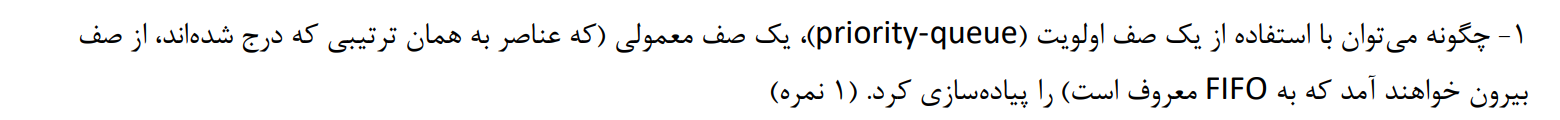
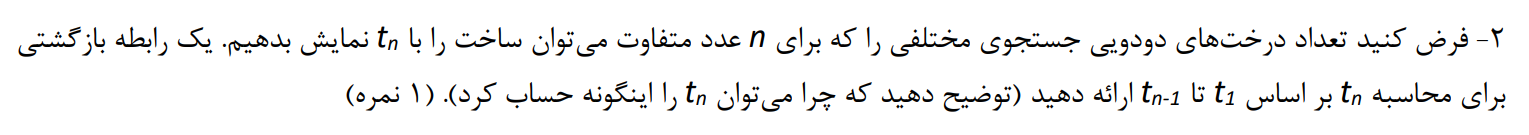
به نام خدا

تمرین شماره ۳ درس ساختمان داده‌ها و الگوریتم‌ها

چمران معینی : ۹۹۳۱۰۵۳





اگر فقط یک عنصر داشته باشیم، مشخص است که فقط یک راه برای تبدیل آن به درخت دودویی جست و جو داریم،‌ یعنی:

اگر دو عنصر داشته باشیم، دو انتخاب برای ریشه داریم. بعد از انتخابِ ریشه هم t1 حالت برای چینش بقیه‌ی عناصر، یعنی:

در حالتی هم که سه عنصر داشته باشیم، سه انتخاب برای ریشه داریم. اگر بزرگ‌ترین یا کوچک‌ترین عنصر ریشه‌مان باشد هر دو عنصرِ دیگر در یک طرف قرار می‌گیرند، یعنی t2 حالت برای چینشِ بقیه‌ی اعضا داریم. اگر هم عوض میانی را انتخاب کرده باشیم t1 \* t1 حالت برای چینشِ بقیه‌ی اعضا داریم، پس:

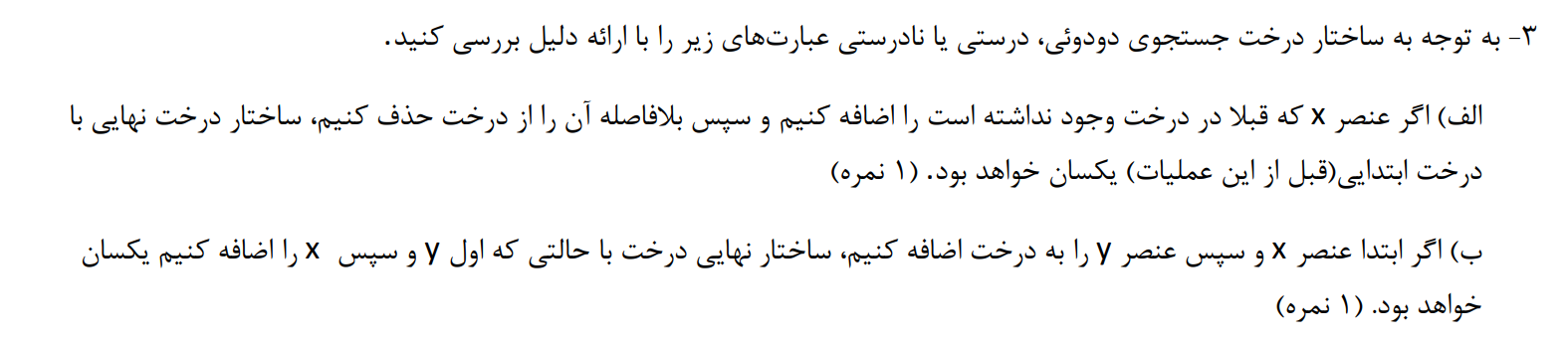
در حالتی که چهار عنصر داریم، چهار انتخاب برای ریشه داریم. در حالتی که بزرگ‌ترین یا کوچک‌ترین عضو را انتخاب کرده باشیم، هر سه عضو در یک طرف قرار می‌گیرند و t3 حالت برای چینش‌شان خواهیم داشت. در حالتی که عضو دوم یا سوم را انتخاب کرده باشیم هم t1 \* t2 حالت برای چینشِ بقیه‌ی اعضا داریم، پس:

وقتی پنج عنصر داشته باشیم هم، پنج انتخاب برای ریشه داریم. اگر بزرگ‌ترین یا کوچک‌ترین عضو باشد t4 حالت برای چینشِ بقیه‌ی عناصر داریم. اگر عضو دوم یا چهارم باشد t1 \* t3 حالت برای چینشِ بقیه‌ی اعضا داریم. اگر عضو سوم ریشه بشود هم t2 \* t2 حالت برای چینشِ بقیه‌ی اعضا داریم، پس:

به همین ترتیب و با همین منطق خواهیم داشت:

پس tn را به این شکل می‌نویسیم:

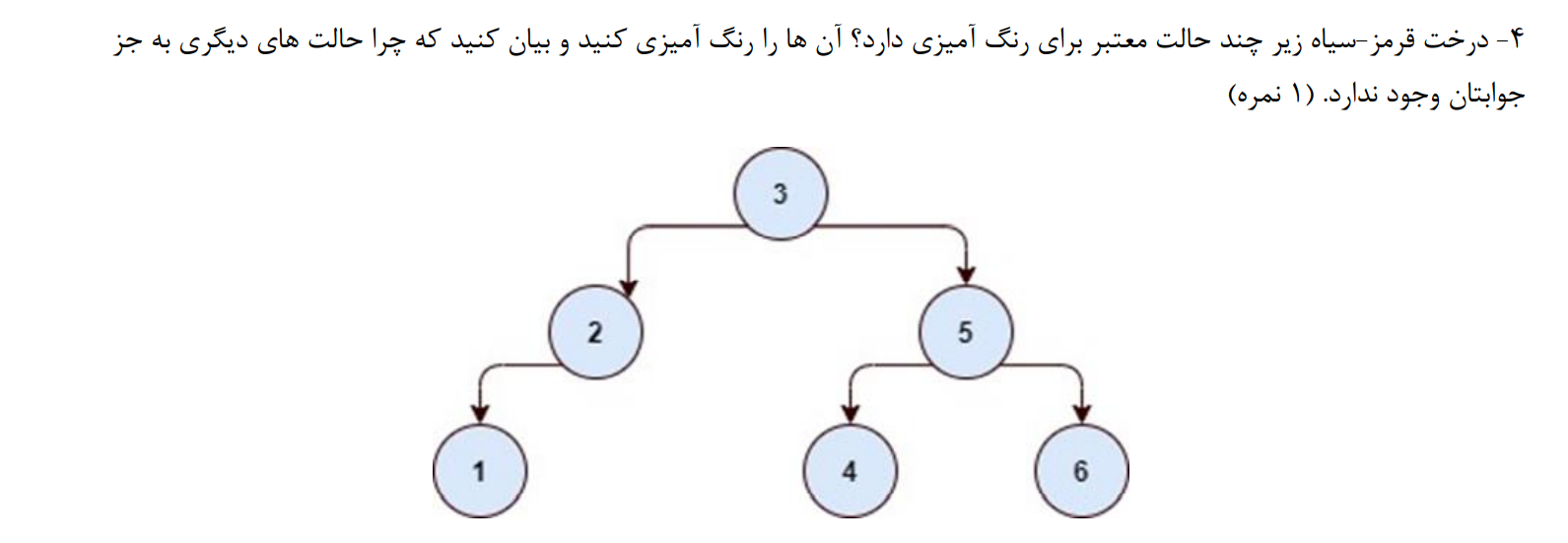
می‌توان این رابطه را به این شکل توضیح داد که از اولین جمله، مربوط به حالتی‌ست که بزرگ‌ترین عنصر را به عنوان ریشه انتخاب کرده باشیم، در این حالت تمام n-1 در یک سمت قرار می‌گیرند و می‌توان آن‌ها را به t(n-1) حالت چید. دومین جمله، مربوط به زمانی‌ست که دومین عنصر بزرگ را به عنوان ریشه انتخاب کنیم، در این حالت یک عنصر در یک سمت قرار می‌گیرد که t1 چینش دارد، بقیه‌ی اعضا هم در سمت دیگر قرار می‌گیرند که t(n-2) چینش خواهند داشت، الی آخر.



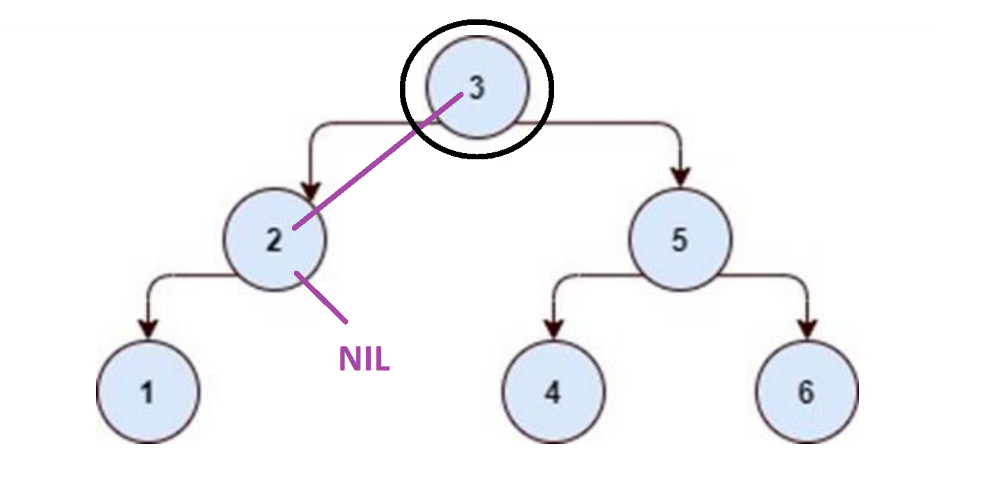
الف) می‌دانیم که عملیاتِ insert در یک ددج، تغییری در جایگاهِ قبلیِ عناصر ایجاد نمی‌کند. تنها کاری که می‌کنیم این است که می‌گردیم و خانه‌ی نال مناسب را پیدا می‌کنیم و عنصرِ جدید را به جای آن می‌گذریم.

اما می‌دانیم که عملیاتِ حذف می‌تواند با تغییراتی در عناصرِ دیگر هم همراه باشد، اما در چه صورت؟ در صورتی که عنصرمان فرزندانی داشته باشد، اما می‌دانیم که عنصری که تازه اضافه شده، به عنوان یک برگ اضافه می‌شود و فرزندی ندارد و با توجه به این که بلافاصله بعد از اضافه کردن اقدام به حذف شده، در نتیجه فقط یک لحظه عنصری به عنوان فرزندِ یکی از بزرگ‌ها اضافه می‌شود و سپس دوباره حذف می‌شود و درخت دقیقا مانند قبل خواهد بود، پس این گزاره صحیح است.

ب) اشتباه است. برای نقض آن، یک مثال نقض کافی‌ست. فرض کنید ددج‌ای داشته باشیم با ریشه‌ی ۱۰، اگر اول ۹ را اضافه کنیم و بعد ۸ را، ۹ فرزندِ ریشه می‌شود و ۸ هم فرزندِ ۹ می‌شود، اما در صورتی که اول ۸ را اضافه کرده باشیم، ۸ فرزندِ ریشه می‌شود و ۹ در سمت راستِ ۸ قرار می‌گیرد.



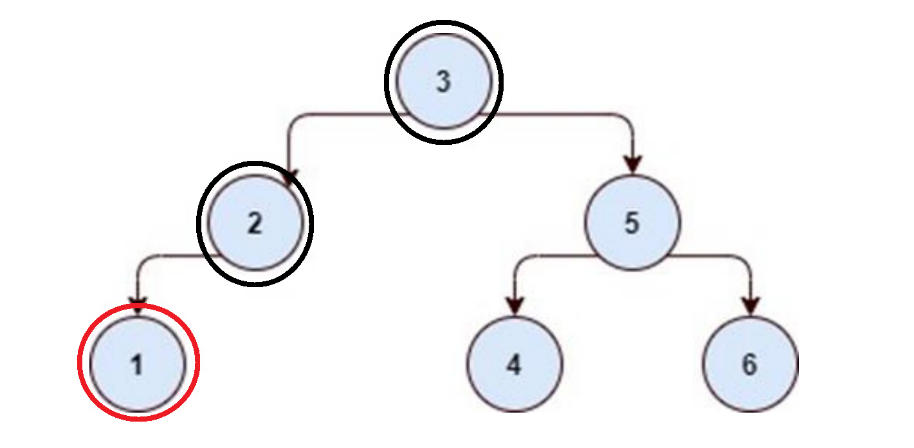
می‌دانیم که ریشه باید سیاه باشد، پس ابتدا آن را رنگ می‌کنیم و بعد سراغ بقیه می‌رویم.



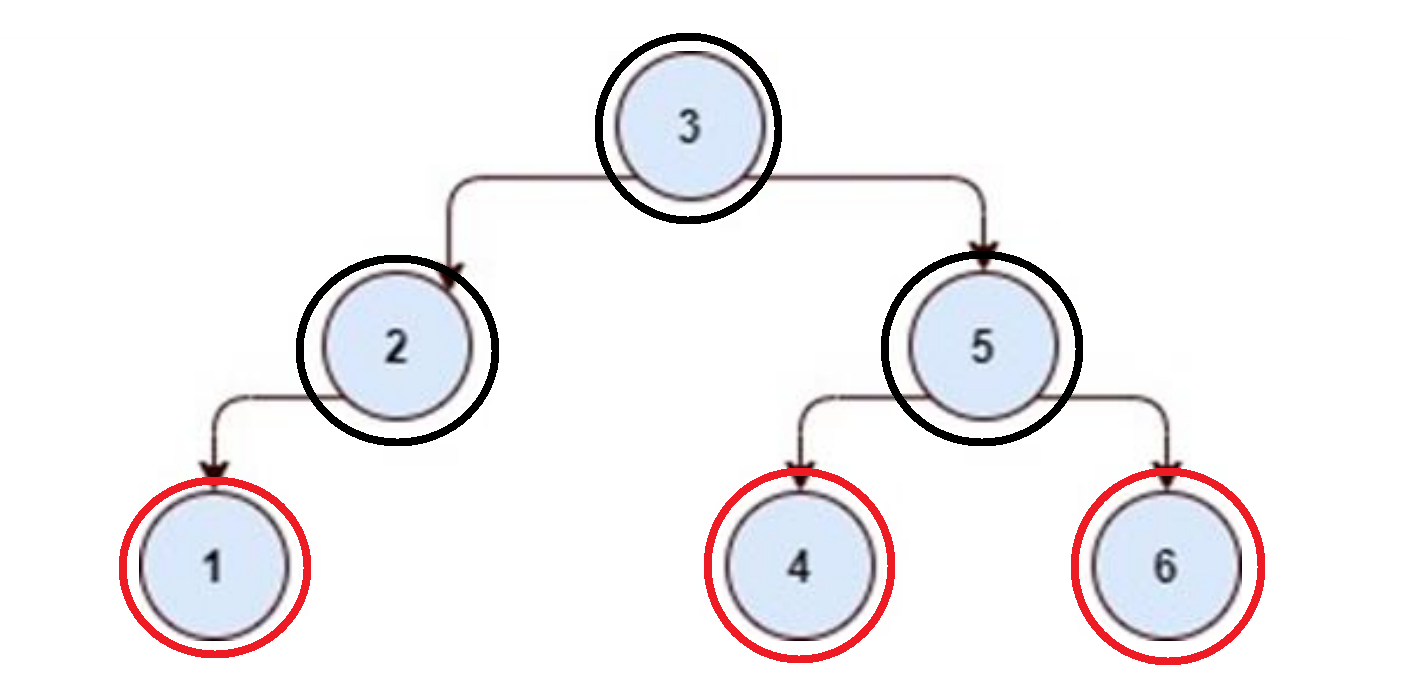
نود ۲ را در دو حالتی که قرمز و سیاه باشد بررسی می‌کنیم.

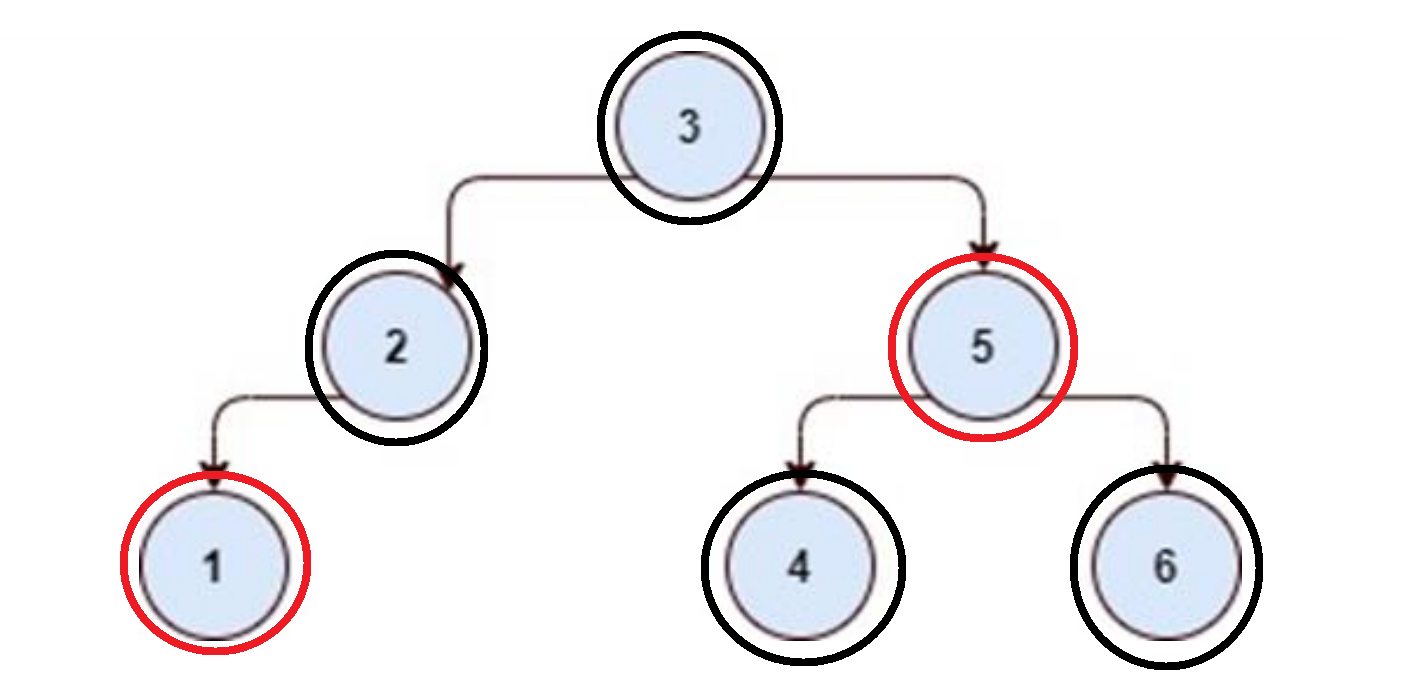
اگر قرمز باشد، مسیر مشخص شده (که یک مسیر root-NIL است)، دارای یک نود قرمز خواهد شد، پس بقیه‌ی مسیرهای ریشه تا نال هم باید تنها یک نود مشکی داشته باشند که ممکن نیست، برای مثال در همین مسیر ۳-۲-۱ ، مجبور می‌شویم نود ۱ را هم قرمز کنیم که آن‌گاه نود ۲ و فرزندش ۱ هر دو قرمز می‌شوند که غیرمجاز است، پس این حالت رد می‌شود.

اگر ۲ سیاه باشد، مسیرِ مشخص شده دارای دو نود سیاه خواهد بود و همه‌ی نودهای ریشه تا نال باید دو نود سیاه داشته باشند. پس نود ۱ را هم قرمز می‌کنیم که مسیر ۳-۲-۱ دارای دو نود سیاه باشد. تا این‌جا درختمان به این شکل در آمد که تنها حالتِ مجاز برای خانه‌های رنگ شده است:

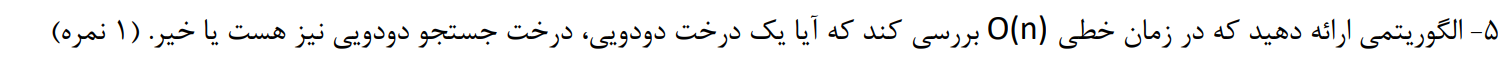


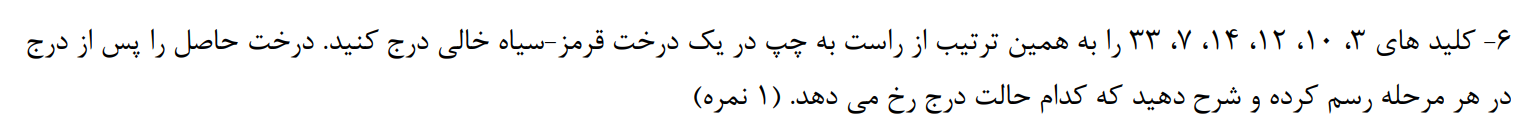
دو مسیر ریشه تا نالِ دیگر داریم ۳-۵-۶ و ۳-۵-۴ که هرکدام یک خانه‌ی مشکی‌اش تثبیت شده، و نیاز به یک خانه‌ی مشکیِ دیگر دارد. می‌توانیم ۵ را قرمز کنیم و بقیه را مشکی، و همچنین می‌توانیم پنج را مشکی کنیم و بقیه را قرمز.



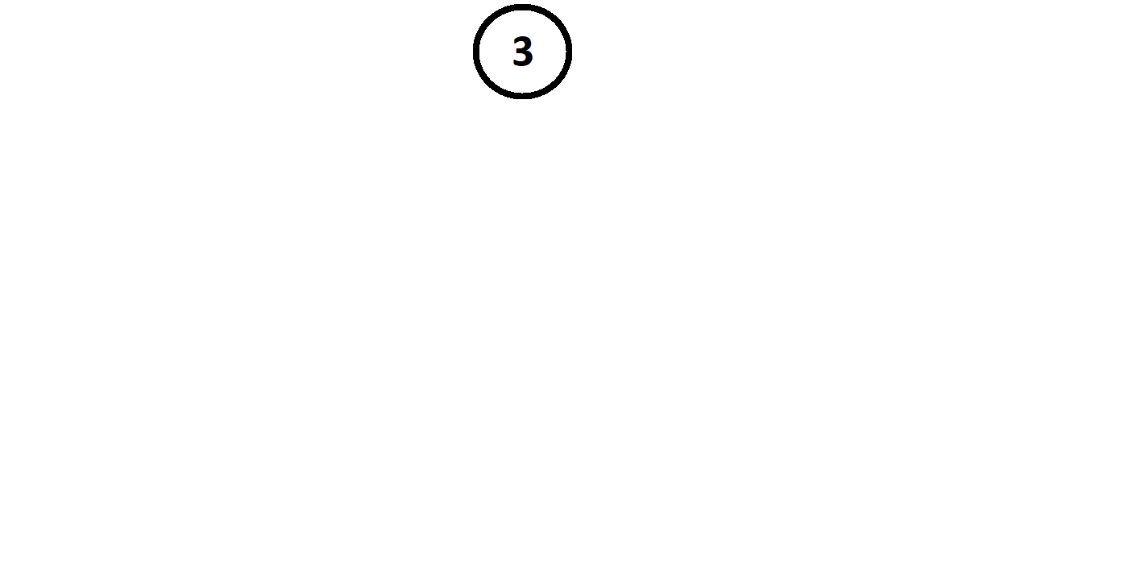


با توجه به این که از ابتدا حالت‌های غیرمجاز را حذف کردیم و حالت‌های مختلف برای انتخاب‌های مجاز را بررسی کردیم، حالتِ‌ مجاز دیگری غیر از این دو وجود ندارد.

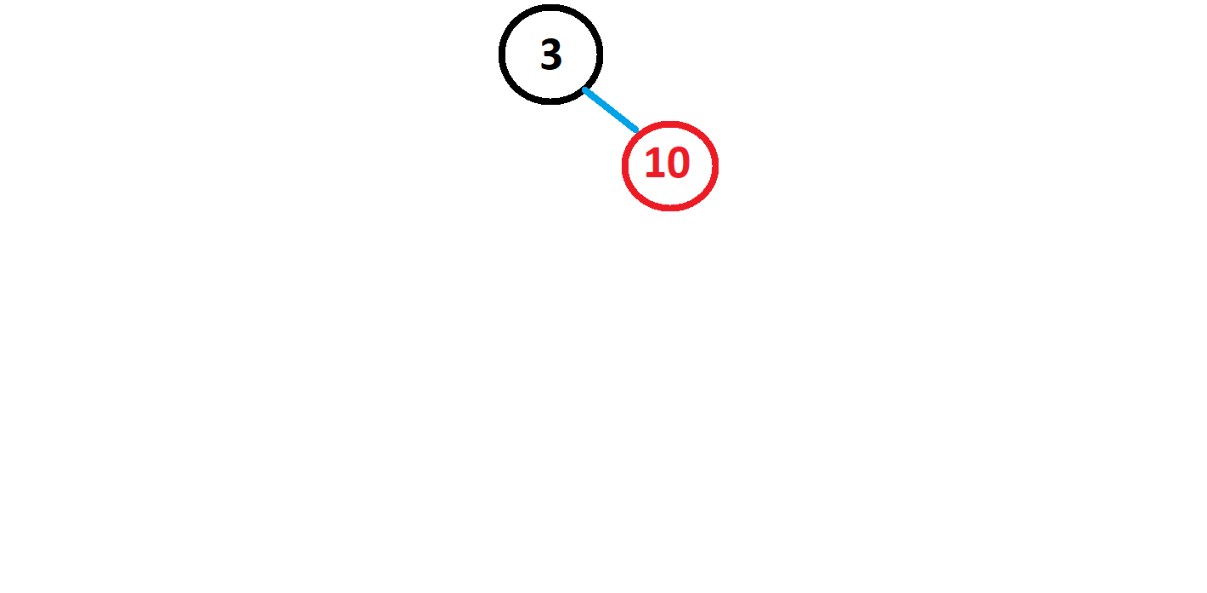




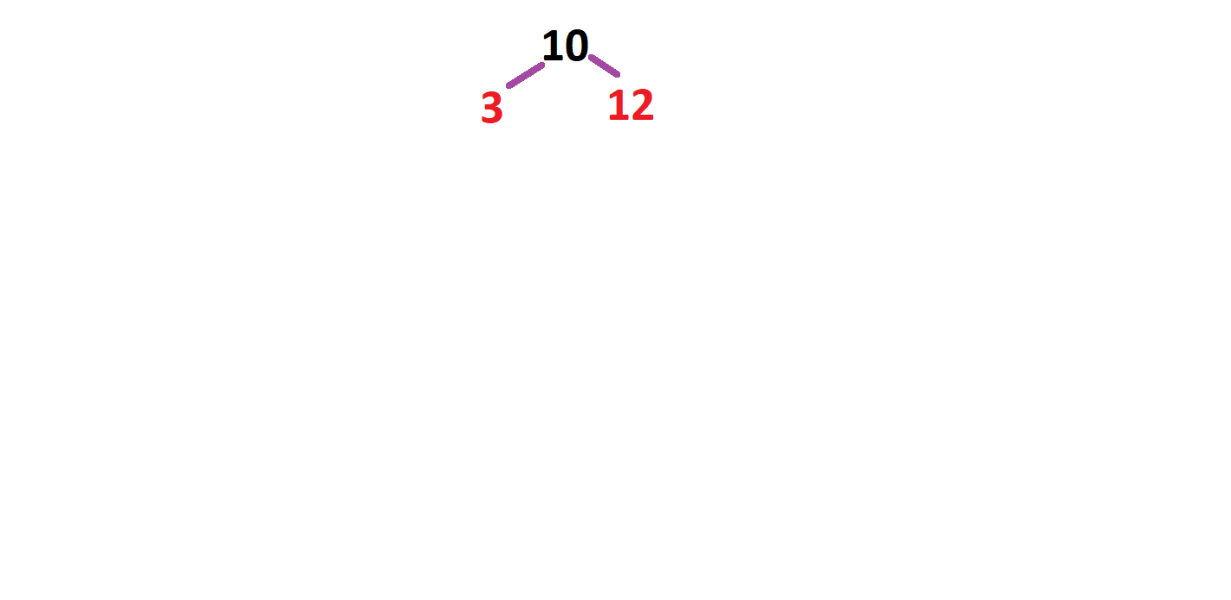
ابتدا ۳ را به عنوان ریشه‌ی سیاه اضافه می‌کنیم:



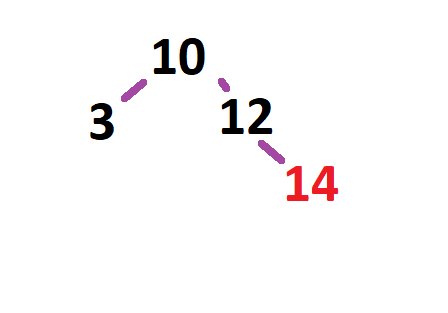
حال ۱۰ را اضافه می‌کنیم که به فرزندیِ راست ۳ می‌رود و چون پدرش سیاه است، به راحتی آن را قرمز می‌کنیم و تمام.



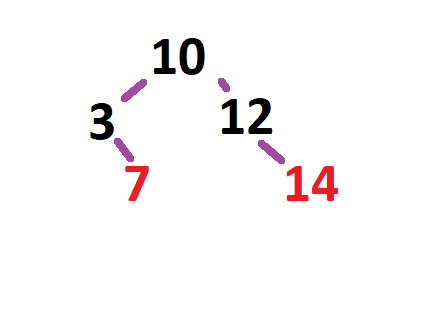
نوبت به ۱۲ می‌رسد که به فرزندیِ راست ۱۰ می‌رود. چون پدرش قرمز است اوضاع کمی پیچیده می‌شود. به سراغ عموی ۱۰ می‌رویم، یعنی فرزندِ دیگرِ‌ ۳ که برابر با نال است، پس مشابه کیس۲ عمل می‌کنیم و پدر یعنی ۱۰ را rotate می‌کنیم تا به جای پدربزرگ برود و درخت‌مان به این شکل می‌شود:



حالا چهارده را اضافه می‌کنیم که به فرزندیِ راستِ ۱۲ می‌رود. پدرش یعنی ۱۲ قرمز است، پس عمو را بررسی می‌کنیم که آن هم قرمز است، پس مطابق کیس۱ عمل می‌کنیم و پدر و عمو را مشکی می‌کنیم. از آن‌جا که پدر‌بزرگ که ۱۰ باشد ریشه است، آن را همان مشکی رها می‌کنیم و از آن‌جایی که ریشه روی همه‌ی مسیرها تاثیر یک‌سان دارد، مشکلی نخواهیم داشت.



نوبت به هفت می‌رود که به فرزندیِ راست ۳ می‌رود. چون پدرش سیاه است، آن را قرمز اضافه می‌کنیم.



نوبت به آخرین عنصر یعنی ۳۳ می‌رد که باید به فرزندیِ راست ۱۴ برود. چهارده پدر قرمز است، پس عمو را بررسی می‌کنیم که نیل است، پس مطابق کیس دو عمل می‌کنیم و چهارده و دوازده را دوران می‌دهیم و فرزند جدید را مشکی اضافه می‌کنیم. با توجه به بقیه‌ی اعضا، می‌بینیم که نیازی به رنگ کردنِ دوباره هم نیست و همه‌ی مسیرهای ریشه-نیل هم دقیقا دو نود سیاه دارند.

