# Selected Problems in CLRS

Section 6

Heap is very useful data structure! It need $O(n)$ time to construct and $O(\lg n)$ time to delete the maximum(minimum) element or insert a element.

## Notifications

***Problem Difficulty*** (count with star)

1. you can solve w/o the brain
2. you can solve if you think a bit
3. you can solve if you think carefully
4. you might solve if you push yourself
5. you can solve if you use other's brain

## Exercise

***6.1-1*** $\star\star$

What are the minimum and maximum numbers of elements in a heap of height h?

> **Note :** heap is complete binary tree

***6.1-2*** $\star\star$

Show that an $n$-element heap has height $\lfloor \lg n \rfloor$

***6.1-4*** $\star\star$

Where in a max-heap might the smallests element reside, assuming that all elements are distinct?

***6.1-7*** $\star\star$

Show that, with the array representation for storing an $n$-element heap, the leaves are the nodes indexed by $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, ..., n$.

***6.2-1*** $\star\star$ (need labor)

Using Figure 6.2 as a model, illustrate the operation of $\textsc{Max}-\textsc{Heapify}(A, 3)$ on the array $A = \langle 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0 \rangle$.

***6.2-4*** $\star$ (think 6.1-7)

What is the effect of calling $\textsc{Max}-\textsc{Heapify}(A, i)$ for $i > A.heap-size/2?

***6.3-1*** $\star\star$ (need labor)

Using Figure 6.3 as a model, illustrate the operation of $\textsc{Build}-\textsc{Max}-\textsc{Heap}$ on the array $A = \langle 5, 3, 17, 10, 84, 19, 6, 22, 9 \rangle$.

***6.3-2*** $\star\star\star$

Why do we want the loop index $i$ in line 2 of $\textsc{Build}-\textsc{Max}-\textsc{Heap}$ to decrease from $\lfloor A.length/2 \rfloor$ to 1 rather than increase from 1 to $\lfloor A.length/2 \rfloor$?

> **Hint :** Think the time complexity of $\textsc{Build}-\textsc{Max}-\textsc{Heap}$, what is the worst case of both methods?

***6.4-2*** $\star\star\star$

Argue the correctness of HEAPSORT using the following loop invariant:

> At the start of each iteration of the **for** loop of lines 2-5, the subarray $A[1..i]$ is a max-heap containing the $i$ samllest element of $A[1..n]$, and the subarray $A[i+1..n]$ contains the $n-i$ largest elements of $A[1..n]$, sorted.

***6.4-3*** $\star\star\star$

What is the running time of HEAPSORT on an array $A$ of length $n$ that is already sorted in increasing order? What about decreasing order?

***6.4-5*** $\star\star\star$

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \lg n)$

***6.5-1*** $\star\star$

Illustrate the operation of $\textsc{Heap}-\textsc{Extract}-\textsc{Max}$ on the heap $A = \langle 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1 \rangle$.

> **Note :** you need to familiar with $\textsc{Max}-\textsc{Heapify}$

***6.5-9*** $\star\star$

Giva an $O(n \lg k)$-time algorithm to merge $k$ sorted list into one sorted list, where $n$ is the total number of elements in all the input lists.

> **Hint :** How many candidates that can be the minimum element? (the miminum element should be the minimum element in each list)