به راست و جابجایی رنگ را پیاده سازی کنیم. توضیحات این اعمال در بخش ۳ آورده شده است و در این جا تنها به بیان الگوریتم‌ها اکتفا می‌کنیم.

۱. الگوریتم عمل چرخش به چپ:

Algorithm 1 LEFT-ROTATE

```
1:  function LEFT-ROTATE(T, x)
2:      y ← x.right
3:      x.right ← y.left
4:      if y.left ≠ NIL then
5:          y.left.parent ← x
6:      y.parent ← x.parent
7:      if x.parent = NIL then
8:          T.root ← y
9:      else
10:         if x = x.parent.left then
11:             x.parent.left ← y
12:         else
13:             x.parent.right ← y
14:     y.left ← x
15:     x.parent ← y
```

۲. الگوریتم عمل چرخش به راست:

الگوریتم عمل چرخش به به راست، مشابه چرخش به چپ پیاده سازی می‌شود.

Algorithm 2 RIGHT-ROTATE

```
1:  function RIGHT-ROTATE(T, x)
2:      y ← x.left
3:      x.left ← y.right
4:      if y.right ≠ NIL then
5:          y.right.parent ← x
6:      y.parent ← x.parent
7:      if x.parent = NIL then
8:          T.root ← y
9:      else
10:         if x = x.parent.left then
11:             x.parent.left ← y
12:         else
13:             x.parent.right ← y
14:     y.right ← x
15:     x.parent ← y
```

۳. الگوریتم عمل جابجایی رنگ:

Algorithm 3 FLIP-COLOR

```
1: function FLIP-COLOR(T, x)
2:     x.color ← BLACK
3:     x.left.color ← RED
4:     x.right.color ← RED
```

۶ الگوریتم درج عددی در درخت قرمز سیاه متمایل به چپ

برای پیاده‌سازی الگوریتم درج، کافی است در حلقه‌ای حالت B2,B1 و B3 را بررسی کنیم تا گره از این حالات خارج شود و به یکی از حالات A1 و A2 برسد.
این الگوریتم به صورت زیر است:

Algorithm 4 LLRBT-INSERT

```
1: function LLRBT-INSERT(T, z)
2:     BINARYSEARCHTREE-INSERT(T, z)
3:     z.color ← RED
4:     while z ≠ T.root or B1 or B2 or B3 do
5:         if B2 or B3 then
6:             if B3 then
7:                 z ← z.parent
8:                 LEFT-ROTATE(T, z)
9:             z ← z.parent
10:            RIGHT-ROTATE(T, z.parent)
11:        z ← z.parent
12:        FLIP-COLOR(T, z)
13:    if A2 then
14:        z ← z.parent
15:        LEFT-ROTATE(T, z)
16:    T.root.color ← BLACK
```

در این الگوریتم، ابتدا در حلقه حالات B2,B1 و B3 بررسی می‌شود و تغییرات لازم روی درخت اعمال می‌شود. این حلقه تا زمانی که گره z از حالات B2,B1 و B3 خارج شود، تکرار می‌شود. پس از رفع کردن حالات B2,B1 و B3 اگر گره z در حالت A2 قرار داشته باشد، با گردش به چپ این حالت نیز رفع می‌شود. در نهایت، رنگ ریشه به سیاه تغییر می‌کند؛ زیرا ممکن است که در حلقه‌ی الگوریتم تا ریشه پیش رفته و رنگ ریشه با FLIP-COLOR(T, z) که $z = T.root$ تغییر کرده‌باشد.