Heap Sort

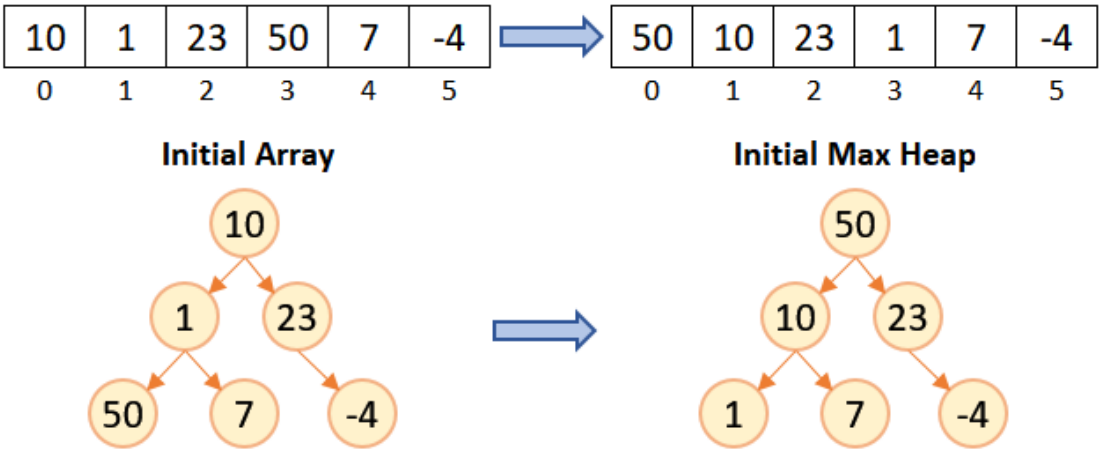
Heap sort uses property of heap to sort the array. It involves building a max heap for increasing order sorting which contains largest element of the heap at the root. For decreasing order sorting min heap is used which contains smallest element of the heap at the root. The process step of for increasing order sorting of heap sort.

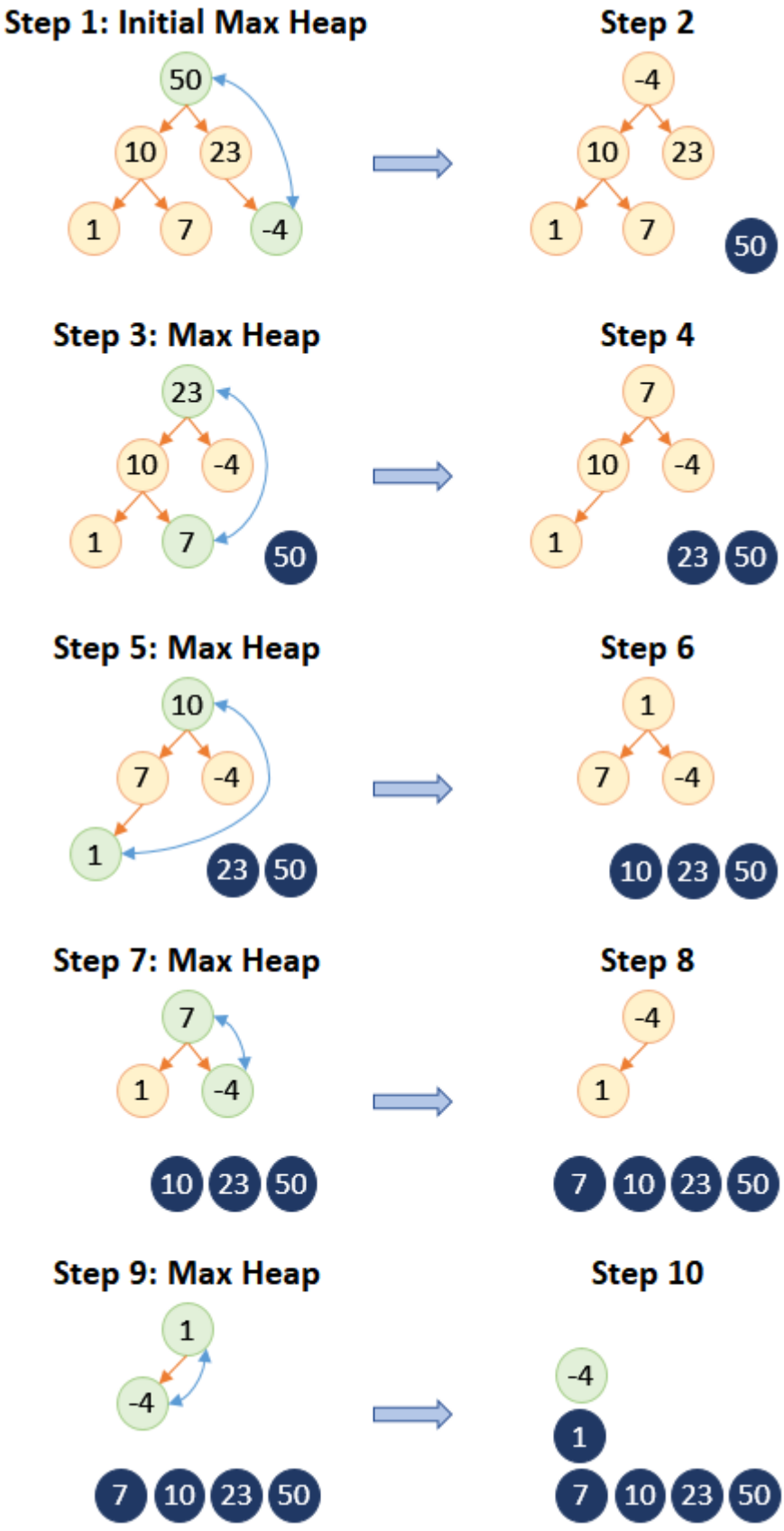
Heap Sort in Java

A complete binary tree is a tree in which each parent node has exactly two child nodes, and its all the levels are completely filled, except the possible last level, in which all the nodes are as far as possible.

- Heap sort, sorts the array by converting them into heaps, which are then converted into min heap and max heap, as required.
- The converted Heaps have the largest element at the top node, and after each iteration we remove the root node from the heap, and move it to it right position in the array, you can have a better idea on the

Algorithm
HEAP_SORT(ARR, N)
Step 1: [Build Heap H]
Repeat for i=0 to N-1
CALL INSERT_HEAP(ARR, N, ARR[i])
[END OF LOOP]
Step 2: Repeatedly Delete the root element
Repeat while N > 0
CALL Delete_Heap(ARR,N,VAL)
SET N = N+1
[END OF LOOP]
 Step 3: END


10 1 23 50 7 -4
0 1 2 3 4 5
50 10 23 1 7 -4
0 1 2 3 4 5
Initial Array
Initial Max Heap
10
1 23
50 7 -4
50
10 23
1 7 -4


Step 1: Initial Max Heap
50
10
23
1
7
-4
Step 2
-4
10
23
1
7
50
Step 3: Max Heap
23
10
-4
1
7
50
Step 4
7
10
-4
1
23
50
Step 5: Max Heap
10
7
-4
1
23
50
Step 6
1
7
-4
10
23
50
Step 7: Max Heap
7
1
-4
10
23
50
Step 8
-4
1
7
10
23
50
Step 9: Max Heap
1
-4
7
10
23
50
Step 10
-4
1
7
10
23
50

The time complexity of creating a heap is *Θ(N)* and time complexity of creating a max heap is *Θ(logN)* and overall time complexity of heap sort is *Θ(NlogN)*.