دانشکده علوم ریاضی و آمار

مدرس: دکتر مجتبی رفیعی نیمسال اول ۱۴۰۰-۱۴۰۱

ساختمان داده‌ها و الگوریتم‌ها

جلسه ۳۶

نگارنده: فاطمه تیموری

۱۴۰۰/۹/۲۱

فهرست مطالب

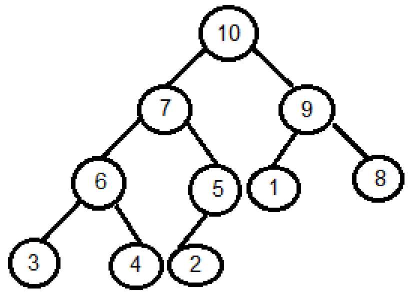
۱ داده ساختار هرم بیشینه (Max-Heap)

یک درخت دودویی تقریبا کامل است که ویژگی‌های زیر را دارا می‌باشد:

۱. هر گروه بزرگتر یا مساوی فرزندانش می باشد،

۲. برگ ها در سطح آخر از سمت چپ به راست پر شده است.

مثال: برای توالی $1, 2, \cdots, 10$، یک درخت (Max-Heap) می‌تواند به صورت زیر ترسیم شود.



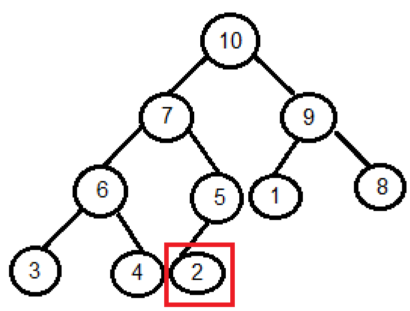
برخی نکات:

۱): در درخت max heap همواره عنصر بیشینه در ریشه قرار دارد.

۲): عمق درخت max heap برای n عنصر همواره از مرتبه‌ی $O(\log n)$ است.

۳): درخت max heap امکان جستجوی سریع یک کلید را فراهم نمی‌کند و جستجو می‌تواند از مرتبه $O(n)$(بدترین حالت) باشد بدین معنا که می‌بایست کل نودهای درخت را پویش کنیم.

به عنوان مثال جستجوی کلید ۲ در درخت رو به رو را در نظر بگیرید:

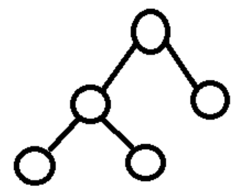


۴): در بررسی درخت BST دیدیم که اگر ۱): یک توالی از اعداد متمایز و ۲): یک اسکلت ثابت برای آن داده شده باشد، آنگاه یک مقداردهی یکتا برای نودهای درخت BST وجود دارد. با این حال با داشتن دو مورد بالا، نمی توان یک مقدار دهی یکتا برای نودهای درخت max heap داشت. در این راستا به مثال زیر دقت کنید.

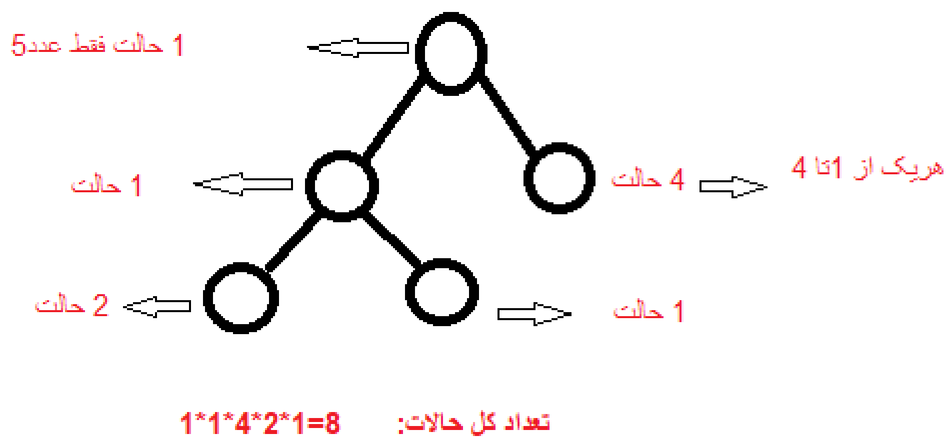
مثال: توالی زیر را در نظر بگیرید:

$$1, 2, 3, 4, 5$$

همچنین اسکلت زیر را نیز برای درخت max heap در نظر بگیرید.

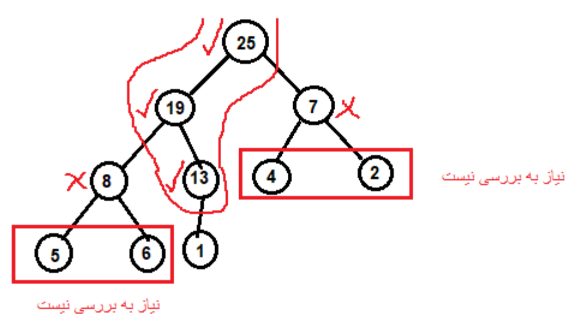


تعداد حالت هایی که می توانیم داشته باشیم به قرار زیر است:

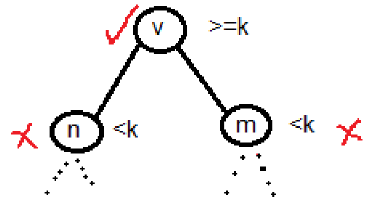


۵): در صورتی که بخواهیم اعداد بزرگتر یا مساوی K را در یک درخت max heap مشخص کنیم، پیچیدگی آن از مرتبه‌ی $O(3v)=O(v)$ است. جایی که v بیانگر تعداد اعدا بزرگتر یا مساوی K در درخت max heap است.

مثال: برای k=۱۰ داریم:

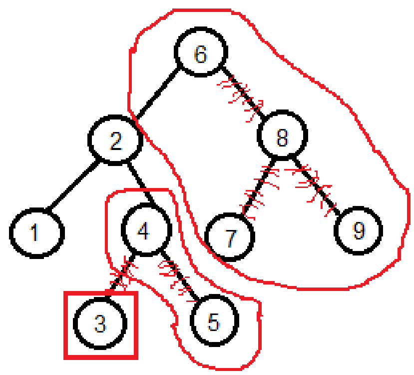


عدد ۳ در مرتبه $O(3v)$ به این خاطر است که در بدترین حالت ریشه بزرگتر یا مساوی از k است ولی فرزندان آن کوچکتر از k هستند و باید ۳ مقایسه برای آن داشته باشیم.



اگر بخواهیم مسئله قبل را با درخت BST متوازن انجام دهیم، پیچیدگی زمان آن از مرتبه O(log n+v) است، جایی که v بیانگر تعداد اعداد بزرگتر یا مساوی kاست و n تعداد گره‌های درخت است.

مثال:



- log n: برای پیدا کردن مکان مناسب برای k (ممکن است خود k در درخت نباشد)،
- v: حالت بندی و پیمایش درخت.

جمع بندی بند ۵: اگر تعداد اعداد گزارش شده (v) کم باشد آنگاه درخت max heap بهتر است و اگر این تعداد زیاد باشد پیچیدگی هر دو (درخت BST و درخت max heap) یکسان است.

۲ ساخت درخت Max-Heap از روی یک توالی

راه حل ۱: ابتدا توالی داده شده را به صورت نزولی مرتب می کنیم، سپس نودهای درخت را سطح به سطح پر می‌کنیم. به عنوان مثال توالی

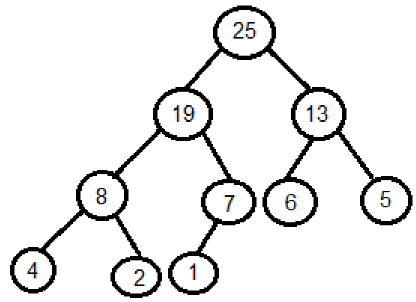
$$5, 6, 1, 8, 13, 19, 4, 2, 7, 25$$

را با استفاده از این راه حل به یک درخت max heap تبدیل می‌کنیم:

- **گام ۱:** مرتب سازی نزولی توالی:

$$25, 19, 13, 8, 7, 6, 5, 4, 2, 1$$

- **گام۲:** تشکیل اسکلت درخت و پرکردن سطح به سطح:



پیچیدگی زمانی راه حل ۱ از مرتبه $O(n \log n+n)=O(n \log n)$ است، جایی که $n \log n$ برای مرتب سازی نزولی است و nبرای پیمایش سطحی اسکلت درخت و بروزرسانی نودهاست.

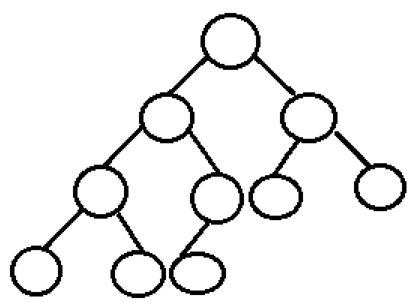
راه حل۲: در این راه حل به دنبال آن هستیم که پیچیدگی زمانی ساخت یک درخت max heap از روی یک توالی با **n** عدد را در مرتبه $O(n)$ انجام دهیم.

برای فهم ساده‌تر، راه حل ۲ را در ادامه همراه با یک مثال تشریح می‌کنیم.

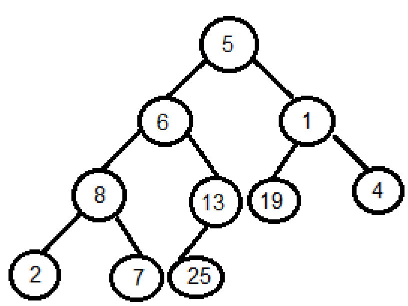
توالی زیر را در نظر بگیرید:

$$5, 6, 1, 8, 13, 19, 4, 2, 7, 25$$

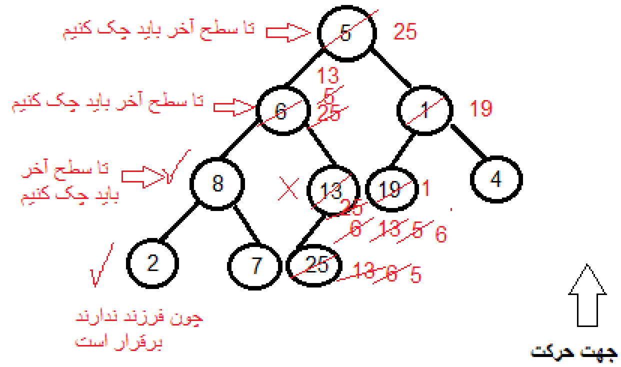
۱): ابتدا اسکلت درخت را می سازیم که شرایط ساختاری max heap را دارا باشد،



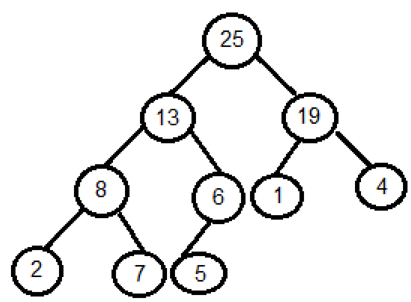
۲): سپس توالی را سطح به سطح در اسکلت بالا درج می‌کنیم:



۳): سپس از سطح آخر به سمت ریشه و سطح به سطح بررسی می‌کنیم که آیا گره پدر با فرزندان خود خصیصه max heap بودن را دارد یا خیر. لازم به ذکر است که بررسی برای هر گروه تا سطح آخر ادامه پیدا می‌کند.



درخت نهایی به صورت زیر بدست می‌آید:



نکته: در هر سطح، متناسب با سطحی که هستیم ممکن است تا برگ مقایسه و جابجایی شکل گیرد.