Data structure

DR. RASTGOO

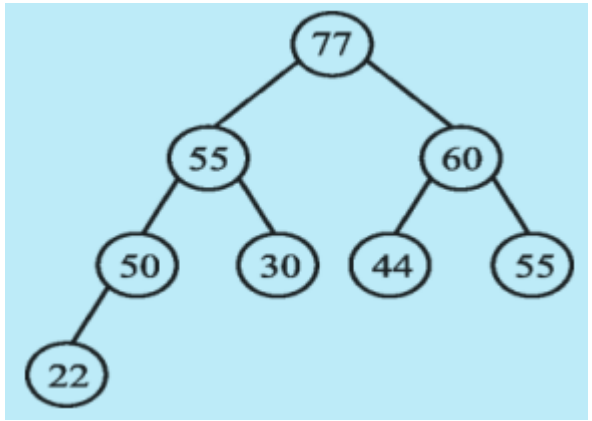
فصل ۸:

درخت های Heap

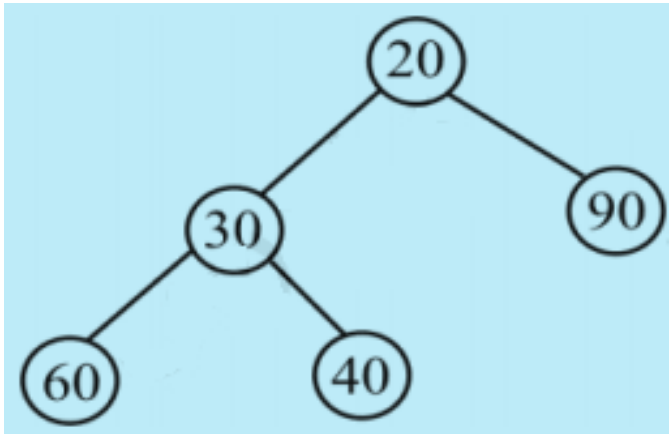
هرم ها (HEAPS)

دو نوع Heap دودویی وجود دارد:

۱- **Max-Heap** : یک درخت دودویی کامل که مقدار کلید هر گره آن بزرگتر یا مساوی مقدار کلیدهای فرزندانش باشد.



۲- **Min-Heap** : یک درخت دودویی کامل که مقدار کلید هر گره آن کوچکتر یا مساوی مقدار کلیدهای فرزندانش باشد.



- ریشه Maxheap حاوی بزرگترین کلید
- ریشه MinHeap حاوی کوچکترین عنصر
- ارتفاع یک هرم با n عنصر برابر $\lfloor \lg n \rfloor$ است.
- تعداد برگ های یک هرم با n عنصر برابر $\left\lceil \frac{n}{2} \right\rceil$ است.

در یک هرم با n گره، عمق یک گره در درایه i ، برابر $\lfloor \lg i \rfloor$ است. $(1 \le i \le n)$

بیشترین تعداد گره های با ارتفاع h در یک هرم با n عنصر برابر است با: $\left\lceil \frac{n}{2^{h+1}} \right\rceil$

کوچکترین عنصر در یک MaxHeap با عناصر متمایز، برگ است. (زیرا در غیر اینصورت دارای فرزند است و باید از آنها بزرگتر باشد، که با "کوچکترین" بودن در تناقض است.)

کوچکترین عنصر در یک MaxHeap با عناصر متمایز را می توان با $\left\lceil \frac{n}{2} \right\rceil - 1$ مقایسه و در زمان $O(n)$ پیدا کرد.

(چون این عنصر در برگ قرار دارد و تعداد برگها نیز برابر $\left\lceil \frac{n}{2} \right\rceil$ می باشد.)

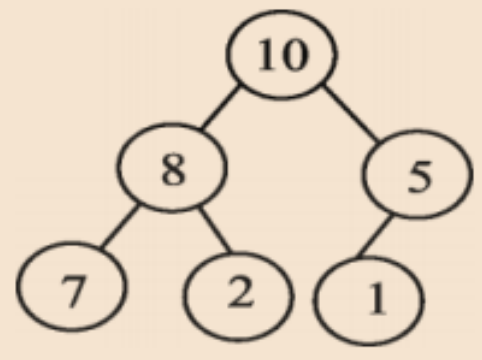
بزرگترین عنصر در یک MinHeap با عناصر متمایز را می توان با $\left\lceil \frac{n}{2} \right\rceil - 1$ مقایسه پیدا کرد.

m امین بزرگ ترین عنصر در یک MaxHeap (یا m امین کوچک ترین عنصر در MinHeap)، می تواند در یکی از خانه های A[2] تا $A[2^m - 1]$ قرار بگیرد. $(m > 1)$

مثال

یک MaxHeap با N عنصر متمایز را در نظر بگیرید که با یک آرایه پیاده سازی شده است. چهارمین بزرگترین عنصر در کدام یک از درایه ها می تواند قرار بگیرد؟

حل: چهارمین بزرگترین عنصر می تواند در هر یک از موقعیت های 2 تا 15 قرار بگیرد. به طور مثال در درختMaxHeap زیر، چهارمین بزرگترین عنصر (مقدار 5) در موقعیت 3 قرار دارد:



درج در درخت Heap

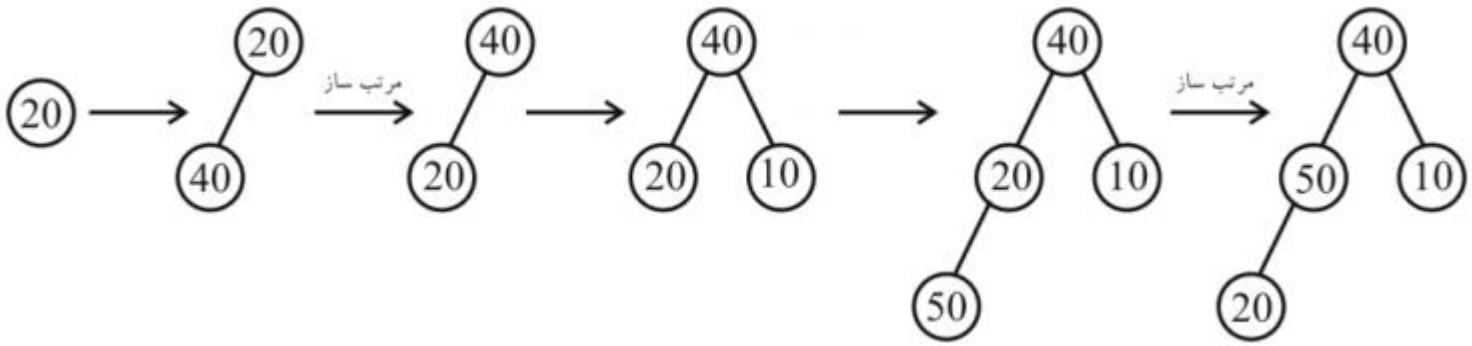
در heap ، درخت از چپ به راست در هر سطح پر شده، سپس سطح بعدی پر می‌شود.

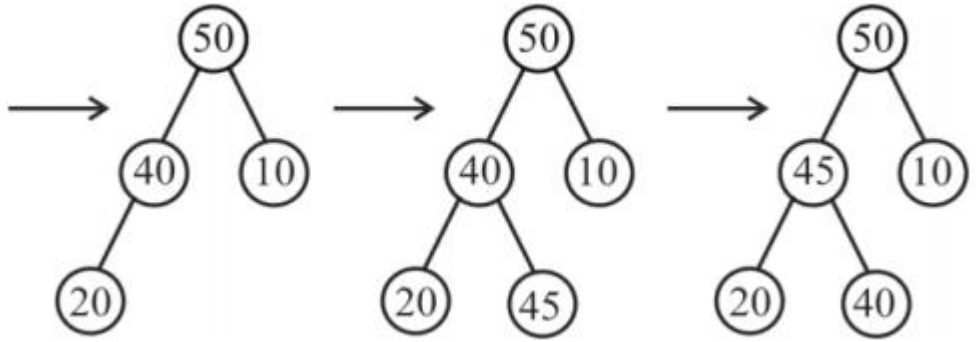
وقتی عنصر جدیدی در درخت وارد می‌شود، ابتدا در سطحی که هنوز به طور کامل پر نشده در چپ ترین جای خالی قرار می‌گیرد

سپس‌عمل مرتب‌سازی درخت(ReHeap) صورت می‌پذیرد و گره در پایین‌ترین سطح تا حد امکان با گره پدرش جا به جا می‌شود.

مثال

ساختن یک MaxHeap با ورود مقادیر 45 , 50 , 10 , 40 , 20 (از چپ به راست) :

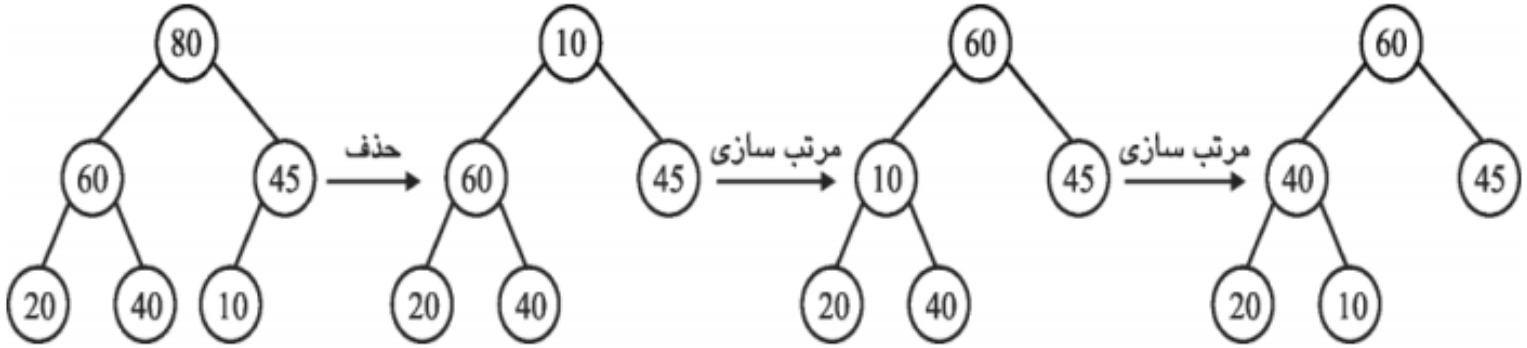




عمل درج یا حذف در درخت Heap به گونه‌ای انجام می‌شود که درخت به صورت Heap باقی بماند. بنابراین بعد از حذف و درج نیاز به عملیات اضافی برای تنظیم درخت می‌باشد.

حذف از درخت Heap

در عمل حذف از Heap، همواره مقدار ریشه حذف شده و سمت راست‌ترین عنصر موجود در پایین‌ترین سطح، در ریشه قرار می‌گیرد و درخت مجدداً تنظیم می‌شود.



اگر همه گره های یک MaxHeap را حذف کرده و به همین ترتیب در یک BST خالی درج کنیم، BST اریب به چپ خواهد شد، چون مقادیر MaxHeap از بزرگ به کوچک بیرون می آیند.

اگر همه گره های یک MaxHeap را حذف کنیم، لیست خروجی نزولی خواهد بود و اگر همه گره های یک MinHeap را حذف کنیم، لیست خروجی صعودی خواهد بود. به این روش مرتب سازی، HeapSort می گویند.

درختی که مقدار کلید هر گره آن بزرگتر یا مساوی کلیدهای فرزندانش باشد را Maxtree و اگر کوچکتر یا مساوی باشد را Mintree می‌گویند.

صف اولویت (Priority Queue)

یکی از رایج ترین کاربردهای heap ، استفاده از آن به عنوان یک صف اولویت کارآمد می باشد.

همانند heap ، دو نوع صف اولویت وجود دارد: صف اولویت ماکزیمم و صف اولویت مینیمم.

یک صف اولویت، ساختمان داده ای برای نگهداری مجموعه S از عناصری است که هر یک دارای یک مقدار مربوطه بنام key است.

یک صف اولویت ماکزیمم (max-priority queue) اعمال زیر را پشتیبانی می کند:

برگرداندن عنصر با بزرگترین کلید	O(1)
درج عنصر x	O(lg n)
حذف و برگرداندن عنصر با بزرگترین کلید	O(lg n)
افزایش مقدار کلید عنصر x به مقدار جدید k	O(lg n)

هر یک از اعمال، "درج یک عنصر، حذف کوچکترین عنصر، کاهش اولویت یک عنصر" را می توان به صورت کارا در یک صف اولویت انجام داد، اما یافتن یک عنصر در صف اولویت را نمی توان به صورت کارا انجام داد، چون یک عنصر خاص جای ثابتی ندارد و باید تمام عناصر جستجو شوند.

روشهای نمایش صف اولویت:

نحوه نمایش صف اولویت	درج	حذف
آرایه	$O(1)$	$O(n)$
لیست پیوندی	$O(1)$	$O(n)$
هرم (heap)	$O(\log n)$	$O(\log n)$

تذکر: در جدول بالا، اگر صف اولویت با آرایه یا لیست پیوندی مرتب شده پیاده سازی شده باشد، آنگاه درج در $O(n)$ و حذف در $O(1)$ انجام می شود.

درخت Deap

deap یک درخت دودویی کامل است که یا تهی است و یا دارای خواص زیر می باشد:

۱- در ریشه عنصری وجود ندارد.

۲- زیر درخت سمت چپ یک minheap است.

۳- زیر درخت سمت راست یک maxheap است.

۴- کلید گره زیر درخت چپ کوچکتر یا مساوی کلید گره متناظر در زیر درخت راست می باشد.