Heap:

Priority Queue:

priority

- finding an item with the highest (i.e., largest) priority
- deleting an item with the highest priority
- adding a new item to the multiset

o Heap:

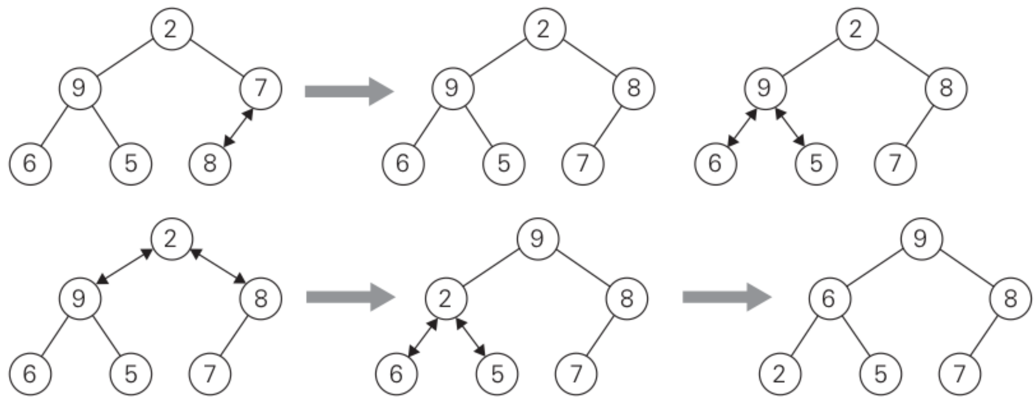
1. The ***shape property***—the binary tree is ***essentially complete*** (or simply ***complete***), i.e., all its levels are full except possibly the last level, where only some rightmost leaves may be missing.
2. The ***parental dominance*** or ***heap property***—the key in each node is greater than or equal to the keys in its children. (This condition is considered automatically satisfied for all leaves.)[5]

o Properties of heaps:

1. There exists exactly one essentially complete binary tree with $n$ nodes. Its height is equal to $\lfloor \log_2 n \rfloor$.
2. The root of a heap always contains its largest element.
3. A node of a heap considered with all its descendants is also a heap.
4. A heap can be implemented as an array by recording its elements in the top-down, left-to-right fashion. It is convenient to store the heap's elements in positions 1 through $n$ of such an array, leaving $H[0]$ either unused or putting there a sentinel whose value is greater than every element in the heap. In such a representation,
   - **a.** the parental node keys will be in the first $\lfloor n/2 \rfloor$ positions of the array, while the leaf keys will occupy the last $\lceil n/2 \rceil$ positions;
   - **b.** the children of a key in the array's parental position $i$ $(1 \le i \le \lfloor n/2 \rfloor)$ will be in positions $2i$ and $2i + 1$, and, correspondingly, the parent of a key in position $i$ $(2 \le i \le n)$ will be in position $\lfloor i/2 \rfloor$.

o Constructing a heap from a given list of array's:

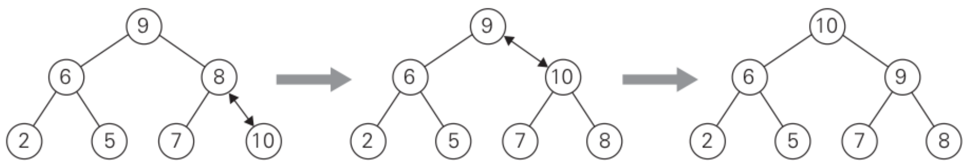
o Bottom-up heap Construction:

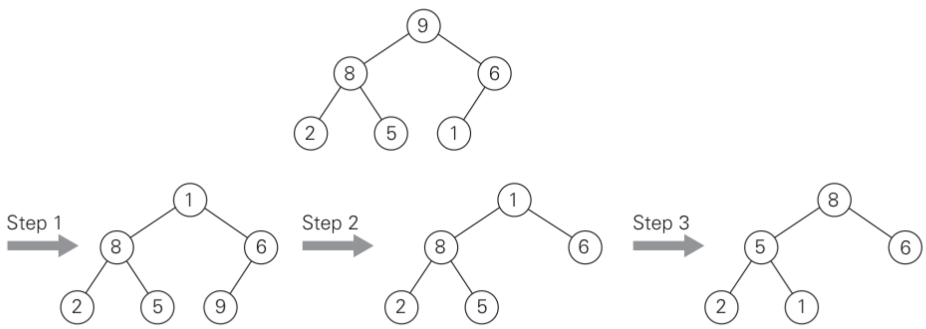



**ALGORITHM** *HeapBottomUp*$(H[1..n])$

    //Constructs a heap from elements of a given array
    // by the bottom-up algorithm
    //Input: An array $H[1..n]$ of orderable items
    //Output: A heap $H[1..n]$
    **for** $i \leftarrow \lfloor n/2 \rfloor$ **downto** 1 **do**
        $k \leftarrow i$;    $v \leftarrow H[k]$
        $heap \leftarrow$ **false**
        **while not** $heap$ **and** $2 * k \le n$ **do**
            $j \leftarrow 2 * k$
            **if** $j < n$    //there are two children
                **if** $H[j] < H[j+1]$  $j \leftarrow j + 1$
            **if** $v \ge H[j]$
                $heap \leftarrow$ **true**
            **else** $H[k] \leftarrow H[j]$;    $k \leftarrow j$
        $H[k] \leftarrow v$

o Top-down heap construction →






o Heapsor-