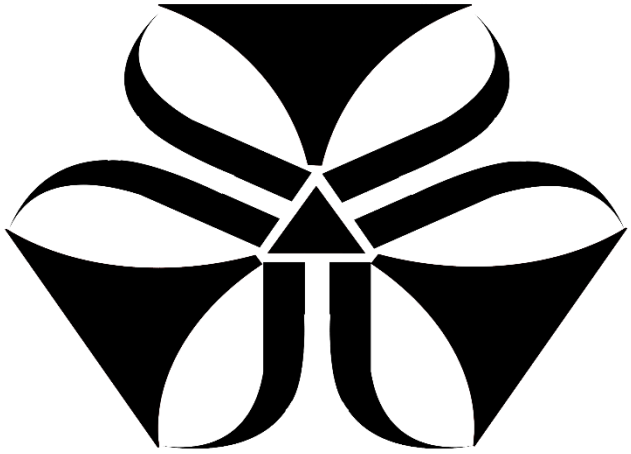


پروژه پایانی درس ساختمان های داده

استاد:دکتر امیری

موضوع پروژه:پیاده سازی درخت red or black

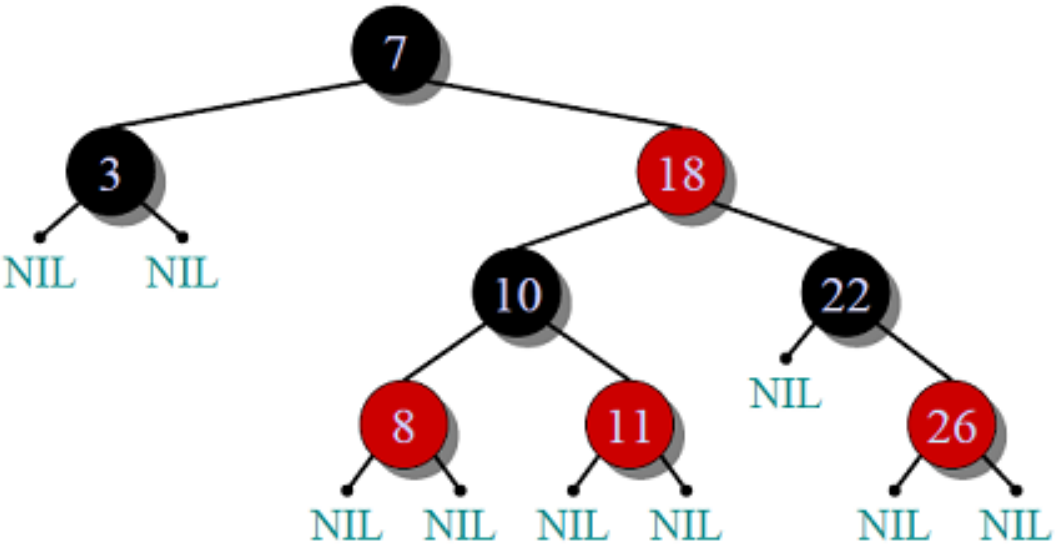
موضوع این پروژه در ارتباط با طراحی و پیاده سازی درخت red or black است.درخت red or black نوعی درخت جستجوی دودویی(bst) است که مانند بقیه درخت های جستجوی دودویی ویژگی self balancing یا تعادل خودکار را دارد.در این درخت 4 قانون اصلی وجود دارد.

1.هر گره یک رنگ دارد.(سیاه یا قرمز)

2.root یا ریشه درخت همیشه مشکی است.

3.دو گره قرمز هیچ گاه نمی توانند در همسایگی یکدیگر قرار گیرند.(یک گره قرمز نمی تواند به عنوان پدر یا فرزند گره قرمز دیگری را بپذیرد)

4.هر مسیر از ریشه تا null دارای تعداد یکسانی از گره های مشکی است.



انواع عمیات بر روی درخت red or black :

Insert ,delete , max , min , search , successor , predecessor

:Insertion

عملیات های insertion علاوه بر خود عملیات باعث تغییراتی در درخت می شوند که از دو متد برای حفظ تعادل یا balance بودن درخت استفاده می کنیم.

1.recoloring

2.Rotating

ابتدا از recoloringاستفاده می کنیم و اگر در ادامه جواب نگرفتیم از rotation استفاده می کنیم.در ادامه الگوریتم کلی را بررسی می کنیم.الگوریتم دو case متفاوت دارد که بسته به رنگ گره پدر اتفاق می

افتد.اگر رنگ عمو(uncle)(به گره ای گفته می شود که هم سطح گره پدر است)) قرمز بود recoloringرا انجام می دهیم و اگر رنگ پدر مشکی بود rotationو یا recoloringرا انجام می دهیم.

رنگ یک گره nullره همیشه مشکی در نظر می گیریم.

فرض کنید x گره ای باشد که می خواهیم به درخت اضافه کنیم.

1.ابتدا عملیات استاندارد اضافه کردن به درخت دودویی(standard BST insertion) را انجام می دهیم و رنگ گره تازه اضافه شده را قرمز می کنیم.

2.اگر xدر ریشه قرار گرفت رنگ x را سیاه می کنیم.

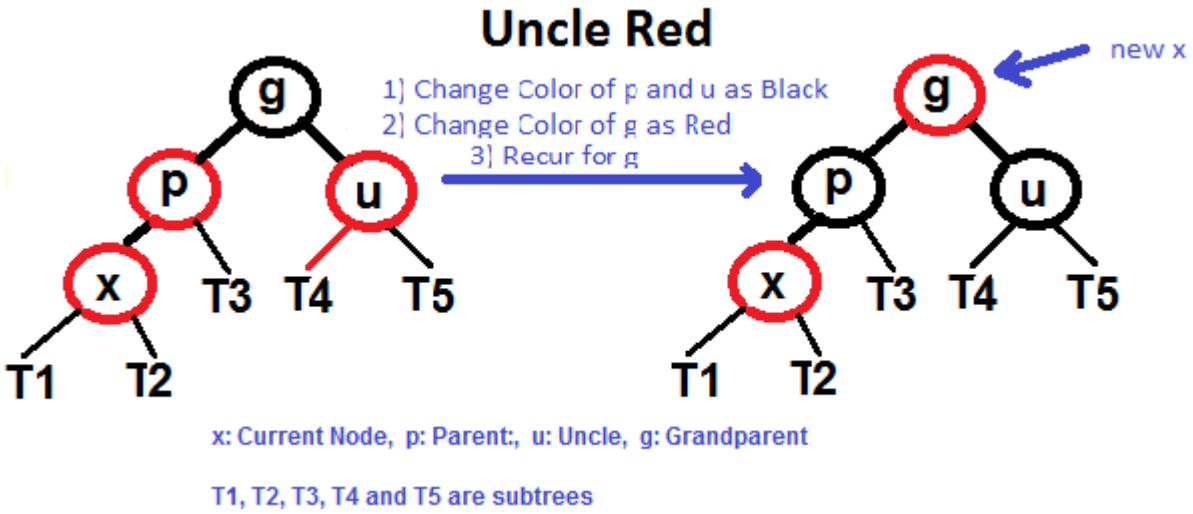
3.اگر xدر ریشه قرار نگرفت و رنگ پدرش سیاه نبود عملیات های زیر را انجام می دهیم:

a)اگر عموی x قرمز بود

1.رنگ پدر و رنگ عمو را به مشکی تغییر می دهیم

2.رنگ پدربزرگ گره(پدر پدر گره) را قرمز می کنیم

3.عمیالت های 2 و 3 را برای پدر بزرگ گره به عنوان xای جدید انجام می دهیم.



b)اگر عموی x مشکی باشد:

در این مورد چهار حالت برای x اتفاق می افتد.(بسته به وضعیت گره پدر pو یا رنگ گره پدربزرگ g)

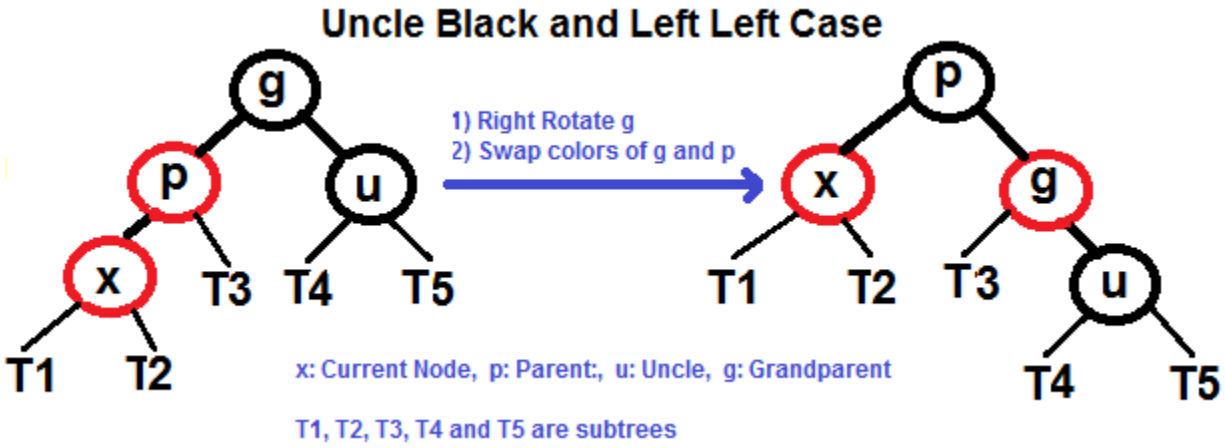
i)اگر pفرزند چپ g و x فرزند چپ p باشد.(left left case)

ii) اگر pفرزند چپ g و x فرزند راست p باشد.(left right case)

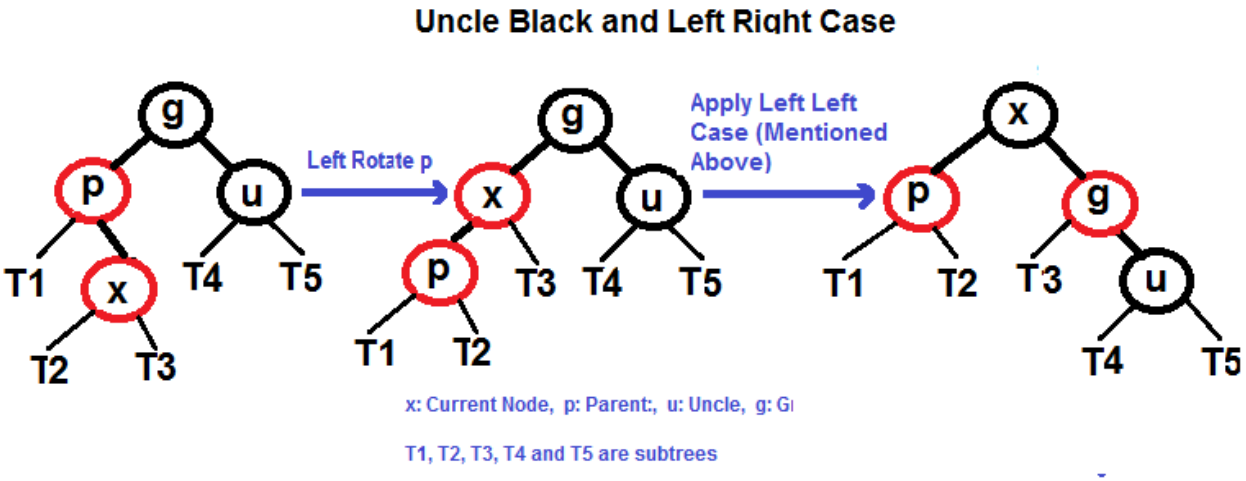
iii)بر عکس مورد i (right right case)

iv)برعکس مورد ii (right left case)

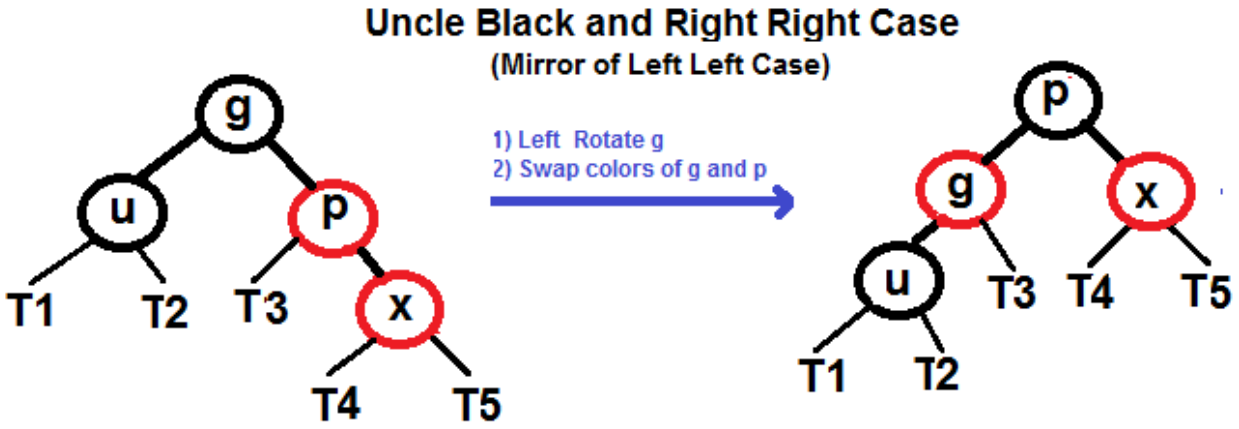
Left Left Case



Left Right Case



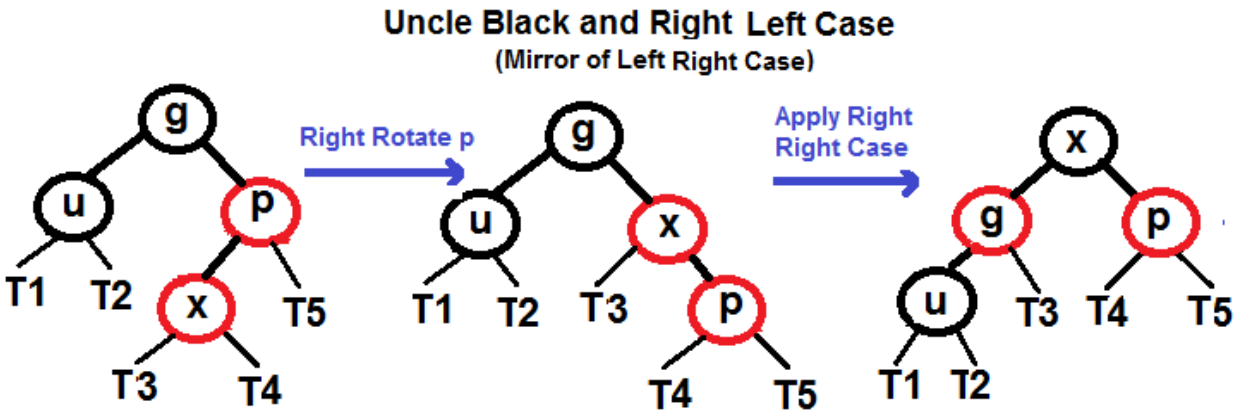
Right Right Case



x: Current Node, p: Parent:, u: Uncle, g: Grandparent

T1, T2, T3, T4 and T5 are subtrees

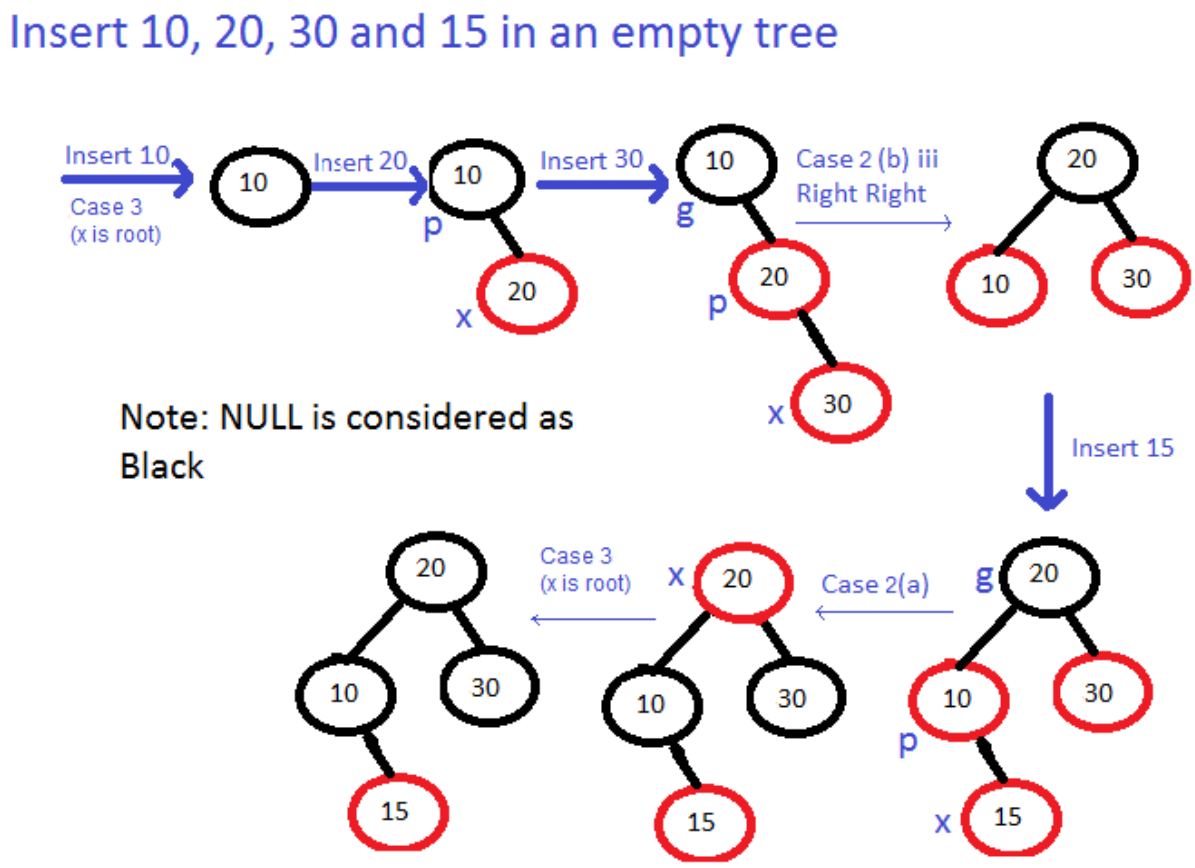
Right Left Case



x: Current Node, p: Parent:, u: Uncle, g: Grandparent

T1, T2, T3, T4 and T5 are subtrees

نمونه ای از عملیات insertion:



Deletion:مشابه عملیات insertion از recoloring و rotation برای حفظ ساختار درخت استفاده می کنیم.

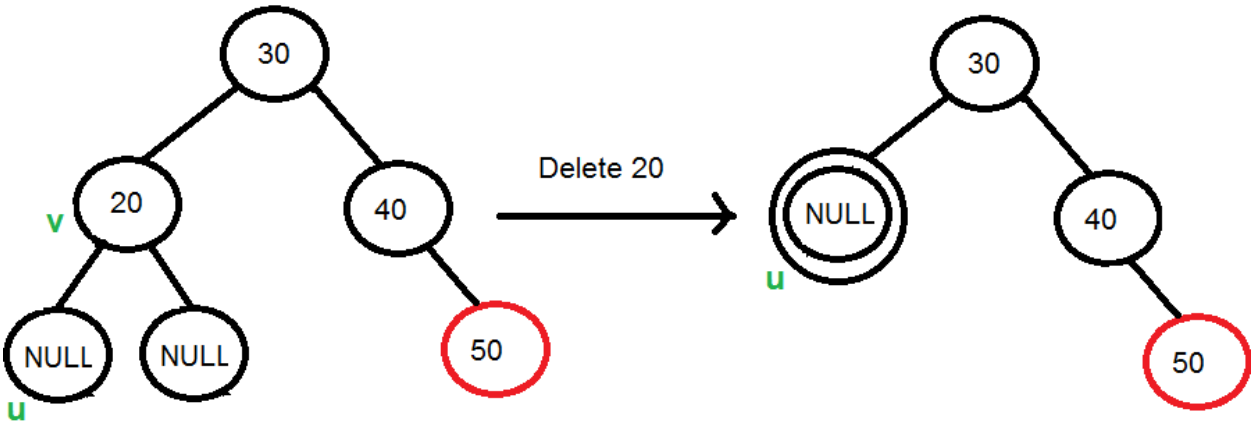
در ادامه الگوریتم deletionرا بررسی می کنیم:

1) عملیات حذف استاندارد جستجوی دودویی (standard BST deletion) را انجام می دهیم.وقتی این عملیات را انجام می دهیم همیشه یک برگ یا یک گره که فقط یک فرزند دارد دیلیت می شود.بنابراین فقط لازم است حالاتی را بررسی کنیم که گره مورد نظر برگ است یا فقط یک فرزند دارد.v را به عنوان گره ای درنظر می گیریم که قصد حذف کردن آنرا داریم و u به عنوان فرزند آن گره که قرار است درجای آن قرار بگیرد.

2) حالت اول:u یا v قرمز باشند,رنگ فرزند جابجا شده را مشکی می کنیم.توجه کنید که u و v نمی توانند همزمان قرمز باشند چون همسایه هم هستند و طبق قوانین پایهred or black دو گره قرمز نمی توانند در همسایگی یکدیگر قرار بگیرند.

3)اگر هم u و هم v سیاه باشند.

3.1)u را دوبار سیاه(double black) می کنیم.حالا باید u را که دوبار سیاه شده تبدیل کنیم به یک گره سیاه.



When 20 is deleted, it is replaced by a NULL, so the NULL becomes double black.
Note that deletion is not done yet, this double black must become single black

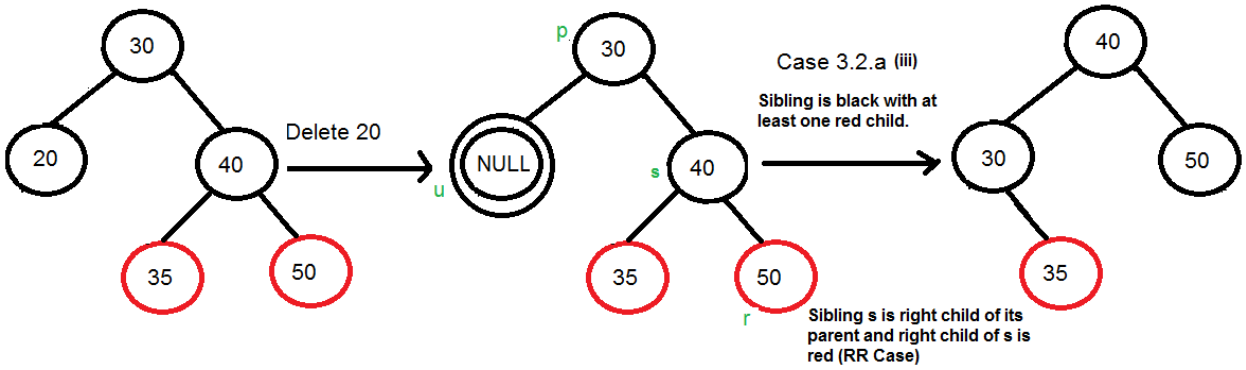
3.2)مراحل زیر برای گره u که اکنون دوبار سیاه شده و در ریشه قرار ندارد انجام می دهیم.فرزند آنرا به عنوان s در نظر می گیریم.

a)اگر s سیاه باشد و حداقل یکی از فرزندان آن قرمز باشند.در این حالت عملیات rotation را انجام می دهیم.به این صورت که فرزند قرمز s را به عنوان r در نظر می گیریم و و بسته به موقعیت sو r چهار حالت اتفاق می افتد.

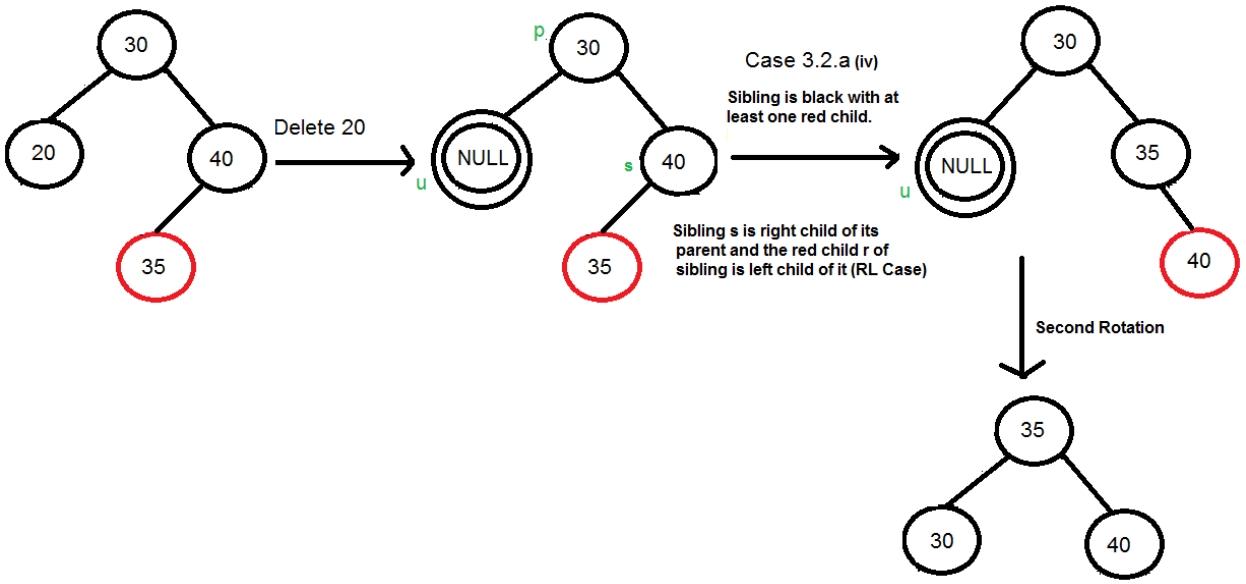
i) s فرزند چپ پدرش باشد و r فرزند چپ s و یا این که دو فرزند s قرمز باشند.(left left case)

ii) sفرزند چپ پدرش باشد و r فرزند راست باشد.(left right case)

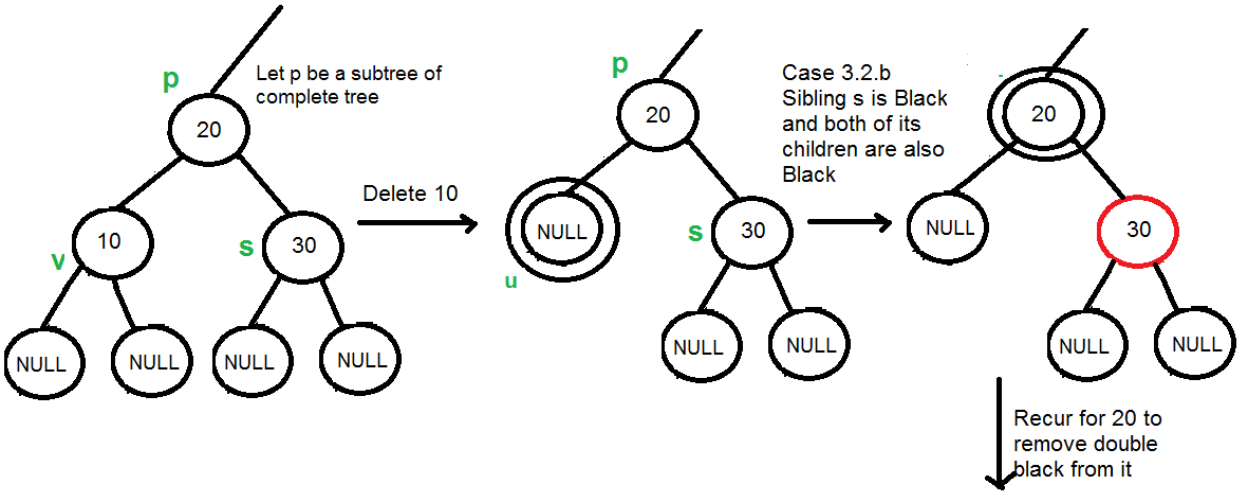
iii) s فرزند راست پدرش باشد و r فرزند راست s و یا این که دو فرزند s قرمز باشند. (right right case)



iv) sفرزند راست پدرش باشد و یا r فرزند چپ s باشد.(left right case)



b) اگ s سیاه بود recoloring را انجام می دهیم و همینکار را برای پدر نیز انجام می دهیم اگر پدر مشکی بود.

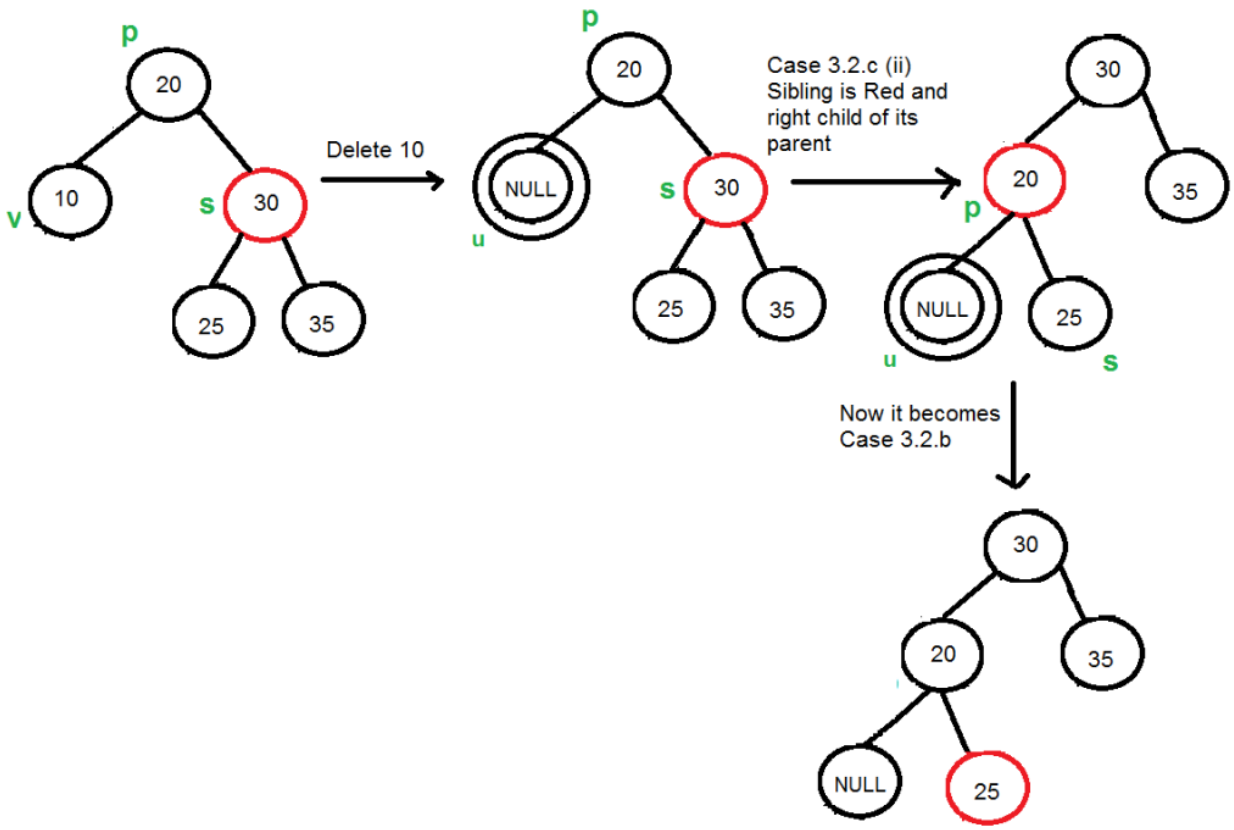


در این حالت اگر پدر قرمز بود احتیاجی به انجام دادن الگوریتم برای پدر احتیاجی نداریم.به سادگی پدر را مشکی می کنیم.

c)اگر s قرمز بود rotation را انجام می دهیم تا فرزند قدیمی را بالا بیاوریم , سپس فرزند قدیمی و پدرش recolor می کنیم.دو حالت به وجود می آید.

i) s فرزند چپ پدرش باشد, در این حالت p را right rotate می کنیم.

ii) s فرزند راست پدرش باشد, در این حالت p را left rotate می کنیم.



3.3) اگر u ریشه باشد آنرا سیاه می کنیم (single black).