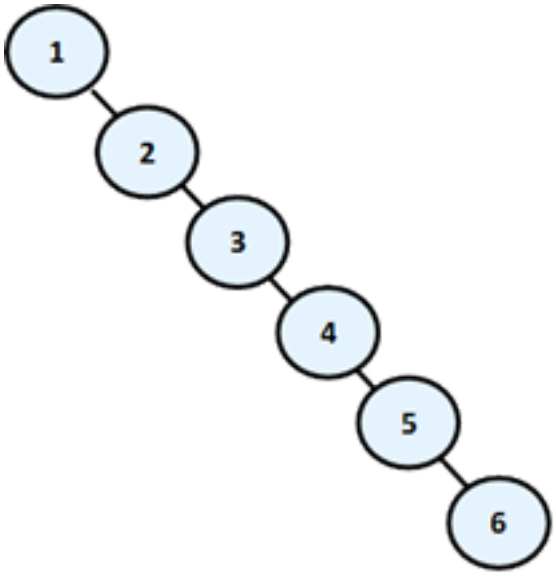
عنوان: **درخت‌ها و گراف قسمت چهارم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۲/۰۶ ۱:۰
آدرس: www.dotnettips.info
گروه‌ها: C#, Data Structures, Tree

در قسمت قبلی مبحث پیاده سازی ساختمان (ساختار) درخت‌های جستجوی دودویی را به پایان رساندیم. در این قسمت قرار است بر روی درخت متوازن بحث کنیم و آن را پیاده سازی نماییم.

درخت متوازن

همانطور که دیدید، عملیات جستجو روی درخت جستجوی دو دویی به مراتب راحت و آسان‌تر است؛ ولی با این حال این درخت در عملیاتی چون درج و حذف، یک نقص فنی دارد و آن هم این است که نمی‌تواند عمق خود را کنترل کند و همینطور به سمت عمق‌های بیشتر و بزرگتر حرکت می‌کند. مثلا ساختار ترتیبی زیر را برای مقداری‌های 1 و 2 و 3 و 4 و 5و 6 در نظر بگیرید:



در این حالت دیگر درخت مانند یک درخت رفتار نمی‌کند و بیشتر شبیه لیست پیوندی است و عملیات جستجو همینطور کندتر و کندتر می‌شود و دیگر مثل سابق نخواهد بود، پیچیدگی برنامه بیشتر خواهد شد و از Log(n) به n می‌رسد. از آنجا که دوست داریم برای عملیات‌های رایجی چون درج و جستجو و حذف، همین پیچیدگی لگاریتمی را حفظ کنیم، از ساختاری جدیدتر بهره خواهیم برد.

درخت دودویی متوازن: درختی است که در آن هیچ برگی، عمقش از هیچ برگی بیشتر نیست.

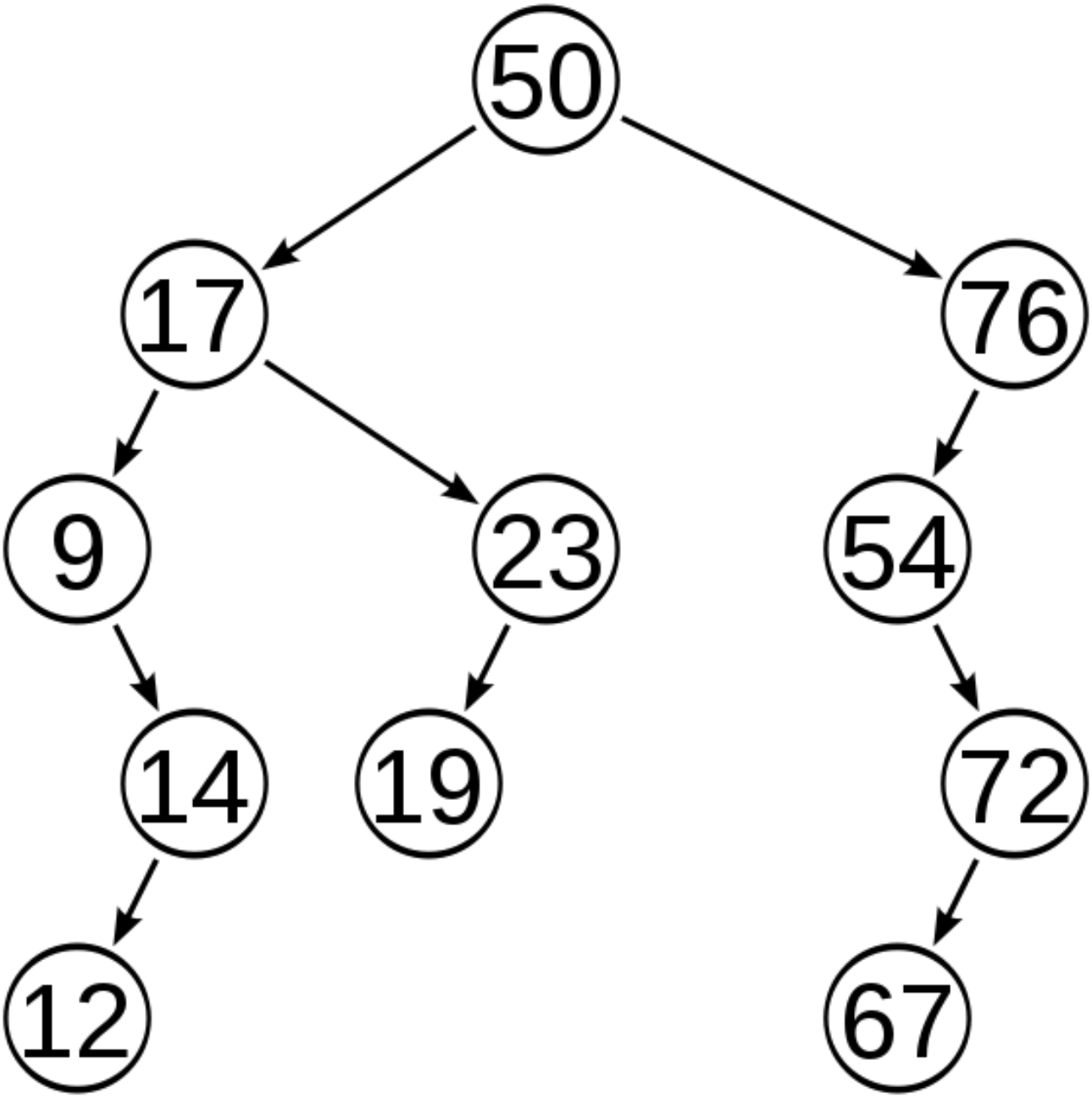
درخت دودویی متوازن کامل: درختی که تفاوتش در تعداد گره‌های چپ یا راست است و حداقل یک فرزند دارند.

درخت دودویی متوازن حتی اگر کامل هم نباشد، در عملیات پایه‌ای چون افزودن، حذف و جستجو در بدترین حالت هم با پیچیدگی لگاریتمی تعداد گام‌ها همراه است. برای اینکه این درخت با به هم ریختگی توازنش روبرو نشود، باید حین انجام عملیات پایه، جایگاه تعداد المان‌های آن بررسی و اصلاح شود که به این عملیات چرخش یا دوران Rotation می‌گویند. انجام این عملیات بستگی دارد که پیاده سازی این درخت به چه شکلی باشد و به چه صورتی پیاده سازی شده باشد. از پیاده سازی‌های این درخت می‌توان به درخت سرخ-سیاه Black Red Tree ، ای وی ال AVL Tree ، اسپلی Splay و ... اشاره کرد.

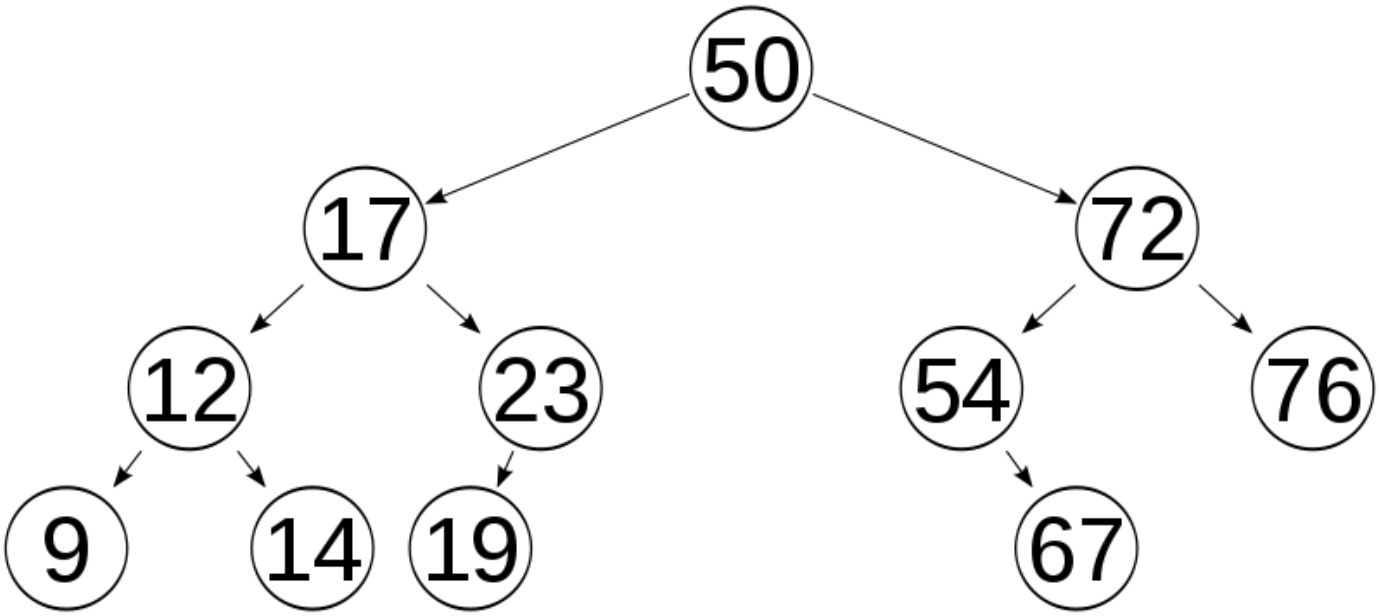
با توجه به موارد بالا میتوانیم به نتایج زیر برسیم که چرا این درخت در هر حالت، پیچیدگی زمانی خودش را در لگاریتم n حفظ

می‌کند:
المان‌ها و عناصرش را مرتب شده نگه می‌دارد.
خودش را متوازن نگه داشته و اجازه نمی‌دهد عمقش بیشتر از لگاریتم n شود.

نمونه ای از درخت جستجوی دو دویی



همان درخت ولی به صورت متوازن با پیاده سازی AVL



درخت‌های متوازن هم می‌متوانند **دو دویی یا باینری** باشند و هم **غیر باینری non-Binary**

درخت‌های دو دویی در انجام عملیات بسیار سریع هستند و در بالا به تعدادی از انواع پیاده سازی‌های آن اشاره کردیم.

درخت‌های غیر باینری نیز مرتب و متوازن بوده، ولی می‌توانند بیش از یک کلید داشته باشند و همچنین بیشتر از دو فرزند. این درخت‌ها نیز عملیات خود را بسیار سریع انجام می‌دهند. از نمونه‌های این درخت می‌توان به `B Tree,B+ Tree,Interval Tree` اشاره کرد.

از آنجا که پیاده‌سازی این نوع درخت کمی دشوار و پیچیده و طولانی است و همچنین پیاده سازی‌های مختلفی دارد؛ تعدادی از منابع موجود را در زیر معرفی می‌کنیم:

در خود دات نت در فضای نام `system.collection.generic` کلاس `TreeSet` یک نوع پیاده سازی از این درخت است که این پیاده سازی از نوع درخت سرخ سیاه می‌باشد و جالب است بدانید، در طی بیست گام می‌تواند در یک میلیون آیتم به جستجو بپردازد ولی خبر بد اینکه استفاده مستقیم از این کلاس ممکن نیست چرا که این کلاس به صورت داخلی `internal` برای استفاده‌ی خود کتابخانه طراحی شده است ولی خبر خوب اینکه کلاس `sortedDictionary` از این کلاس بهره برده است و به صورت عمومی در دسترس ما قرار گرفته است. همچنین کلاس `SortedSet` هم از دات نت 4 نیز در دسترس است.

کتابخانه‌های خارجی جهت استفاده در دات نت که به پیاده‌سازی درخت‌های متوازن پرداخته‌اند:

پیاده سازی Splay Tree با سی شارپ

پیاده سازی AVL به صورت بهینه و آسان برای استفاده

پیاده سازی درخت AVL با کارایی بالا

پیاده سازی درخت AVL به همراه نمایش گرافیکی آن

درخت سرخ-سیاه

کتابخانه ای متن باز برای درخت سرخ-سیاه

درخت متوازن به همراه جست و جو و حذف و پیمایش ها

غیر دودویی‌ها

پیاده سازی B Tree
پیاده سازی Interval Tree
پیاده سازی Interval Tree در سی ++