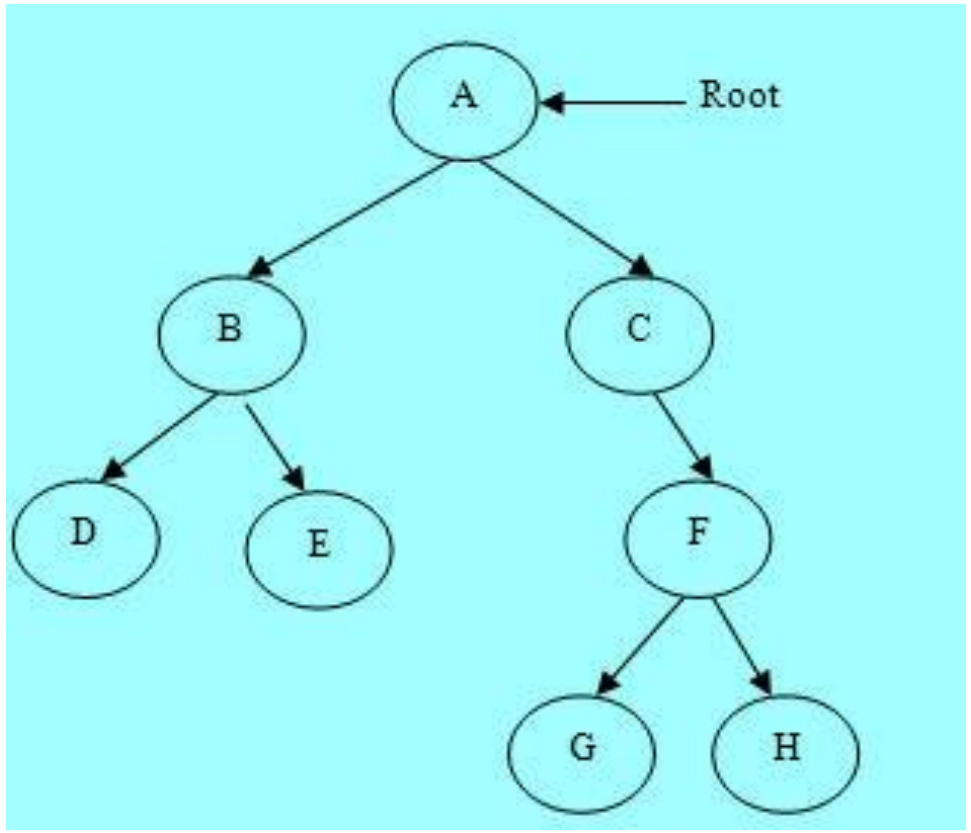
Data Structure

DR. RASTGOO

فصل ۷:

درخت های جستجو

درخت جستجوی دودویی (BST)



درختی را جستجوی دودویی(Binary Search Tree) می نامند که :

۱- عناصر زیر درخت چپ، کوچکتر از ریشه باشند.

۲- عناصر زیر درخت راست، بزرگ‌تر از ریشه باشند.

۳- زیر درختان چپ و راست ، درختان جستجوی دودویی باشند.

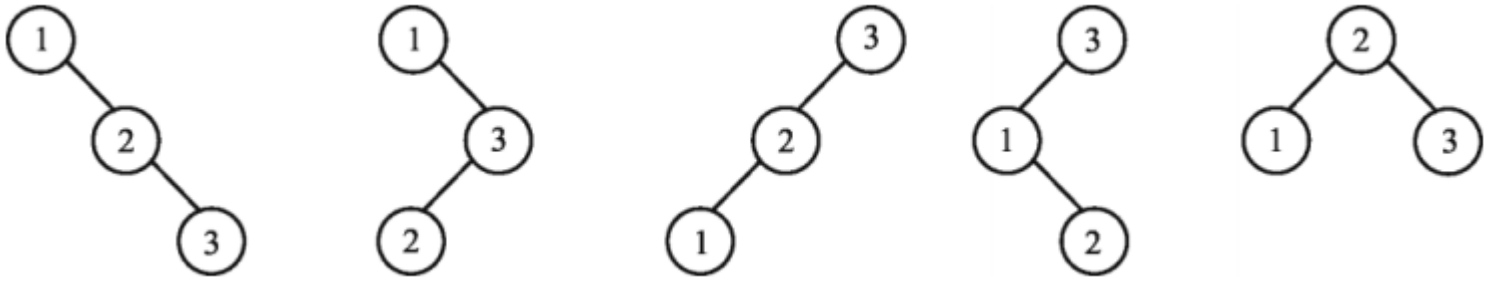
۴- گره با عنصر تکراری نداشته باشد.

مثال

به چند حالت می توان با وارد کردن مقادیر 3,2,1 به هر ترتیب دلخواه در یک BST تهی، یک BST با سه گره ساخت؟

پاسخ:

ترتیب های ممکن برای ورود سه عدد برابر 213 , 231 , 312 , 321 , 132 , 123 می باشند که BST ساخته شده بـه ازای هر ترتیب برابر است با:



ترتیب 213 و231 منجر به یک درخت می شوند.

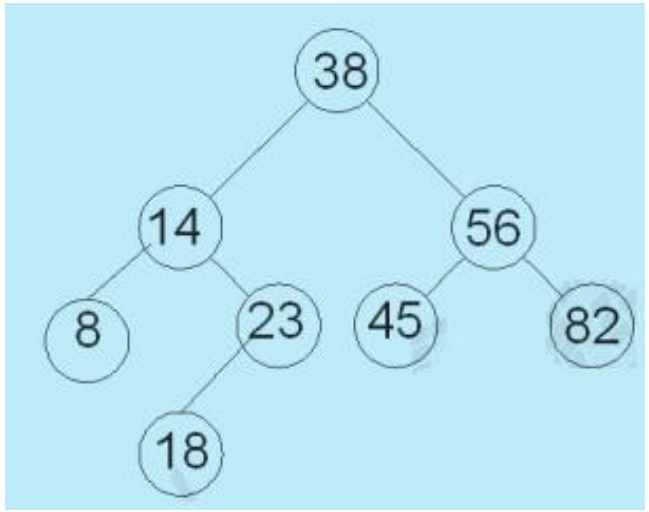
ارتفاع درخت جستجوی دودویی به طور متوسط برابر $\log_2^n$ و در بدترین حالت برابر n می باشد.

در یک BST ، با n گره متمایز که خاصیت MaxHeap هم دارد، داریم: $h = o(n)$

مثال

در BST که به صورت آرایه نمایش داده شده، آیا می توان به جای عدد 56 عدد 35 قرار داد؟

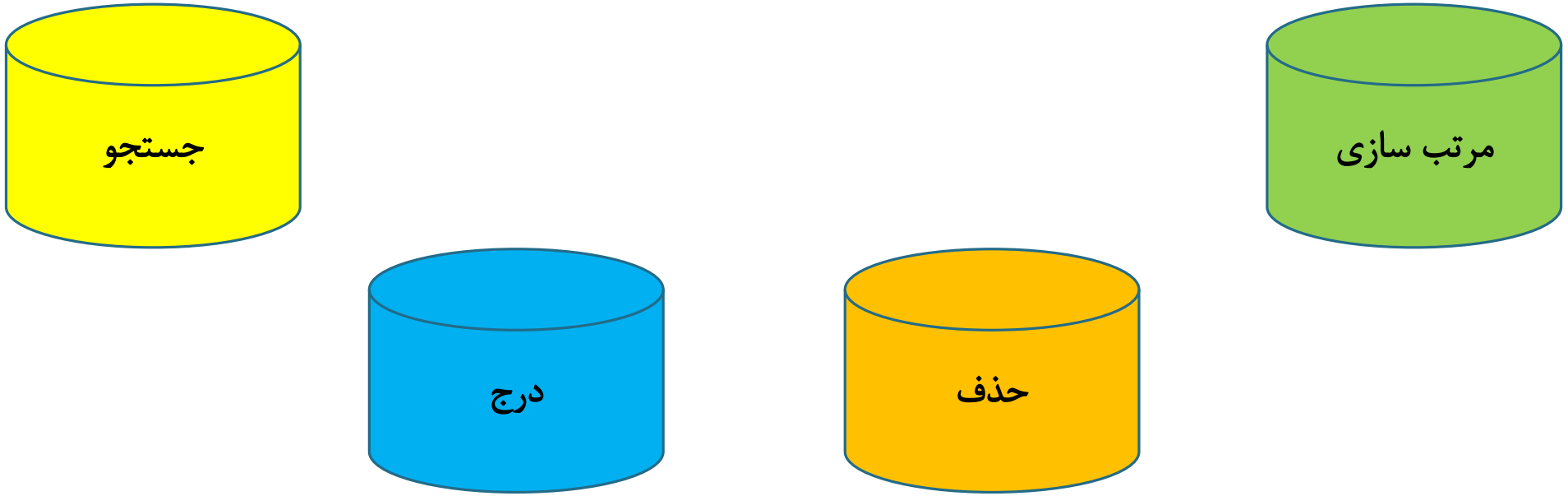
38 , 14 , 56 , 8 , 23 , 45, 82 , - , - , 18



بنابراین مشخص است که اگر به جای عدد 56 عدد 35 قرار گیرد، درخت BST نخواهد بود، چون عدد 35 از ریشه کوچکتر است و نمی تواند در سمت راست ریشه قرار بگیرد.

عملیات بر روی یک BST

می توان عملیاتی چون جستجو،درج،حذف و مرتب سازی را بر روی یک درخت جستجوی دودیی انجام داد.



عملیات بر روی یک BST

جستجوی یک عنصر در BST

برای جستجوی عنصری در درخت جستجوی دودویی، ابتدا مقدار کلید عنصر مورد نظر با ریشه مقایسه می‌شود، اگر برابر باشد به نتیجه رسیده‌ایم، در غیر این صورت اگر کمتر از مقدار ریشه باشد، زیردرخت چپ و اگر بزرگتر از مقدار ریشه باشد، زیر درخت راست را به صورت بازگشتی جستجو می نماییم.

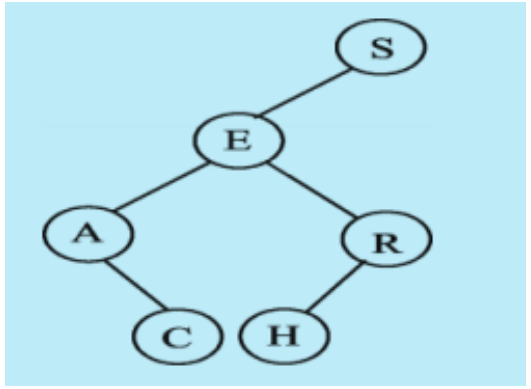
عملیات بر روی یک BST

جستجوی یک عنصر در BST

مثال

یک درخت جستجوی باینری با کلمه "SEARCH" بسازید. تعداد متوسط مقایسه برای پیدا کردن یک کاراکتر در این درخت کدام است؟

پاسخ: برای پیدا کردن S به یک مقایسه نیاز داریم و برای پیدا کردن E به دو مقایسه نیاز داریم(ابتدا با S مقایسه شده و چون کوچکتر است به چپ s می‌رویم و با E مقایسه می‌کنیم)، برای پیدا کردن A و یاR به سه مقایسه و برای C و یا H به چهار مقایسه نیاز داریم.بنابراین متوسط تعداد مقایسه‌ها برابر است با:



$$\frac{1+2+3+3+4+4}{6}=\frac{17}{6}=2/83$$

عملیات بر روی یک BST

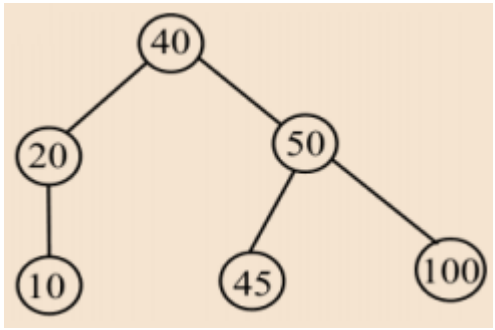
جستجوی یک عنصر در BST

جستجوی موفق(یا ناموفق) در هر درخت دودویی به طور میانگین به اندازه $O(\lg n)$ طول می کشد.
اما جستجو در بدترین حالت، به اندازه $O(n)$ طول می کشد.(اگر درخت یک مسیر باشد.)

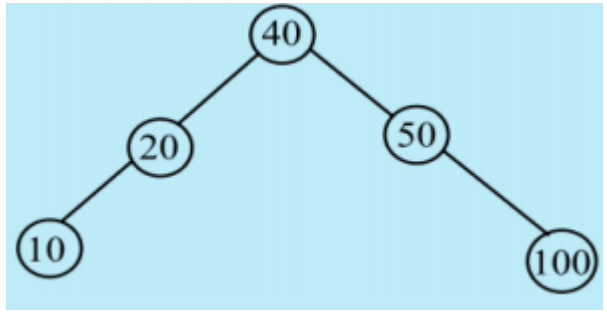
عملیات بر روی یک BST

درج یک عنصر به BST

برای اضافه کردن گره ای به درخت جستجوی دودویی، کلید مورد نظر ابتدا با ریشه مقایسه شده و در صورت کـوچکتر بـودن، به زیر درخت چپ و در صورت بزرگتر بودن به زیر درخت راست اضافه می شود.



اضافه کردن ۴۵



عملیات بر روی یک BST

درج یک عنصر به BST

درج (یا حذف) در BST با n گره و به ارتفاع h در زمان $o(h)$ انجام می‌گیرد.

برای حذف عناصر تکراری یک لیست، یک درخت جستجوی دودویی با آن عناصر می‌سازیم. بـه عبـارتی BST بهتـرین ساختمان داده ها برای حذف داده های تکراری می باشد.

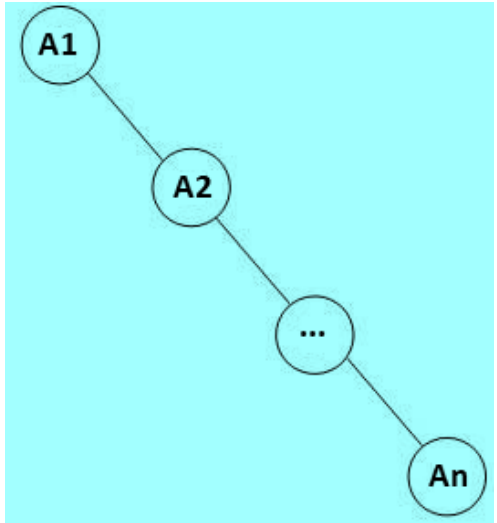
عملیات بر روی یک BST

درج یک عنصر به BST

مثال

فرض کنید n عنصر An,..,A2,A1 از قبل مرتب شده‌اند (A1<A2<…<An). این عناصر را به ترتیب در درخت BST خالی اضافه می‌کنیم. ارتفاع درخت حاصل کدام است؟

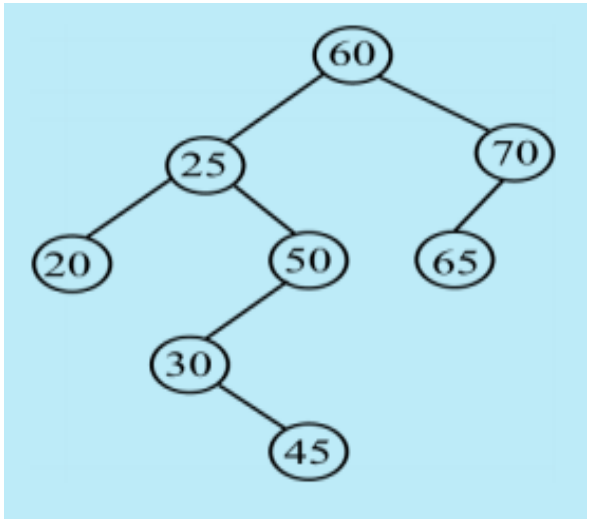
پاسخ: در صورت ساختن یک BST با n عدد که به صورت صعودی مرتب می باشند، درخت اریب به راست با ارتفاع n حاصل می شود.



عملیات بر روی یک BST

مرتب سازی به کمک BST

با ایجاد یک درخت جستجوی دودویی با n عدد و سپس پیمایش میانوندی (inorder) درخت، می توان یک لیست مرتب شده از اعداد را بدست آورد. به این روش مرتب سازی، Tree Sort می گویند.



پیمایش میانوندی

20 , 25 , 30 , 45 , 50 , 60 , 65 , 70

عملیات بر روی یک BST

مرتب سازی به کمک BST

مرتبه اجرایی الگوریتم مرتب سازی درختی در بهترین حالت برابر $o(n \lg n)$ و در بدترین حالت (BST اریب)، برابر $o(n^2)$ می‌باشد.

عملیات بر روی یک BST

پیدا کردن عنصر کمینه در BST

عنصری که کلید آن در BST ، مینیمم است را می توان با دنبال کردن اشاره گرهای فرزندان چپ از ریشه تا رسیدن به NULL پیدا کرد.

مرتبه اجرایی روال های min یا max روی یک درخت BST با ارتفاع h برابر O(h) می باشد.

عملیات بر روی یک BST

حذف یک گره از BST

پس از یافتن گره مورد نظر برای حذف، یکی از سه حالت زیر ممکن است رخ دهد:

۱- اگر گره فاقد فرزند باشد، به سادگی حذف شده و نیازی به تنظیم درخت نمی‌باشد.

۲- اگر گره فقط دارای یک فرزند باشد، فرزند آن به طرف بالا در درخت منتقل می‌شود.

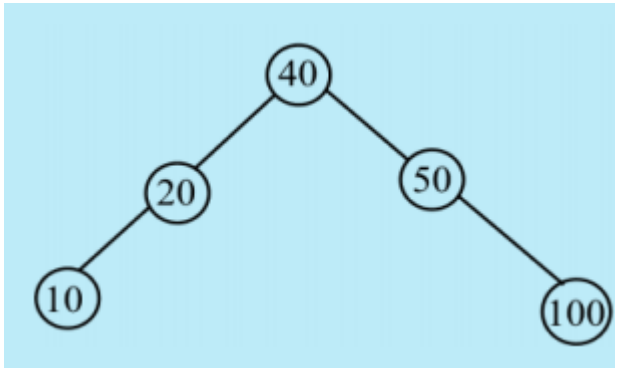
۳- اگر گره دارای دو فرزند باشد، گره بعدی یا قبلی آن در پیمایش میانوندی جای آن را می‌گیرد. به عبارتی عنصر حذف شده، با مقدار بزرگ‌ترین عنصر زیر درخت چپ یا کوچکترین عنصر زیر درخت راست جایگزین می‌شود.

در BST غیر اریب، می‌توان کوچکترین عنصر را با مرتبه $o(\log n)$ حذف کرد.

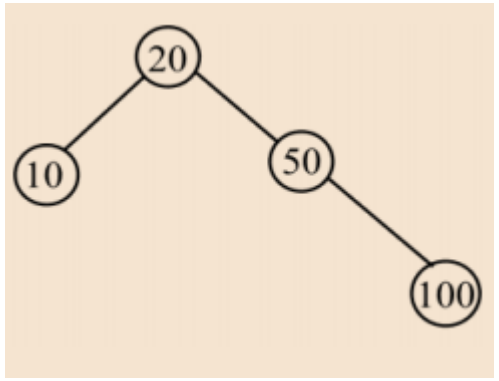
عملیات بر روی یک BST

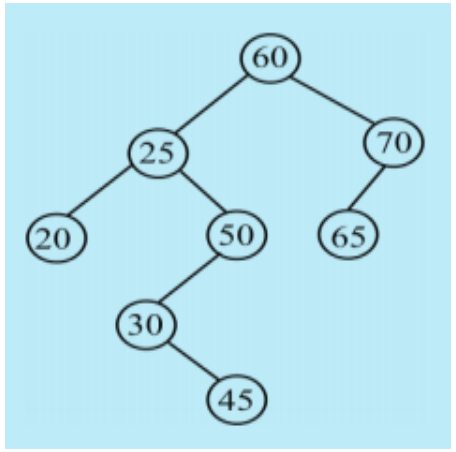
حذف یک گره از BST

مثال

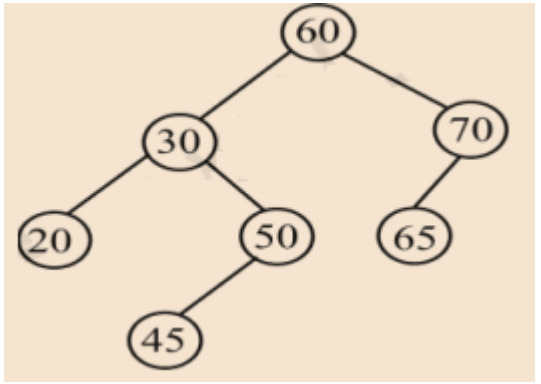


حذف ۴۰



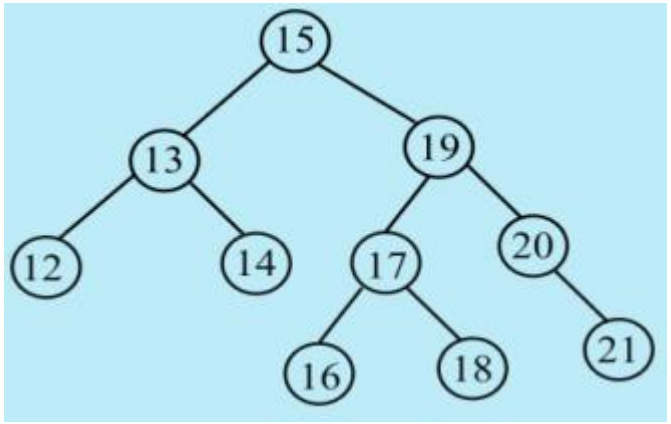


حذف ۲۵



درخت AVL

درخت AVL یک درخت جستجوی دودویی است که حداکثر اختلاف ارتفاع دو زیر درخت چپ و راست آن برابر یک بوده($|h_R - h_L| \leq 1$) و این خاصیت در هر یک از زیر درختان نیز برقرار است. درخت زیر یک AVL می باشد:



تذکر: در درخت AVL ریشه را در سطح صفر فرض می‌کنیم. بنابراین ارتفاع درخت بالا 3 است.

درخت AVL

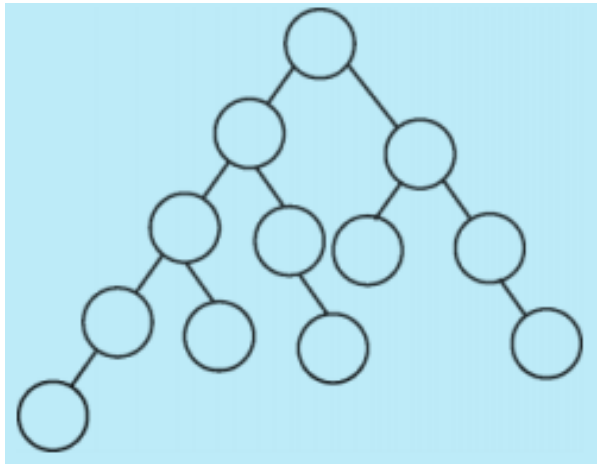
حداقل تعداد گره های مورد نیاز برای ساختن AVL

اگر تعداد عناصر یک درخت AVL کمینه باشد، تعداد عناصر زیر درخت های آن هم باید کمینه باشد. بنابراین حداقل تعداد گره مورد نیاز برای ساختن یک درخت AVL با ارتفاع h برابر است با:

$$G_h = G_{h-1} + G_{h-2} + 1$$
$$G_0 = 1, G_1 = 2$$

مثال

شکل زیر یک درخت AVL با ارتفاع چهار، با حداقل تعداد گره ها یعنی 12 را نشان می‌دهد:

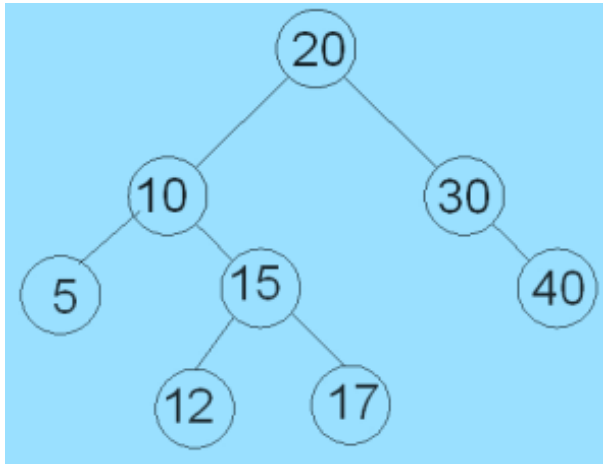


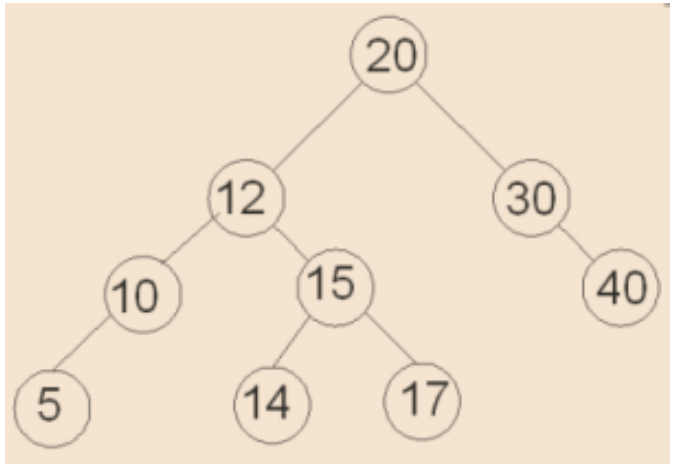
حداکثر ارتفاع یک درخت AVL با n گره از $O(\log n)$ است.

مرتبه اجرایی هر یک از اعمال "جستجو، حذف، درج" در AVL ، برابر $O(\log n)$ می باشد.

مثال

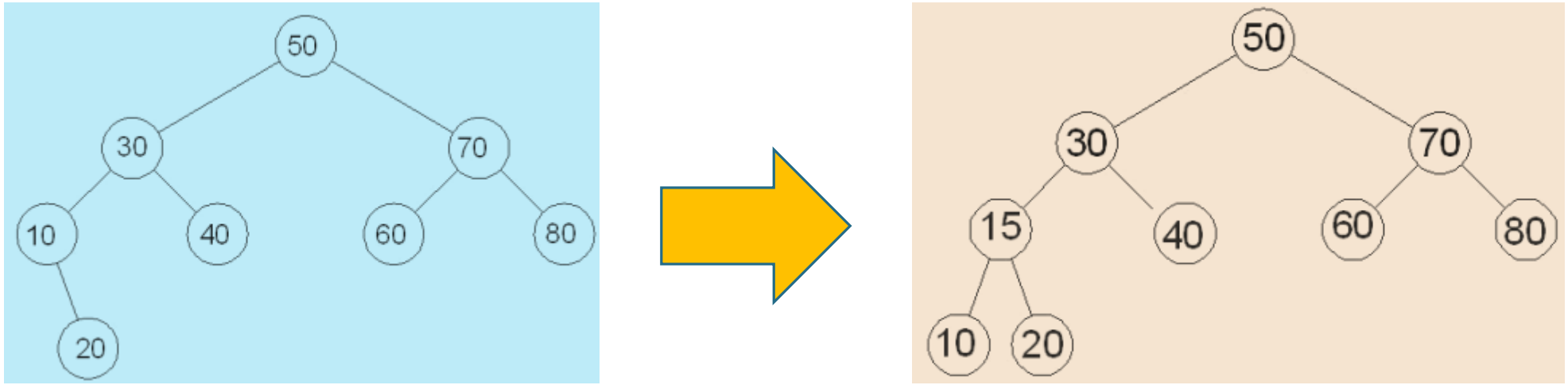
عنصر با کلید 14 را به درخت AVL زیر اضافه کنید.





مثال

عنصر با کلید 15 را به درخت AVL زیر اضافه کنید.



درخت قرمز-سیاه

درخت قرمز- سیاه یک BST است که با اضافه کردن یک بیت به هر گره (معرف رنگ قرمز یا سیاه)، کاری می کنیم که همیشه ارتفاع درخت حداکثر حدود $2\log n$ شود.

در این درخت، هر یک از اعمال درج ، حذف، جستجو و اعمال دیگری که در BST قابل انجام است، می توان در زمان لگاریتمی انجام داد.

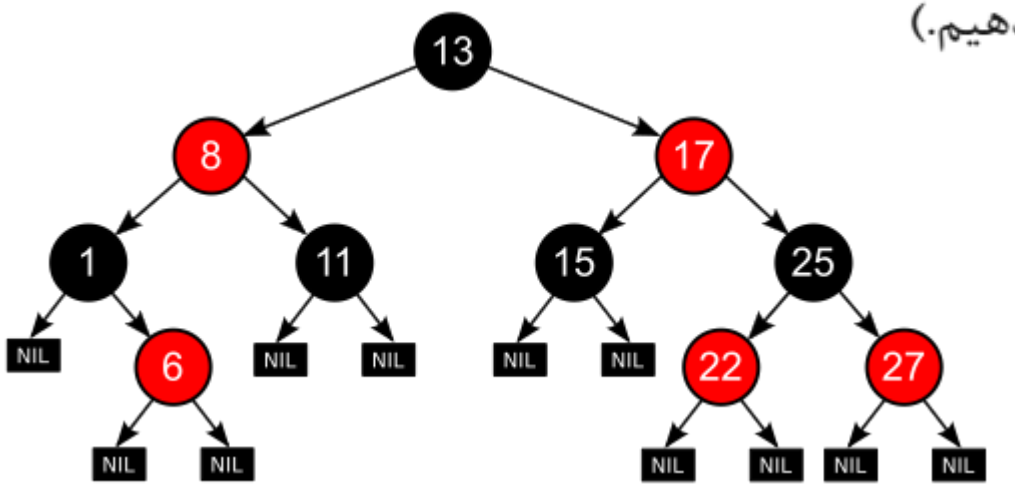
در یک درخت قرمز- سیاه:

۱- رنگ ریشه سیاه است.(این شرط ضروری نیست)

۲- رنگ برگ ها سیاه است.

۳- پدر یک گره قرمز، حتما سیاه است.

۴- به ازای هر گره داخلی x ، تعداد گره های سیاه(بجز خود x) موجود که از x تا هر یک از نواده برگ تهی آن برود، برابر است. (به این تعداد، ارتفاع سیاه x می گوییم و با bh(x) نشان می دهیم.)



حداکثر ارتفاع یک درخت قرمز- سیاه که دارای n گره داخلی باشد، برابر $2\log(n+1)$ است.

ارتفاع یک گره درخت قرمز- سیاه با n گره داخلی $O(\log n)$ است.

درخت مرتبه آماری

درخت مرتبه آماری، یک درخت قرمز- سیاه است که مجموعه ای از n عنصر را که به صورت پویـا در آن درج یـا از آن حـذف می شوند، طوری پیاده سازی می کند که اعمال مرتبه آماری را در هر زمان بتوان در $O(\log n)$ انجام داد.

هر گره در درخت مرتبه آماری، علاوه بر اطلاعات مربوط به خاصیت قرمز- سیاه ، یک مولفه size[x] هم دارد که تعداد عناصر موجود در زیر درختی به ریشه x را نشان می دهد. یعنی:

$$size[x] = size[left[x]] + size[right[x]] + 1$$

درخت ۳-۲

یک درخت ۳-۲ ، درختی است که هر گره داخلی یا ۲ یا ۳ فرزند دارد و همه برگها در عمق یکسان هستند. (درختی کاملا متوازن).

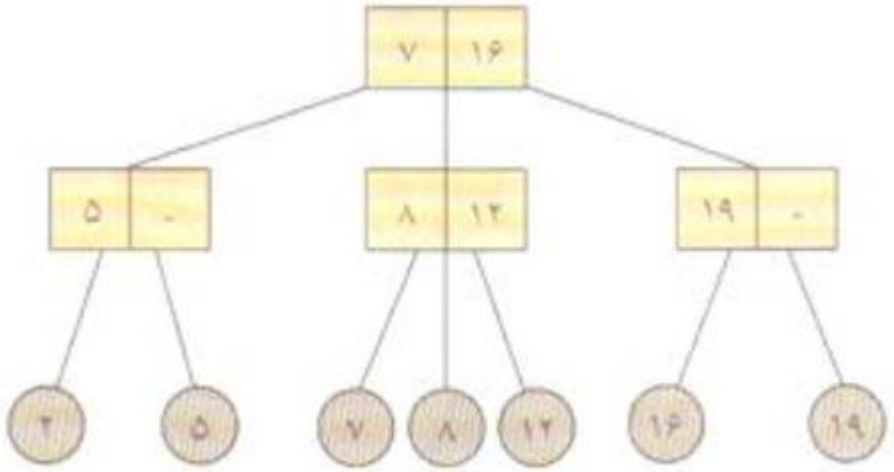
عناصر در این درخت، به ترتیب کلیدشان از چپ به راست فقط در برگها قرار دارند و فرض می شود که کلید هر عنصر تک است.

یک درخت ۳-۲ با یک عنصر، تنها یک برگ است.

در درخت ۳-۲، در هر گره داخلی x ، دو مقدار ذخیره می شود:

۱- کلید کوچکترین عنصر در زیر درخت دوم x

۲- کلید کوچکترین عنصر در زیر درخت سوم x (البته در صورت وجود)



یک درخت ۳-۲ به ارتفاع h ، حداقل 2^h و حداکثر 3^h برگ دارد و همین تعداد عنصر را می تواند در خود ذخیره کند.

به عبارتی اگر h ارتفاع یک درخت ۳-۲ با n عنصر باشد، داریم:

$$\left\lfloor \log_3^n \right\rfloor \leq h \leq \left\lfloor \log_2^n \right\rfloor$$

هر یک از اعمال "درج، حذف و جستجو" در درخت ۳-۲ از $O(\log n)$ می باشند.

مثال

یک درخت ۳-۲ با ۹ برگ، چه تعداد گره داخلی دارد؟

حل: این درخت دو حالت می تواند داشته باشد. با ۴ گره داخلی و با ۷ گره داخلی: