BITS Pilani
Hyderabad Campus

Data Structures and Algorithms Design

Febin.A.Vahab

Building a heap

Running Time of Build-Max-Heap

Trivial Analysis: Each call to Max-Heapify requires log(n) time, we make n such calls ⇒ O(n log n).

Tighter Bound: Each call to Max-Heapify requires time O(h) where h is the height of node i. Therefore running time is

$$\sum_{h=0}^{\log n} \frac{n}{(2^{h+1})} * O(h) = \mathbf{O(n)}$$

Building a heap –Bottom Up

Intuition: uses Max-Heapify in a bottom-up manner to convert unordered array A into a heap.

Key point is that the leaves are already heaps. Elements A[($\lfloor n/2 \rfloor$ + 1) ... n] are all leaves.

So the work starts at parents of leaves...then, grandparents of leaves...etc.

The number of nodes at height h in a max heap-Proof [Not Mandatory]

innovate achieve lead

Proof of tighter bound ($O(n)$) relies on following theorem:

Theorem 1: The number of nodes at height h in a maxheap $\lceil n/2^{h+1} \rceil$.

Height of a node = longest distance from a leaf.

Depth of a node = distance from the root.

– Let H be the height of the tree. If the heap is not a **full** binary tree (because the bottom level is not full), then the nodes at a given depth don't all have the same height. Eg., although all the nodes with depth H have height 0, the nodes with depth H-1 may have either height 0 or 1.

Proof

Theorem : The number of nodes at height h in a maxheap $\lceil n/2^{h+1} \rceil$.

Proof: Let H be the height of the heap.

The proof is by induction on h, the height of each node. The number of nodes in the heap is n.

Basis: Show the thm holds for nodes with $h = 0$. The tree leaves (nodes at height 0) are at depths H and H-1.

Let x be the number of nodes on the (possibly incomplete) lowest level of the heap.

Note that n-x is odd, since the n-x nodes above the last row of the tree form a complete binary tree, which has an odd number of nodes.

Therefore, if n is even, x is odd, and if n is odd, x is even.

Proof

If x is even, then there are x/2 nodes at depth H - 1 that are parents of depth H nodes, so there are $2^{H-1} - x/2$ nodes at depth H-1 that are not parents of depth H nodes. Thus the total number of height-0 nodes is

$$x + 2^{H-1} - x/2 = 2^{H-1} + x/2 = (2^H+x)/2 = \lceil (2^H+x-1)/2 \rceil = \lceil n/2 \rceil$$

If x is odd, then by a similar argument to the even case we obtain that the total number of height 0 nodes is

$$x + 2^{H-1} - (x+1)/2 = 2^{H-1} + (x-1)/2 = (2^H+x-1)/2 = \lceil n/2 \rceil$$

Thus, the # of leaves = $\lceil n/2^{0+1} \rceil$ and the thm holds for the base case.

Proof

Inductive step: Show that if the thm holds for height h-1, it holds for h.

Let n_h be the number of nodes at height h in the n-node tree T.

Consider the tree T′ formed by removing the leaves of T. It has $n' = n - n_0$ nodes. We know from the base case that $n_0 = \lceil n/2 \rceil$, so $n' = n - \lceil n/2 \rceil = \lfloor n/2 \rfloor$

Note that the nodes at height h in T would be at height h-1 if the leaves of the tree were removed--i.e., they are at height h-1 in T′. Letting n'_{h-1} denote the number of nodes at height h-1 in T′, we have $n_h = n'_{h-1}$

$n_h = n'_{h-1} \le \lceil n'/2^h \rceil$ (by the IHOP) $= \lceil \lfloor n/2 \rfloor /2^h \rceil \le \lceil (n/2)/2^h \rceil = \lceil n/2^{h+1} \rceil$