ساختمان داده‌ها (۲۲۸۲۲)

مدرس: حسین بومری

[**بهار** 99]

جلسه ۲۰: AVL and red black tree

نگارنده: محمد مهدی زارع

در جلسه قبل در مورد درخت های جست و جو دودویی [۱] و مرتبه زمانی و حافظه ی عملیات های مختلف روی ان ها صحبت کردیم و در این جلسه قصد داریم با درخت قرمزسیاه[۲] وavl که دو نوع از درخت هایی جستجوی دودویی خود متوازن کننده هستند بیشتر اشنا شویم.

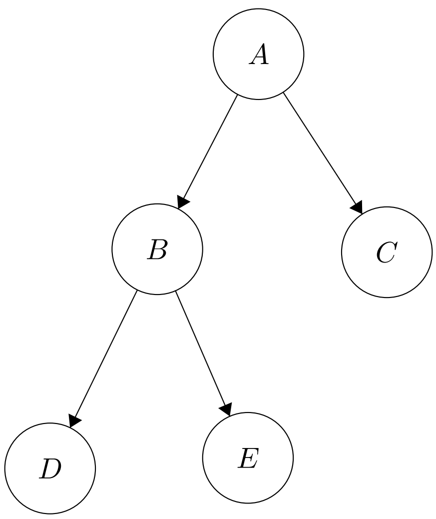
۱ AVL tree

در درخت های متوازن جستوجوی دودویی ممکن است بعد از حذف کردن یا درج کردن یک عنصر توازن درخت به هم بریزد پس لازم است که انرا متوازن[۳] کنیم. برای اینکار از چرخش ها استفاده میکنیم که انواع ان **چرخش چپ چپ** [۴] و**چرخش راست راست** [۵] است.

۱.۱ چرخش چپ چپ

در این حالت توازن در سمت چپ رخ داده است .

برای مثال در شاخه D رخ داده است و میخواهیم عمق ان یک واحد به بالاتر برود .



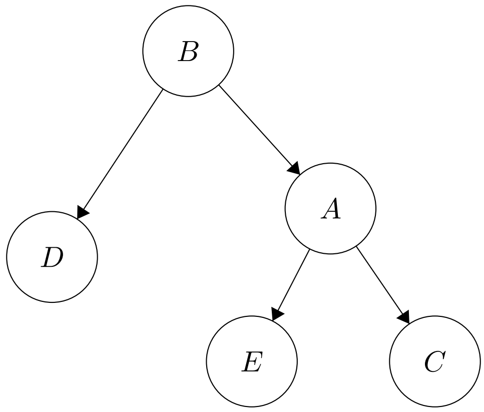
با تصویر کردن درخت به ترتیب زیر میرسیم

---

[1] Binary Search Tree
[2] red black tree
[3] balance
[4] left-rotate
[5] right-rotate

۱. $DBEAC$

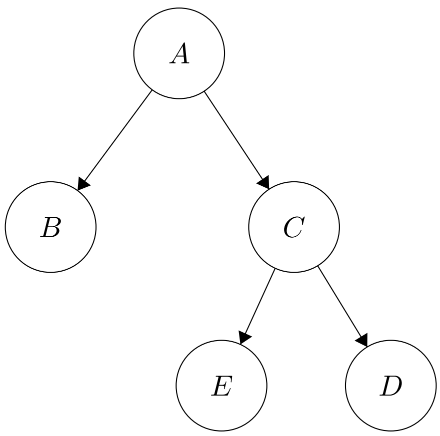
پس B را به عنوان ریشه انتخاب میکنیم . داریم :



توجه کنید که بعد از چرخش باید توجه کنیم توازن در شاخه های بالایی به هم ریخته نشود و اگر نامتوازن باشد این چرخش ها بازگشتی به سمت بالا حرکت می کنن

۲.۱ چرخش راست راست

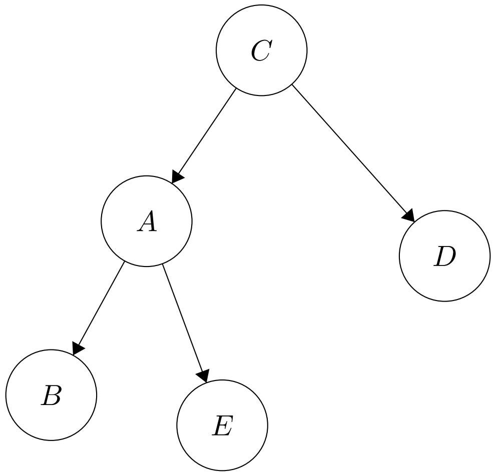
در این حالت توازن در سمت راست یک عنصر رخ داده است . برای مثال فرض کنید توازن در D به هم ریخته است و بخواهیم عمق ان یک واحد بالاتر برود



با تصویر کردن درخت به ترتیب زیر میرسیم

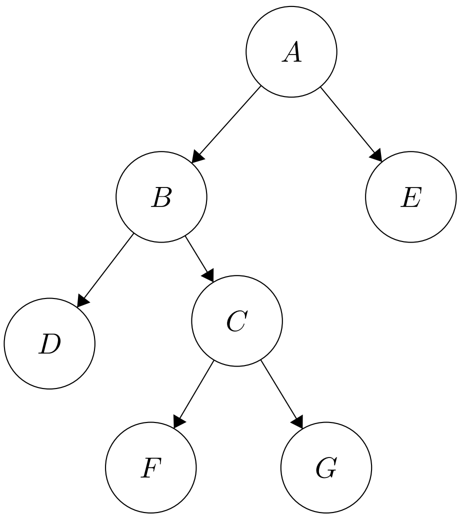
۱. $BAECD$

پس C را به عنوان ریشه انتخاب میکنیم . داریم :



۳.۱ چرخش چپ راست

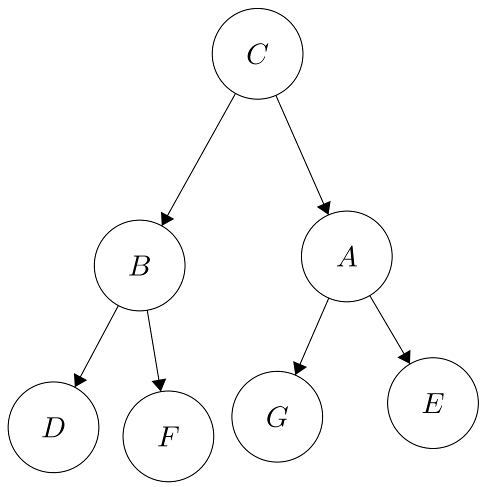
توازن در C به هم ریخته است



درخت را تصویر میکنیم

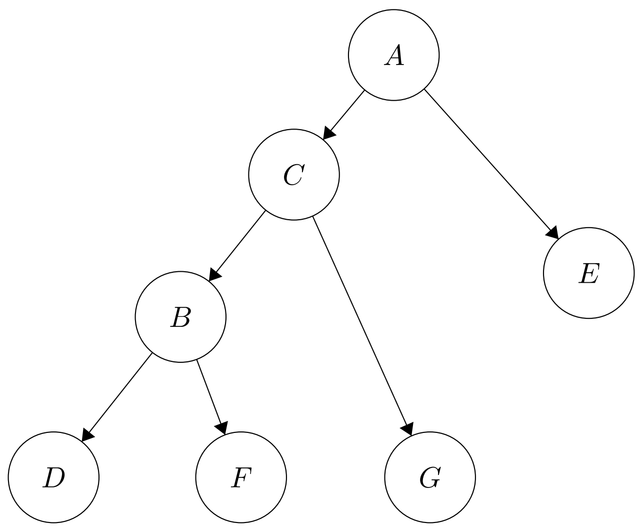
۱ $DBFCGAE$

برای درخت ریشه C را در نظر میگیریم



البته در کتاب راه حل و الگوریتم دیگری هست که بعدا گفته میشود . در این روش عناصر درخت را به ترتیب مینویسیم و درخت را از اول مینویسیم. در روش دیگر میتوانیم جایB و C را عوض کنیم . و با این کار از حالت چپ راست به چپ چپ رسیدیم. برای مثال همین درخت این بخش را در دوگام زیر ببینید.

۱.جابه جایی B و C



حالا با چرخش چپ چپ بالانس میشود.

۴.۱ چرخش راست چپ

مشابه بالا ابتدا از راست چپ به راست راست تبدیل میکنیم و سپس با چرخش راست راست متوازن میشود .

توجه کنید مرتبه زمانی همه این چرخش ها از $O(logn)$ است .چون در هر چرخش مقدار ثابتی کار انجام میدهیم و این کار ممکن است تا ریشه ادامه پیدا کند.

۲ درخت قرمز سیاه

نوعی درخت جست و جوی دودویی متوازن است که گره های ان رنگ هم دارند.

۱.۲ ویژگی های درخت قرمز سیاه

۱.هر گره یا قرمز است یا سیاه

۲.ریشه سیاه است

۳.تمام برگ ها نال و سیاه هستند

۴اگر یک گره قرمز باشد انگاه دو فرزند ان سیاه هستند

۵. برای هر گره ،تمام مسیر ها از گره تا برگ ها حاوی تعداد مساوی گره سیاه هست (ارتفاع سیاه)

۱.۱.۲ ارتفاع سیاه BH

چون دو قرمز نمی توانند پشت هم باشند پس ارتفاع درخت (H):

$$H \leq 2BH$$

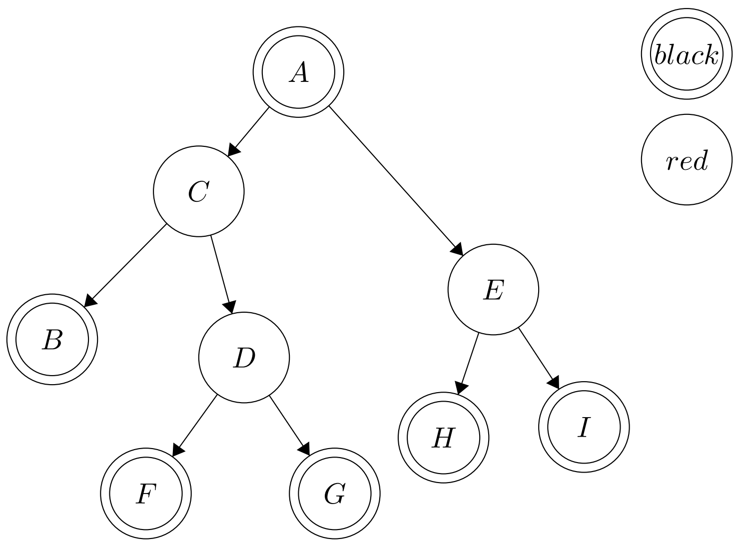
حال میخواهیم ماکزیمم عمق یک درخت قرمز سیاه را پیدا کنیم . فرض کنید برگ های قرمز را در نظر نگیریم پس به یک درخت کامل میرسیم (با نوجه به ویژگی های درخت قرمز سیاه) پس ارتفاع سیاه این درخت از $O(logn)$ است پس :

$$H \leq 2(O(logn)) \rightarrow H = O(logn)$$

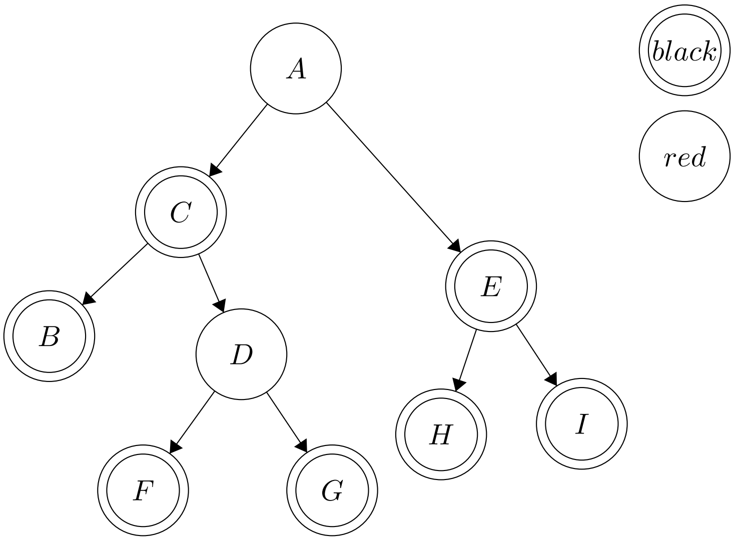
۲.۲ درج کردن عنصر

عنصری که درج میشود را معمولا قرمز در نظر میگیریم

فرض کنید حالت زیر به دلیلی به وجود اماده باشد و دو گره قرمز پشت هم باشند



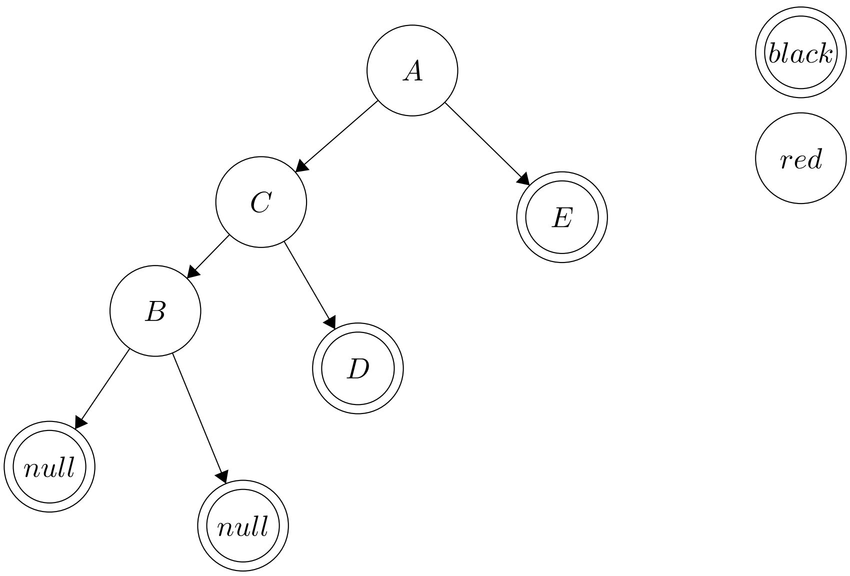
حال با تغییر رنگ های گره ها میتوتنیم دوباره شرایط درخت قرمز سیاه را فراهم کنیم



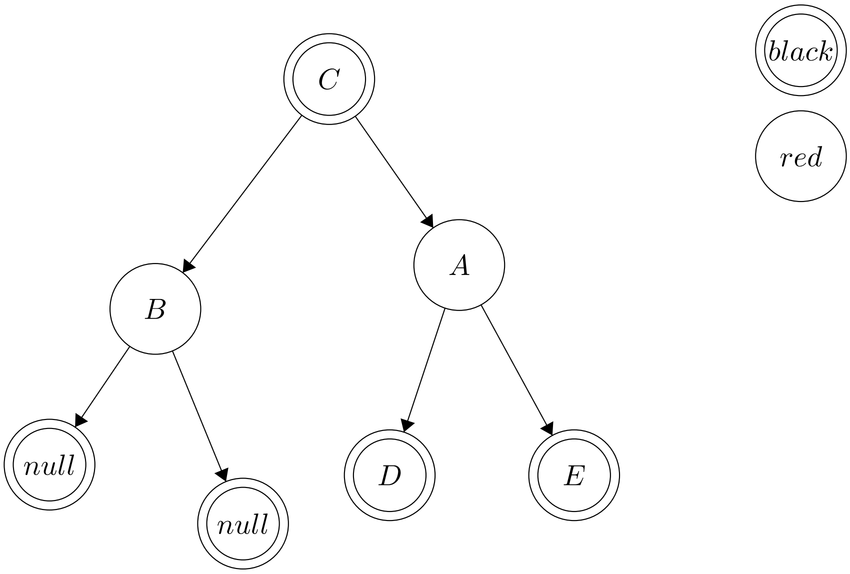
البته توجه کنید اگر A ریشه بود رنگ ان مشکی می ماند و انرا به قرمز تغییر نمیدهیم و باز شرایط درخت قرمز سیاه برقرار است.

همچنین در حالتی که C, E سیاه باشند باز مشکلی پیش نمی اید چون ارتفاع سیاه تغییر نکرده .

حالا حالتی در نظر بگیرید که C قرمز و E سیاه باشد .اگر گره درج شده فرزند گره E باشد که همچنان مشکلی پیش نمی اید ولی اگر فرزند C باشد . درخت را تصویر میکنیم و باز سازی میکنیم و سپس درخت را طوری رنگ امیزی میکنیم که شرایط برقرار باشد .دقت کنید که شرایط ارتفاع سیاه تغییر نکرده و فقط دو گره قرمز به دنبال هم امده اند.



که پس از بازسازی و رنگ کردن به حالت زیر میرسیم



بین این دو درخت AVL بالانس تر هست چون در درخت قرمز سیاه ممکنه عمق یک طرف دو برابر طرف دیگ باشه.

۳ پیماییش ها

۱.۳ میان وندی [۶]

در این نوع پیمایش اول درخت سمت چپ گره پیمایش میشود و سپس خود گره و بعد از ان درخت سمت راست پیمایش میشود. در این نوع پیمایش ترتیب صعودی عناصر درخت بدست می اید و مرتبه ان از $O(n)$است ولی ساختار درخت را حفظ نمیکند

۲.۳ پسوندی[۷]

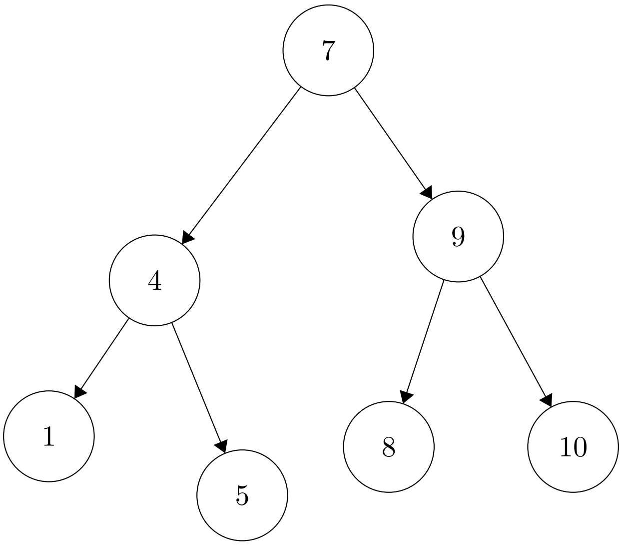
در این نوع پیمایش اول سمت چپ درخت را میبینم سپس سمت راست ان و در اخر خود گره و مرتبه ان از $O(n)$است و ساختار درخت را حفظ میکند

۳.۳ پیشوندی[۸]

در این نوع پیمایش اول خود گره را میبینیم سپس درخت سمت چپ و بعد از ان درخت سمت راست و مرتبه ان از $O(n)$است و ساختار درخت را حفظ میکند

پیمایش درخت زیر را در نظر بگیرید

[6] infix
[7] postfix
[8] prefix



$$infix = 1, 4, 5, 7, 8, 9, 10$$

$$prefix = 7, 4, 1, 5, 9, 8, 10$$

$$postfix = 1, 5, 4, 8, 10, 9, 7$$

مراجع

[1] Cormen, Thomas H., et al. *Introduction to Algorithms*. 3rd ed., MIT Press, 2009, pp. 18-22.