1. Balance Factor را برای یک گره از یک درخت دودویی به صورت قدرمطلق اختلاف ارتفاع زیر درخت سمت چپ و زیر درخت سمت راست راست آن گره تعریف میکنند.
در یک درخت AVL مقدار Balance Factor برای تمامی گره ها حداکثر برابر یک است.
فرض کنید Fh حداقل تعداد گره های مورد نیاز برای ساخت درخت AVL به ارتفاع h باشد. نشان دهید که Fh از رابطه ی فیبوناچی پیروی میکند و سپس نتیجه بگیرید که h_{AVL} حداکثر برابر با لگاریتم n در پایه ی z است که n تعداد رئوس درخت و z نسبت طلایی دنباله ی فیبوناچی است.

پاسخ :
با تعدادی گره ی ثابت برای آنکه نامتوازن ترین درخت ممکن را ایجا کنیم از روش بازگشتی کمک میگیریم. فرض کنید F_h تعداد راس های درخت مطلوب ما باشد در این صورت هر یک از دو زیر درخت چپ و راست این درخت هم دقیقا دارای همین خاصیت میباشند. بعلاوه میدانیم اختلاف ارتفاع زیر درخت چپ و زیر درخت راست دقیقا یک است پس تعداد رئوس زیر درخت چپ برابر F_{h-1} و تعداد رئوس زیر درخ راست F_{h-2} است و چون یک راس هم به عنوان ریشه قرار داده شده است بنابرین داریم

$F_h=F_{h-1}+F_{h-2}+1$

حال با استفاده از تغییر متغیر $G_h=F_h-1$ خواهیم داشت

$G_h=G_{h-1}+G_{h-2}$

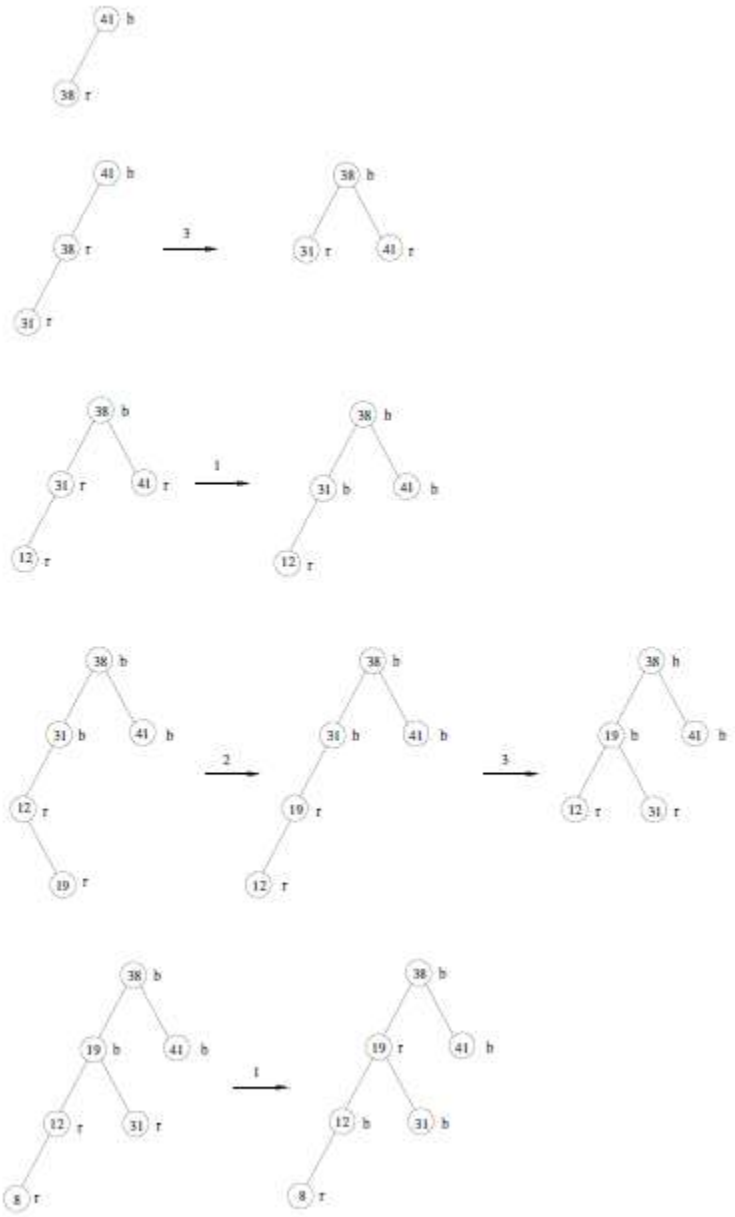
حال باتوجه به آن که G_h , h امین عدد از دنباله ی فیبوناچی است پس رشدی نمایی بر پایه ی نسبت طلایی دارد پس با لگاریتم گرفتن خواهیم داشت $h_{AVL} <= \log_{1.68} n$

2. نشان دهید که بلندترین مسیر ساده ی ممکن از یک گره ی x در یک Red Black Tree تا یک برگ نوه ی x حداکثر برابر است با دوبرابر کوتاهترین مسیر ساده از گره ی x تا یک برگ نوه ی x.

پاسخ:
با توجه به این ویژگی از درخت های سیاه قرمز که برای هر گره تمام مسیر های از آن گره تا برگ های نوه ی آن گره دارای تعداد مساوی گره سیاه است بنابرین بلند ترین و کوتاهترین مسیر باید تعداد برابری راس سیاه داشته باشند حال با توجه به آنکه اگر یک گره قرمز باشد هر دو فرزندش سیاه است بنابرین هر راس دیگری در طولانی ترین مسیر باید سیاه باشد و درنتیجه طولش حداکثر دوبرابر کوتاهترین مسیر است.

3. در یک درخت خالی به ترتیب اعداد 8,19,31,12,38,41 را درج کنید و یک Red Black Tree بسازید

پاسخ:



4. T1 و T2 دو درخت Red Black هستند که به ترتیب n و m عنصر دارند. با شرایط زیر الگوریتم‌هایی ارائه دهید که مشخص کند آیا عناصر T1 زیرمجموعه‌ی عناصر T2 هستند یا خیر.
   a. زمان O(nlogm) باشد و حافظه از O(1).
   b. زمان O(n+m) باشد و حافظه O(n+m)
   c. زمان O(n+m) باشد و حافظه(O(logn + logm

پاسخ:

a. برای هر کدام از n عنصر T1 با هزینه‌ی logm بررسی می‌کنیم که در T2 باشد.

b. پیمایش inorder هر درخت را با هزینه‌ی n + m در دو آرایه‌ی N و M قرار می‌دهیم. این دو آرایه مرتب اند و و اگر برای بررسی اینکه N[i] در M باشد k عنصر M را دیده باشیم نیاز به بررسی مجدد آنها نیست و برای بررسی وجود N[i+1] کافی است عناصر بعد از k در M بررسی شوند. با این روش هر عنصر N و M یکبار دیده می‌شوند.

c. مانند راه قسمت قبل عمل می‌کنیم اما نیاز نیست که عناصر را در آرایه ذخیره کنیم، درحین پیمایش درخت‌ها همان کار را می‌کنیم و درواقع حرکت مستقیم روی آرایه‌ها را با پیمایش درخت جایگزین می‌کنیم. (در پیمایش درخت از کوچکترین عنصر شروع می‌کنیم و هربار عنصر کوچکتر بعدی را داریم. حافظه O(1) است که O(logm + logn) میباشد.

5. حداقل و حداکثر تعداد nodeهای داخلی یک درخت red black در حالتی که black height برابر k باشد را بدست آورید.

پاسخ:

حداقل این مقدار با سیاه بودنnode ها برابر $2^K - 1$ است. اگر راس قرمزی اضافه شود k را تغییر نمی‌دهد و تعداد راس‌ها بیشتر می‌شود و برای حد پایین باید راس‌های قرمز کمترین مقدار باشند.

برای حد بالا باید بیشترین نود قرمز در درخت باشد، و چون دو راس قرمز نمی‌توانند متوالی باشند راس‌های طبقه‌های زوج (مانند ریشه که ارتفاعش صفر است) را سیاه و طبقه‌های فرد را قرمز درنظر می‌گیریم که تعداد نودها برابر $2^{2K} - 1$می‌شود.

کوشا باشد و امیدوار