فهرست مطالب

۱ آشنایی با درخت های AVL

۱.۱ سرفصل ها

۱. یادآوری درخت جستجوی دودویی

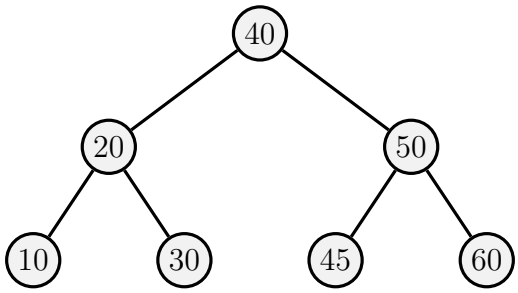
۲. چگونه می توان درخت جستجوی دودویی را بهینه تر کرد

۳. درخت AVL چیست

۴. چرخش های در درخت AVL

۵. چگونگی ساخت درخت AVL

۲.۱ یادآوری درخت جستجوی دودویی



همانطور که قبلاً مطالعه کریدم درخت جستجوی دودویی درختی است که :

- مقدار تمام اعضای سمت چپ هر نود از مقدار آن نود کمتر است
- مقدار تمام اعضای سمت راست هر نود از مقدار آن نود بیشتر است

برای پیدا کردن عناصر به بهینه ترین روش از درخت جستجوی دودویی استفاده می کنیم و بیشترین تعداد مقایسه برای پیدا کردن یک عنصر بستگی به ارتفاع آن درخت دارد .

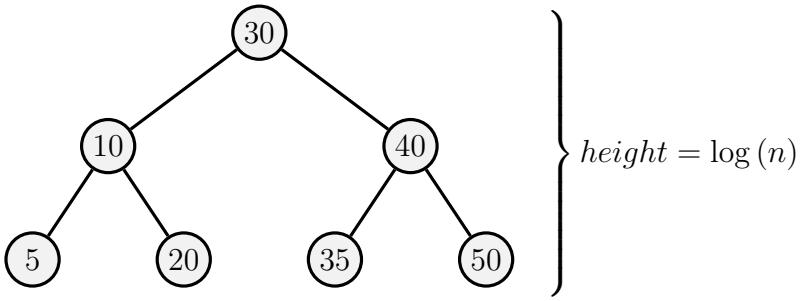
$$\text{ارتفاع درخت جستجوی دودویی} \Rightarrow \begin{cases} Minimum \Rightarrow \log(n) \\ Maximum \Rightarrow n \end{cases}$$

۲ نمونه ای از ساخت درخت جستجوی دودویی

اگر ترتیب ورودی اعداد برای ساخت درخت جستجوی دودویی به صورت

$$\text{keys}: 30, 40, 10, 50, 20, 5, 35$$

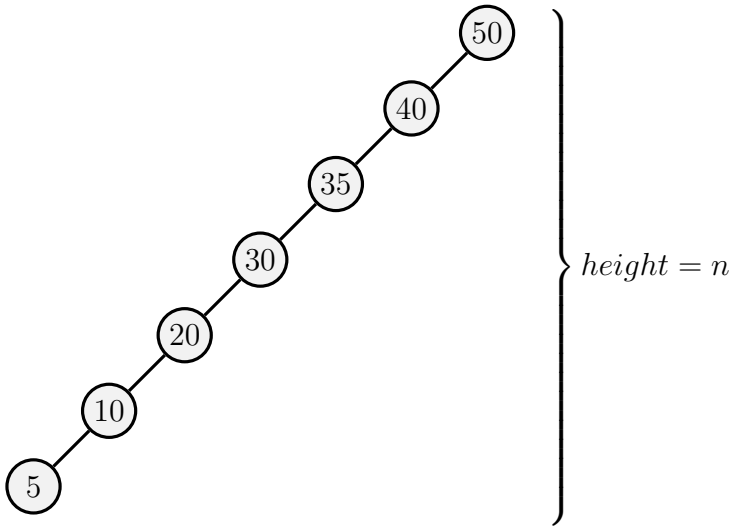
باشد ، آنگاه درخت حاصل به صورت زیر خواهد بود ، همانطور که مشاهده می کنید در این حالت ارتفاع درخت جستجوی دودویی برابر با بهترین حالت خود یعنی $\log(n)$ خواهد بود



در صورتی که ترتیب ورودی اعداد را به شکل

$$\text{keys}: 50, 40, 35, 30, 20, 10, 5$$

داشته باشیم ، آنگاه درخت به دست آمده به شکل زیر می شود ، در این حالت ما بیشترین ارتفاع درخت را داریم و عملکرد درخت ما مشابه با لیست پیوندی خواهد بود .



ارتفاع درخت جستجوی دودویی به این بستگی دارد که ما عناصر را چگونه وارد کنیم

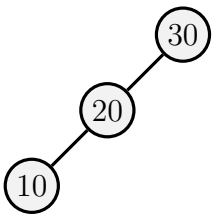
۳ آیا می توانیم درخت جستجوی دودویی را بهینه تر کنیم؟

فرض کنید که سه عنصر مثل

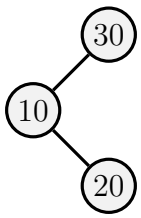
$$30, 20, 10$$

داریم ، آنگاه بسته به اینکه چطور عناصر را وارد کنیم شکل های زیر از درخت جستجوی دودویی حاصل می شود .

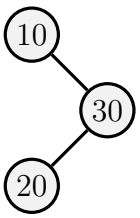
30, 20, 10



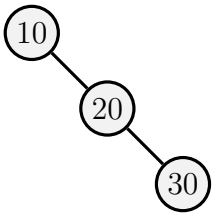
30, 10, 20



10, 30, 20

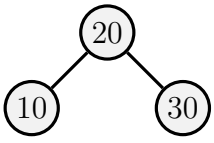


10, 20, 30



20, 10, 30

20, 30, 10



آیا راه حلی برای تبدیل شکل های ۱ تا ۴ به شکل ۵ وجود دارد ؟
جواب : بله ، با تعریف چرخش ها !

۱.۳ انواع چرخش ها

۱.۱.۳ LL-Rotation

۲.۱.۳ RR-Rotation

۳.۱.۳ RL-Rotation

LR-Rotation ۴.۱.۳

۴ درخت AVL چیست ؟

درخت AVL یک درخت جستجوی دودویی است که بین زیر درخت راست هر نود با زیر درخت چپ همان نود از نظر ارتفاع توازن وجود دارد .
برای ایجاد توازن در درخت جستجوی دودویی ما معیاری را به نام لاتین balance factor یا معیار توازن ایجاد می کنیم .

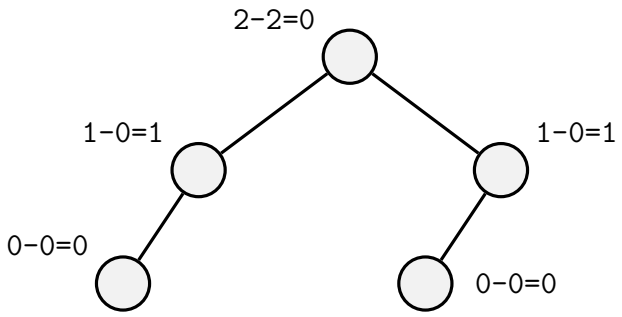
$$\text{balance factor} = \text{ارتفاع زیر درخت راست - ارتفاع زیر درخت چپ}$$

برای وجود توازن در هر نود درخت باید قوانین زیر برقرار باشند .

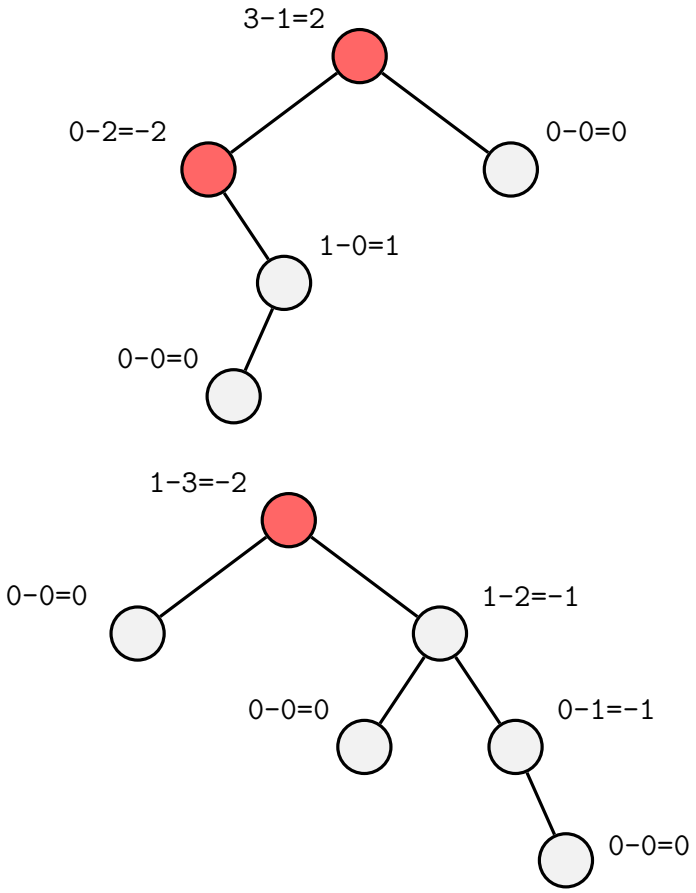
$$b_f = h_l - h_r = \{-1, 0, 1\}$$
$$|b_f| = |h_l - h_r| \leq 1$$

۱.۴ نمونه هایی از محاسبه ی balance factor هر نود

همانطورکه در شکل زیر مشاهده می شود balance factor تمام نود ها بین -1 و 1 قرار دارد بنابراین درخت ما متوازن است .

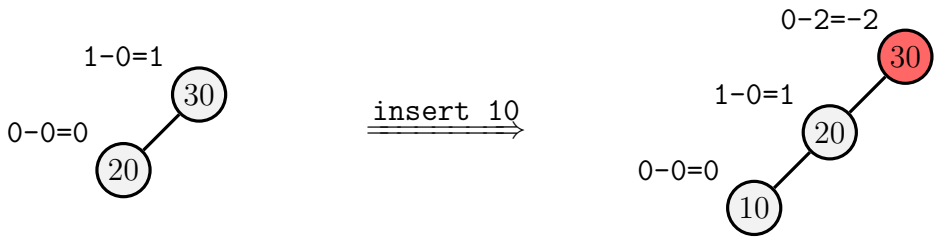


در شکل های زیر نود های رنگی نامتوازن شده اند زیرا balance factor آنها بین -1 و 1 قرار ندارد .



LL-Rotation ۲.۴

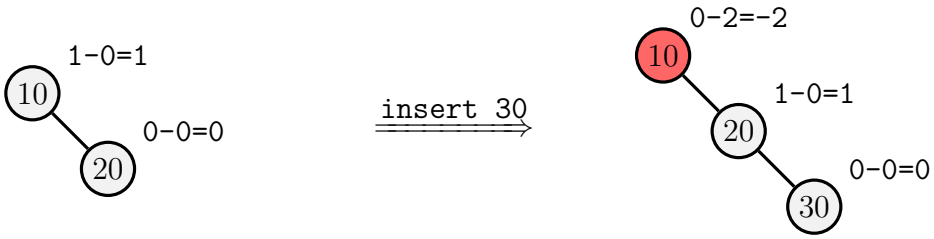
در شکل زیر پس از اینکه ما عنصری با مقدار 10 را به درخت اضافه کردیم ، نود با مقدار 30 به حالت نامتوازن در آمده است .



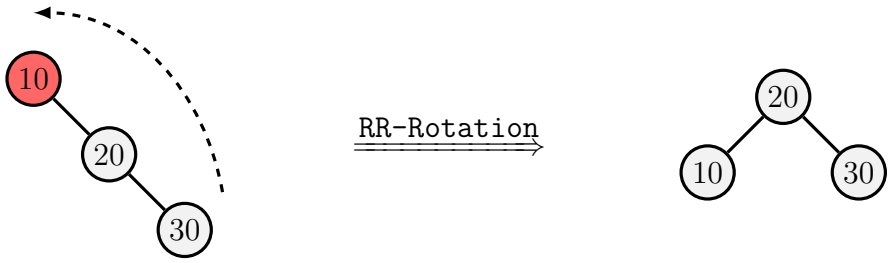
عنصر 30 نامتوازن شده است زیرا ما عنصر 10 را در سمت چپ و دوباره سمت چپ اضافه کردیم بنابراین برای رفع نامتوازنی باید LL-Rotation بزنیم

RR-Rotation ۳.۴

در شکل زیر پس از اینکه ما عنصری با مقدار 30 را به درخت اضافه کردیم ، نود با مقدار 10 به حالت نامتوازن در آمده است .

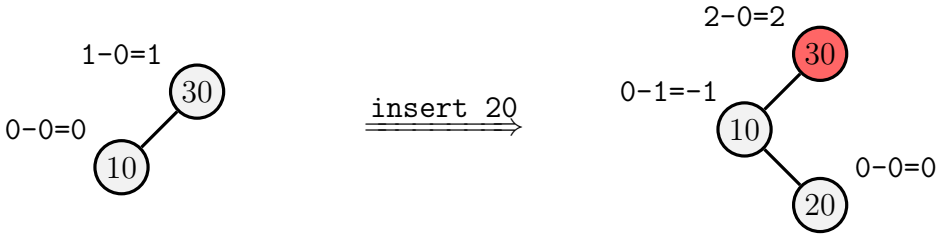


عنصر 10 نامتوازن شده است زیرا ما عنصر 30 را در سمت راست و دوباره سمت راست اضافه کردیم بنابراین برای رفع نامتوازنی باید RR-Rotation بزنیم

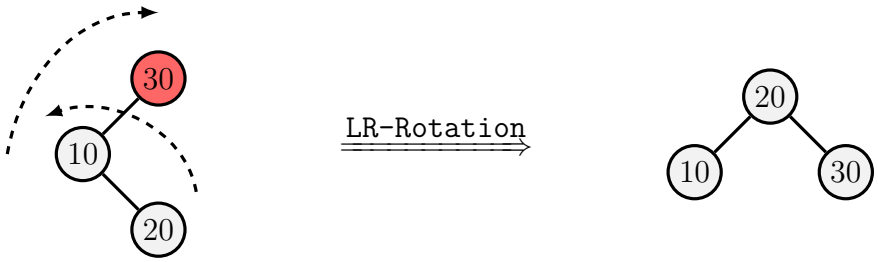


LR-Rotation ۴.۴

در شکل زیر پس از اینکه ما عنصری با مقدار 20 را به درخت اضافه کردیم ، نود با مقدار 30 به حالت نامتوازن در آمده است .

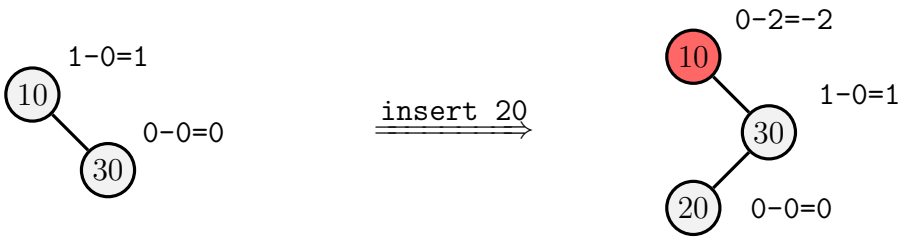


عنصر 30 نامتوازن شده است زیرا ما عنصر 20 را در سمت چپ و سپس سمت راست اضافه کردیم بنابراین برای رفع نامتوازنی باید LR-Rotation بزنیم

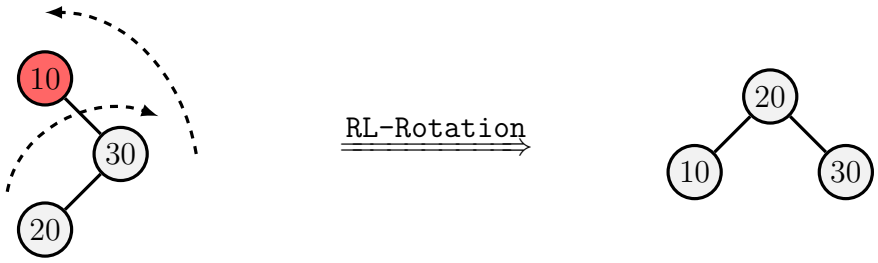


RL-Rotation ۵.۴

در شکل زیر پس از اینکه ما عنصری با مقدار 20 را به درخت اضافه کردیم ، نود با مقدار 10 به حالت نامتوازن در آمده است .



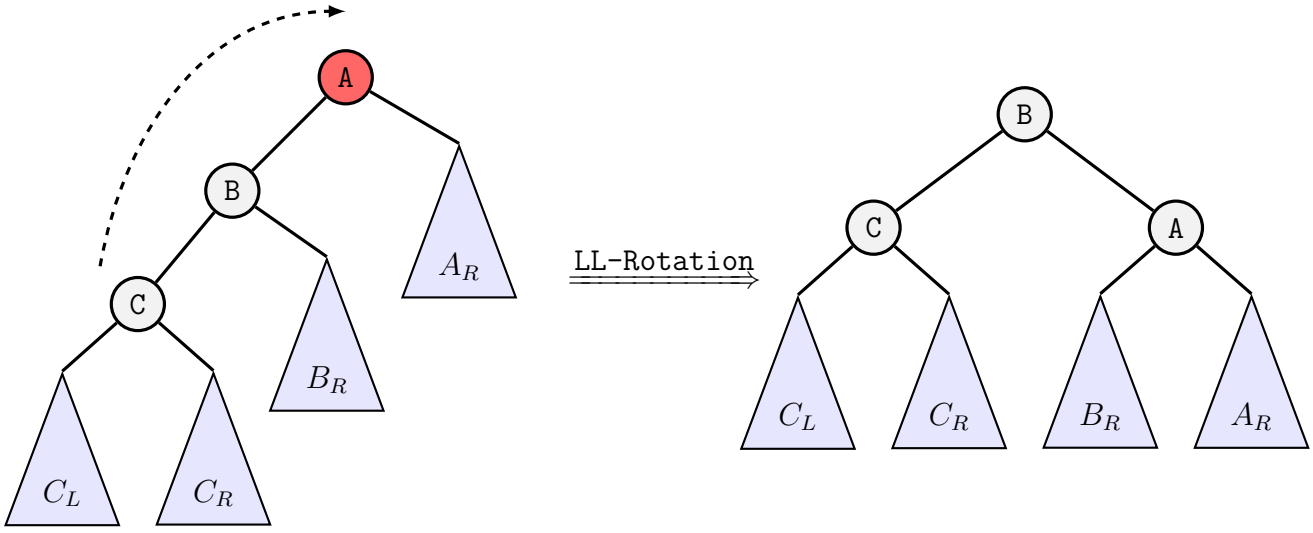
عنصر 10 نامتوازن شده است زیرا ما عنصر 20 را در سمت راست و سپس سمت چپ اضافه کردیم بنابراین برای رفع نامتوازنی باید RL-Rotation بزنیم



۵ چرخش با وجود زیر درخت ها

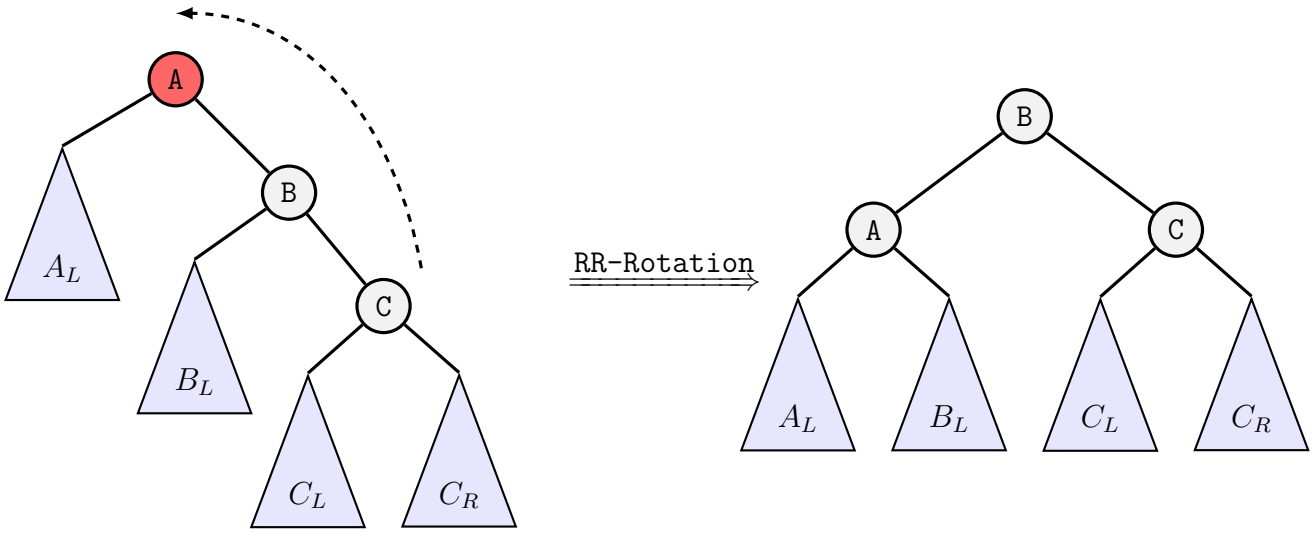
۱.۵ چرخش LL-Rotation با وجود زیر درخت ها

در چرخش LL-Rotation عناصر زیر درخت B_R که مقدار کمتری از عنصر A دارند در سمت چپ این نود قرار می گیرند .



۲.۵ چرخش RR-Rotation با وجود زیر درخت ها

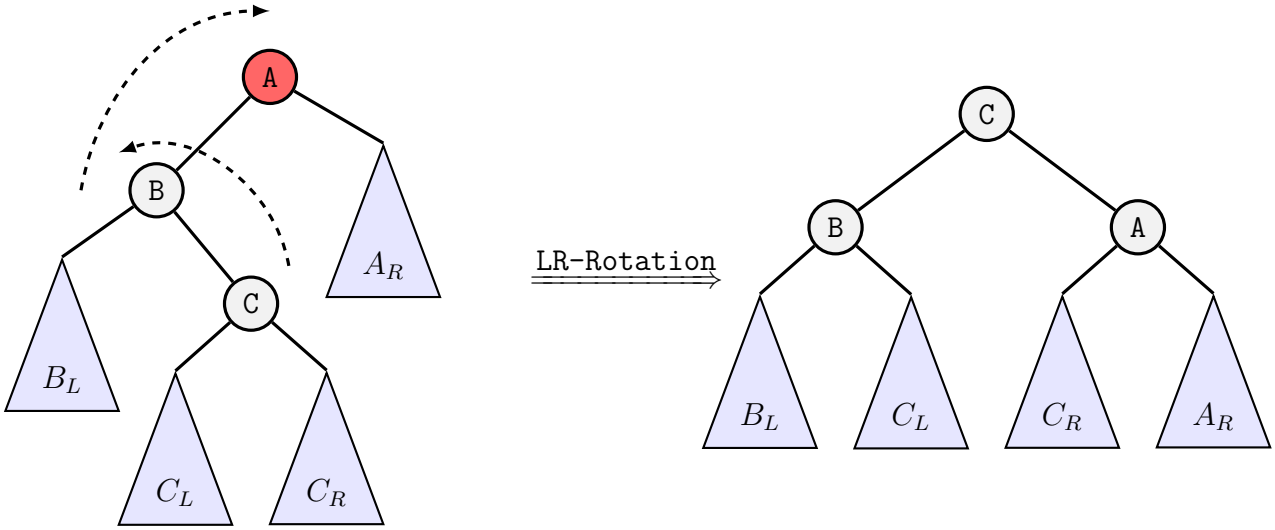
در چرخش RR-Rotation عناصر زیر درخت B_L که مقدار بیشتری از عنصر A دارند در سمت راست این نود قرار می گیرند .



۳.۵ چرخش RL-Rotation با وجود زیر درخت ها

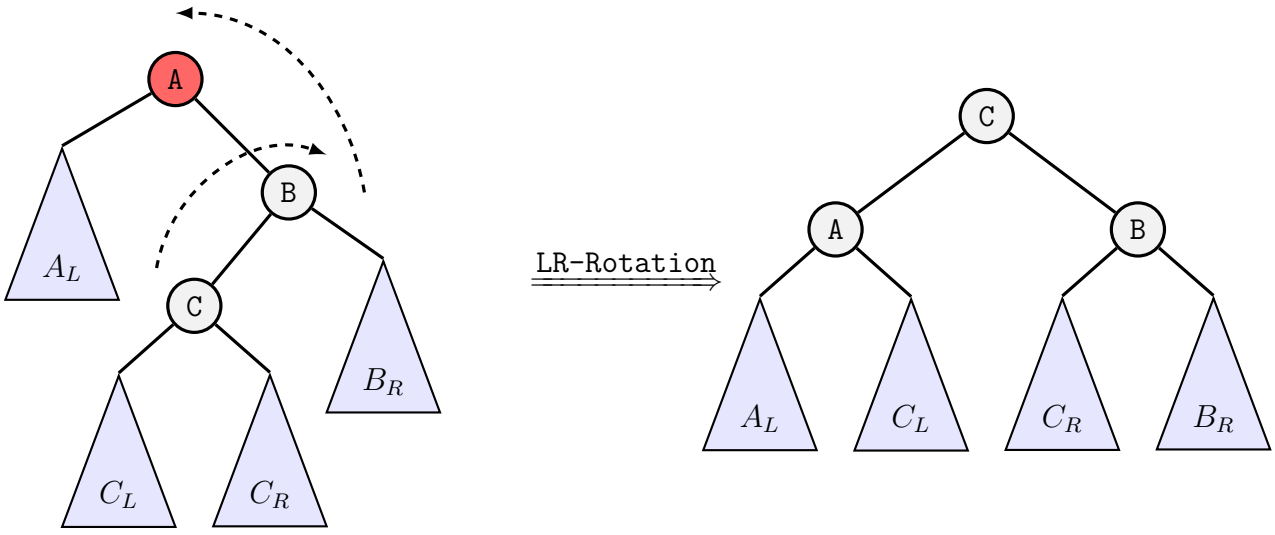
در چرخش RL-Rotation عناصر زیر درخت C_L که مقدار بیشتری از عنصر B دارند در سمت راست این نود قرار می گیرند و عناصر زیر درخت C_R که مقدار کمتری از A دارند در سمت چپ این نود قرار

می گیرند .



۴.۵ چرخش LR-Rotation با وجود زیر درخت ها

در چرخش LR-Rotation عناصر زیر درخت C_L که مقدار بیشتری از عنصر A دارند در سمت راست این نود قرار می گیرند و عناصر زیر درخت C_R که مقدار کمتری از B دارند در سمت چپ این نود قرار می گیرند .

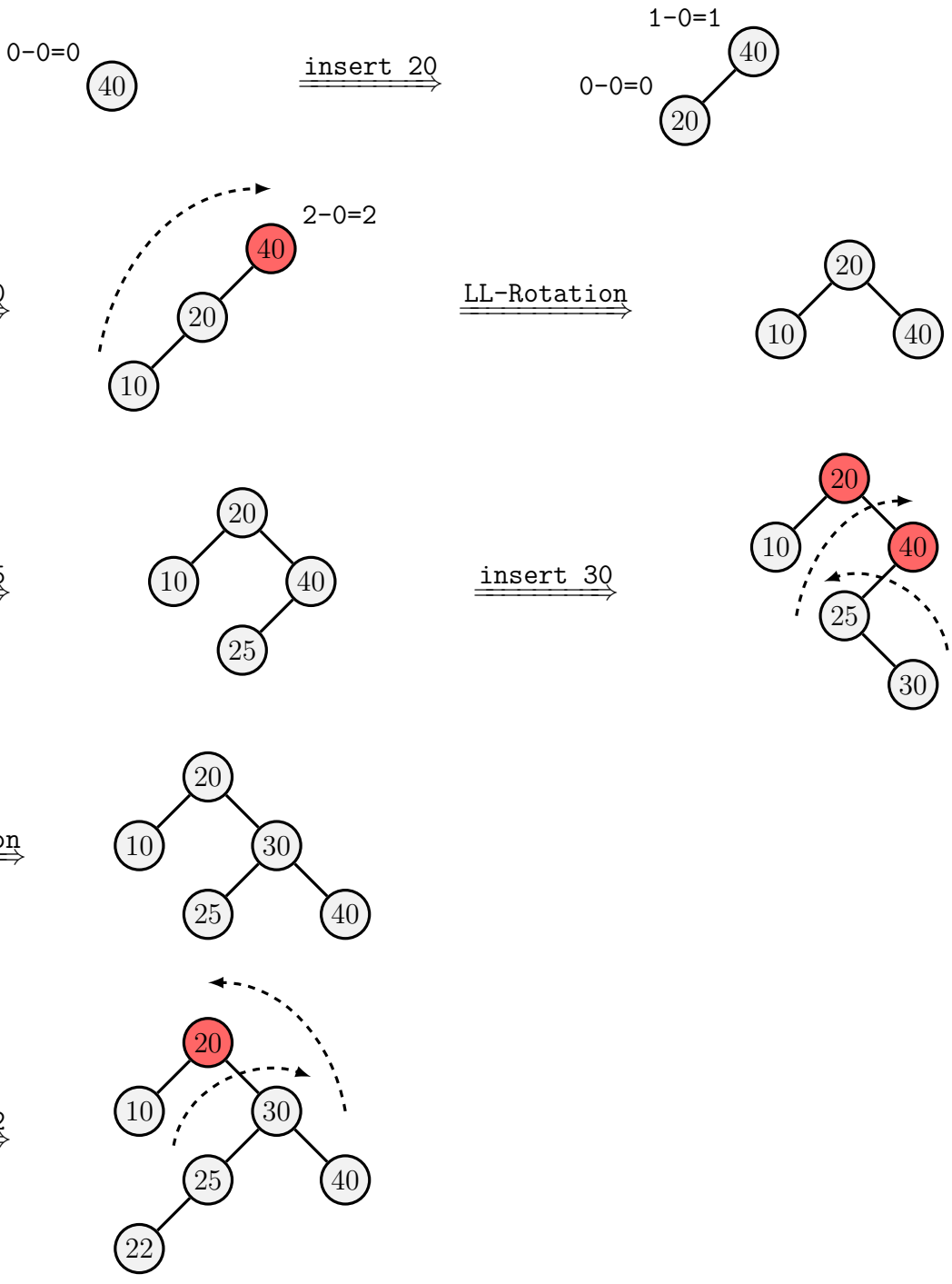


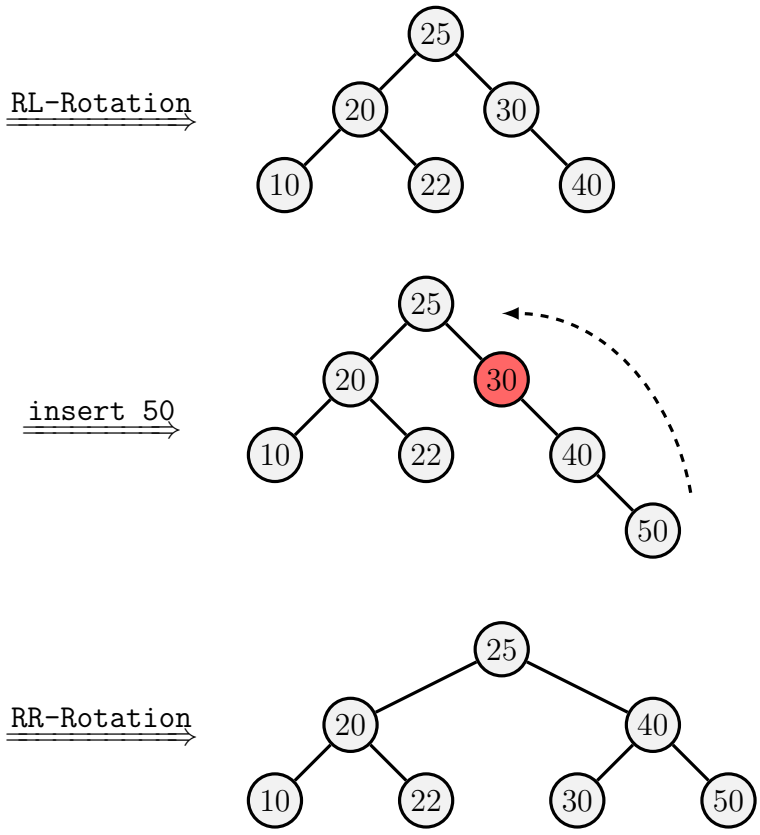
۶ نمونه ی ایجاد درخت AVL

در صورتی که بخواهیم با ورودی های

$$\text{keys} : 40, 20, 10, 25, 30, 22, 50$$

یک درخت AVL ایجاد کنیم ، نحوه ی ایجاد درخت به صورت زیر خواهد بود .

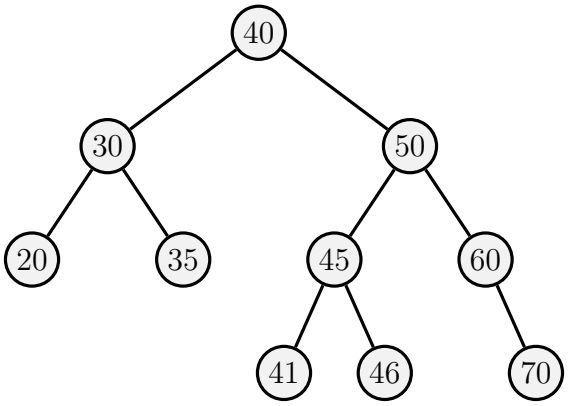




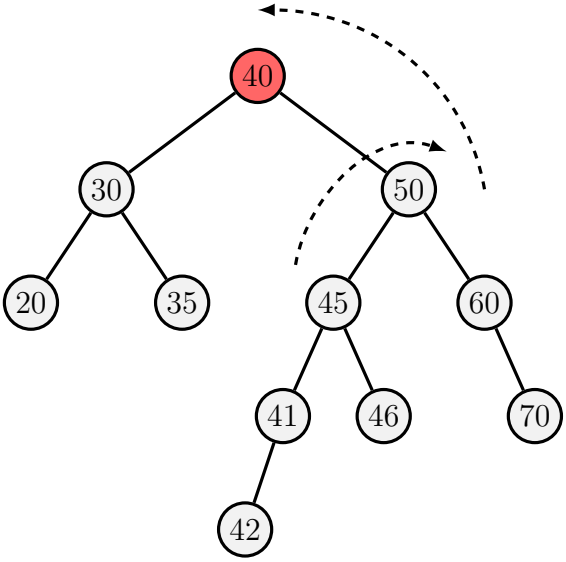
۷ نکته مهم

به هیچ نودی اجازه ندهید که balance factor آن از -2 کمتر و یا از $+2$ بیشتر شود و به محض مشاهده ی عدم توازن ، چرخش های مورد نیاز را روی درخت اعمال کنید .

۸ مثالی دیگر



insert 42



RL-Rotation