Building Heap

Bottom Up

vs

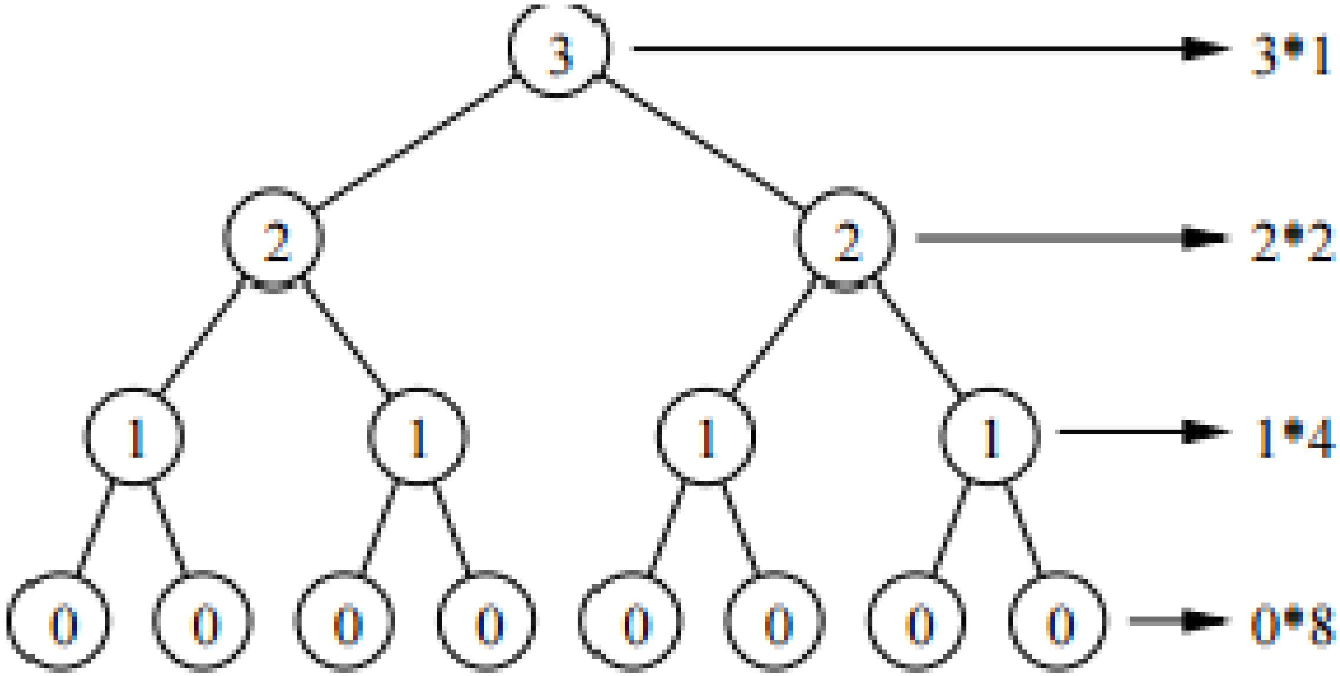
Top Down

BuildHeap Bottom up

1. Assume $n = 2^{(h+1)} - 1$
2. Binary tree is complete
3. The height is h

level	**Number of nodes**	**Length of the path from node to a leaf**
0 (root)	1	h
1	2	h -1
2	4	h -2
h - 1	2^{h-1}	1
h	2^h	0

Example: n = 15, h = 3
Binary tree nodes showing their height



BuildHeap Bottom up

Thus maximum number of operations (in the worst case) is

$$\sum_{j=0}^{h} j2^{h-j} = 2^h \sum_{j=0}^{h} j2^{-j}$$

Since

$$\sum_{j=0}^{h} j2^{-j} < \sum_{j=0}^{\infty} j2^{-j} = 2$$

Note: An operation in this case consists of finding the maximum among three values and perform a swap in the worst case.

BuildHeap Bottom up

Thus maximum number of operations (in the worst case) is

$$\sum_{j=0}^{h} j2^{h-j} < 2^{h+1}$$

Since n = $2^{(h+1)} - 1$,

$$\sum_{j=0}^{h} j2^{h-j} < n + 1 = O(n)$$

BuildHeap Top down

1. Assume $n = 2^{(h+1)} - 1$
2. Binary tree is complete
3. The height is h

level	**Number of nodes**	**Length of the path from node to a root**
0 (root)	1	0
1	2	1
2	4	2
h - 1	2^{h-1}	h - 1
h	2^{h}	h

BuildHeap Top down

Thus maximum number of operations (in the worst case) is

$$\sum_{j=0}^{h} j2^j = O\left(h2^h\right) = O(nlogn)$$

Actual numbers

n	3	7	15	31	63	
h	1	2	3	4	5	
Bottom Up	1	4	11	26	57	n-log(n+1)
Top Down	2	10	34	98	258	(n+1)log(n+1) – 2n