Min-heap and Max-heap

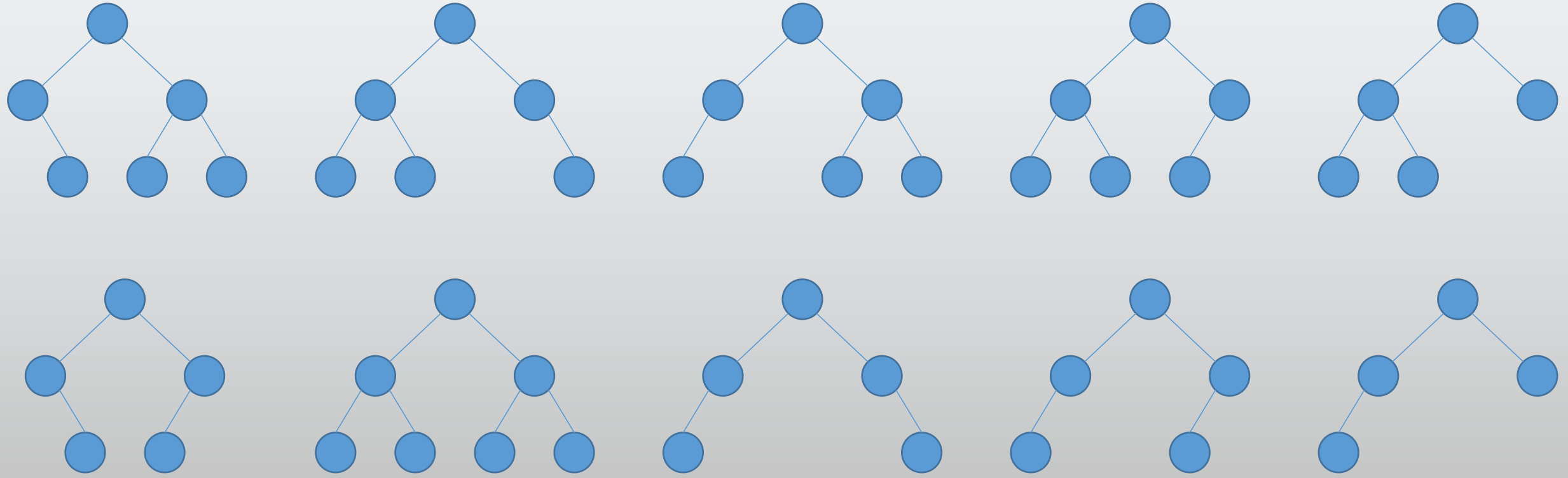
by wwy

What is heap

- Complete binary tree
  - A complete binary tree is a special type of binary tree where all the levels of the tree are filled completely except the lowest level nodes which are filled from as left as possible.
  - A binary tree has a limitation as any node of the tree has at most two children: a left and a right child.

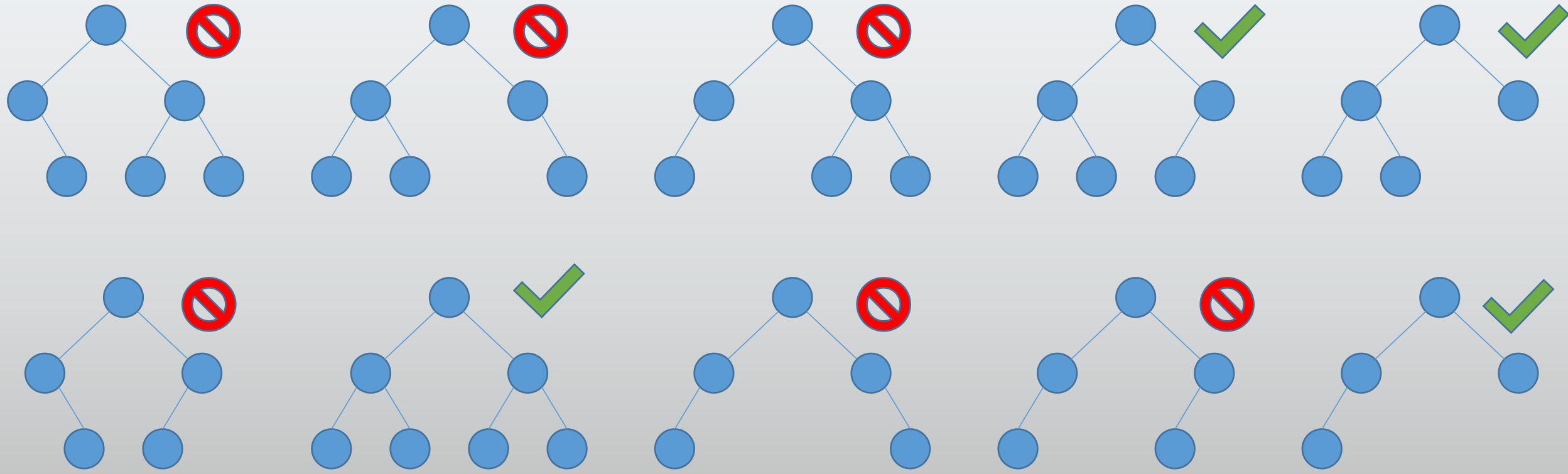
Heap & Complete binary tree

- Which is complete binary tree



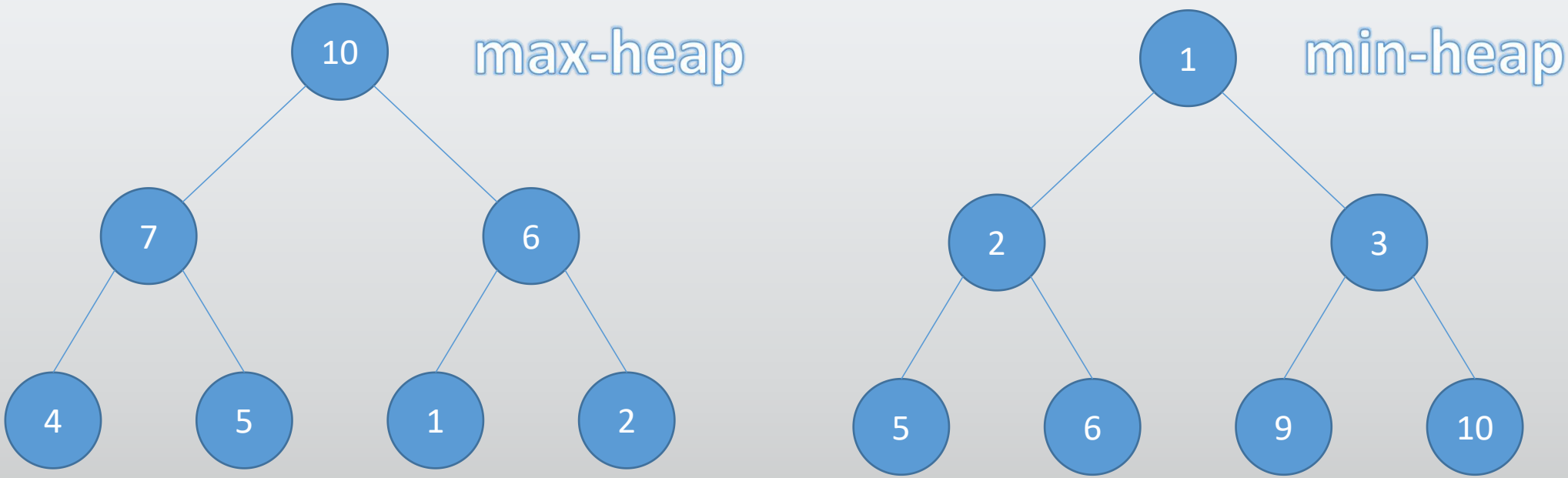
Heap & Complete binary tree

- Which is complete binary tree



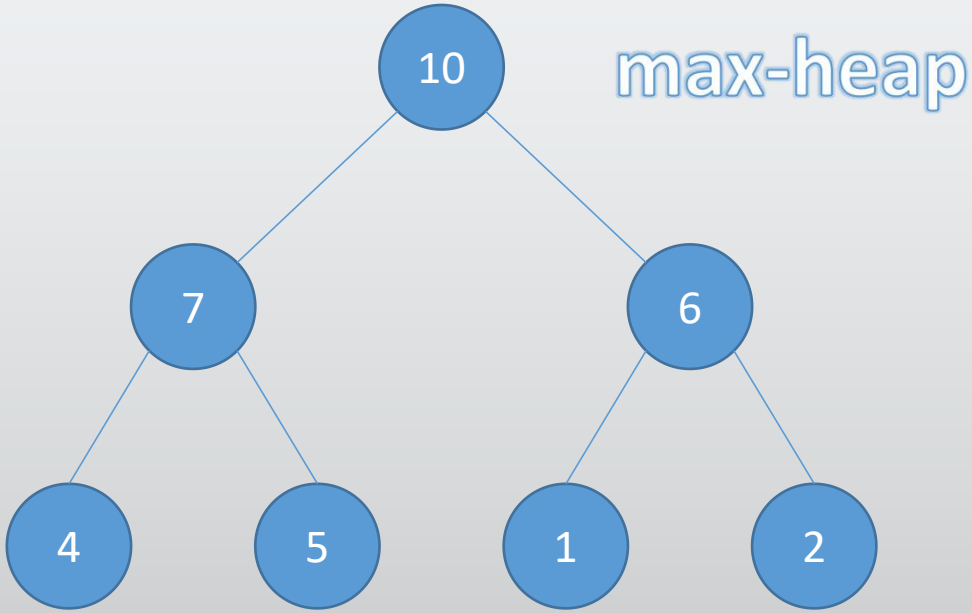
Max/Min heap

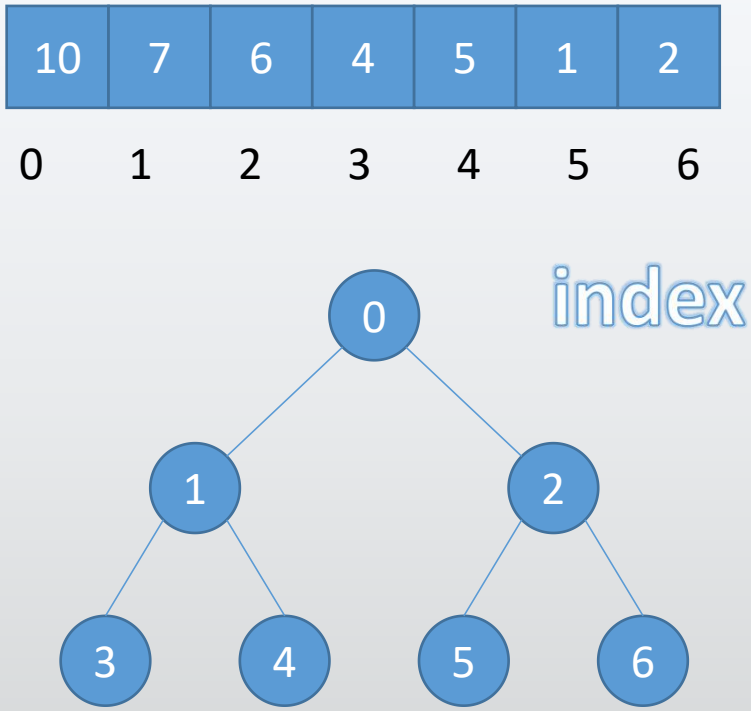
- Two types of heap



Max/Min heap

- Storage

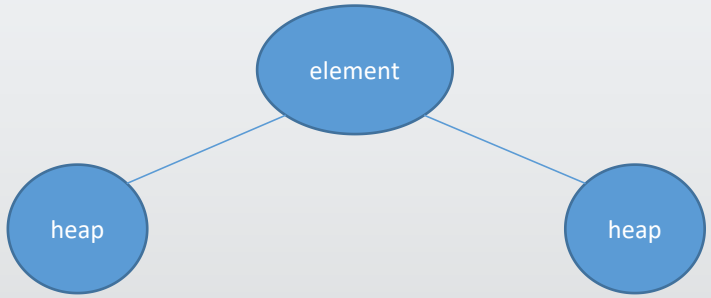




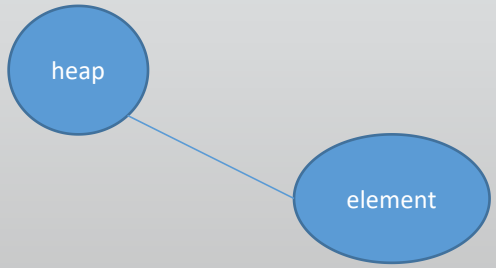
$index\_root$: i
$index\_left$: $2i + 1$
$index\_right$: $2i + 2$

Basic operation

- Percolate Down
- the element need to be moved down to maintain the heap's property



- Percolate UP
- the element need to be moved up to maintain the heap's property



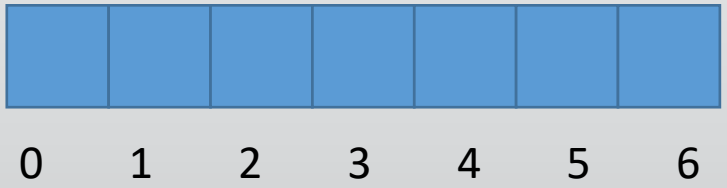
O(logN)

Build Max-heap top-down

- Following a **top-down** approach
- Insert element to end of the heap
- Do the percolate up

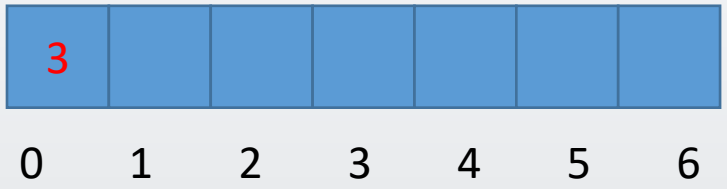
- Example

build $array$ $[3,4,5,6,1,7,8]$ to $max - heap$



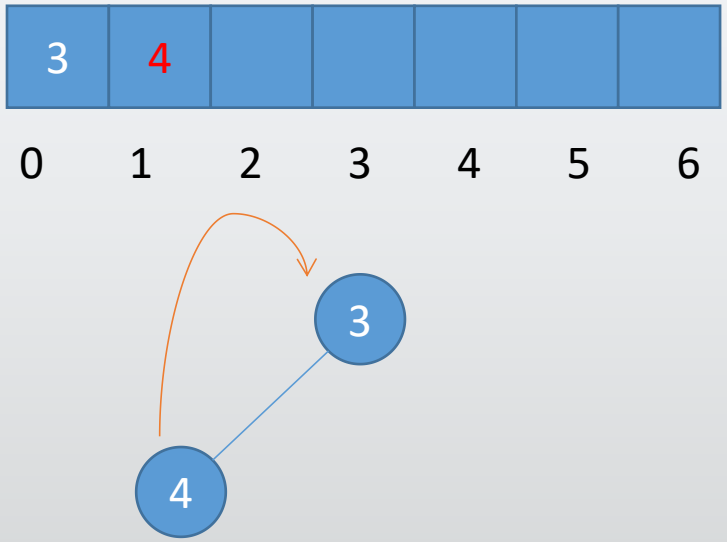
Build Max-heap top-down

$array\ [3,4,5,6,1,7,8]$

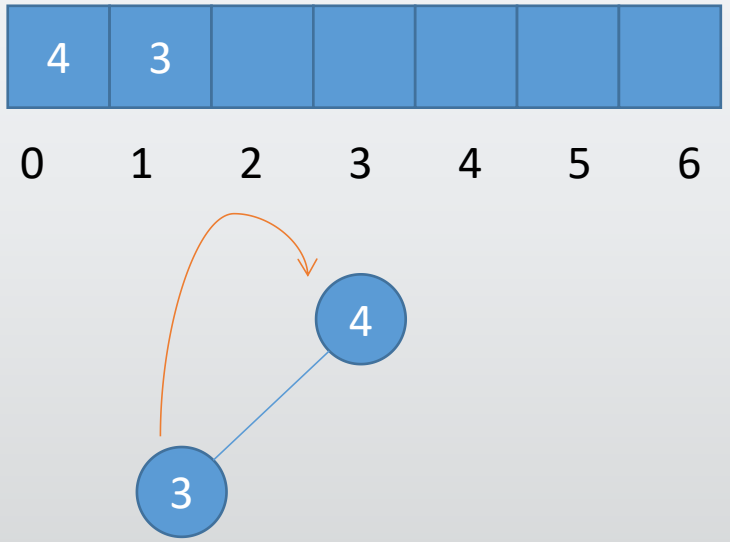


Build Max-heap top-down

$array$ [3,4,5,6,1,7,8]

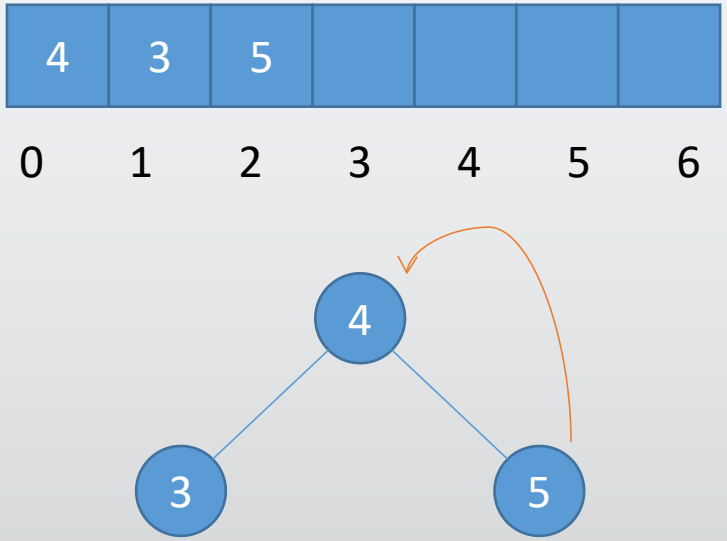


$array$ [3,4,5,6,1,7,8]

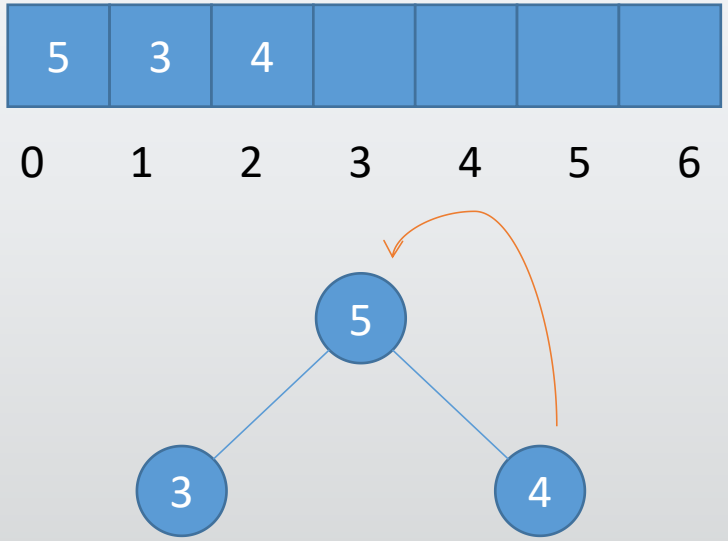


Build Max-heap top-down

$array$ [3,4,5,6,1,7,8]

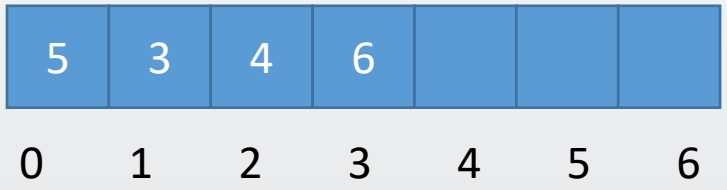


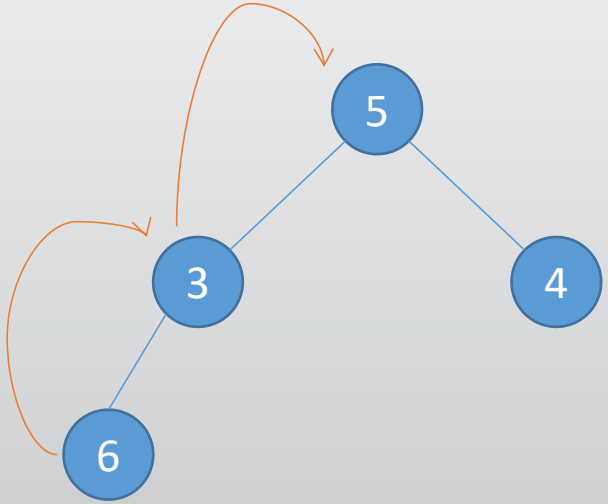
$array$ [3,4,5,6,1,7,8]



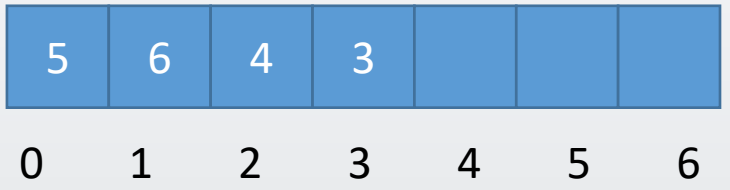
Build Max-heap top-down

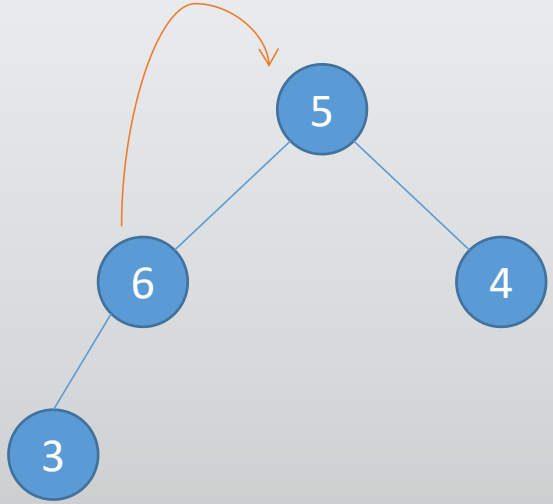
$array$ [3,4,5,6,1,7,8]



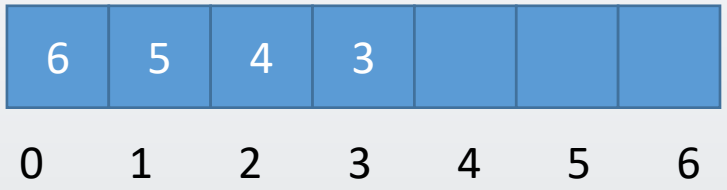


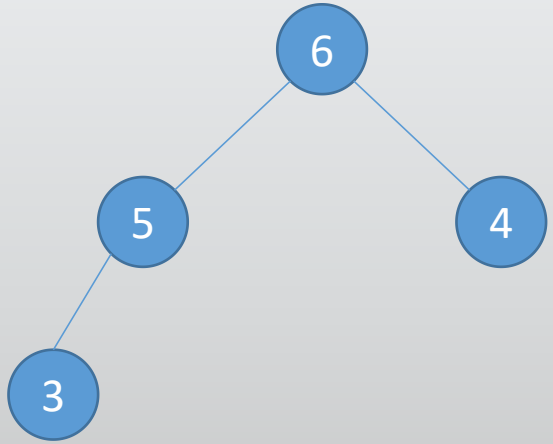
$array$ [3,4,5,6,1,7,8]





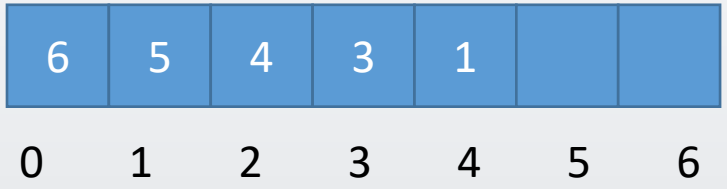
$array$ [3,4,5,6,1,7,8]

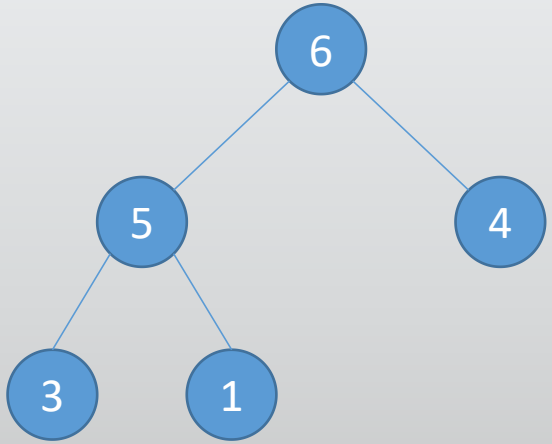




Build Max-heap top-down

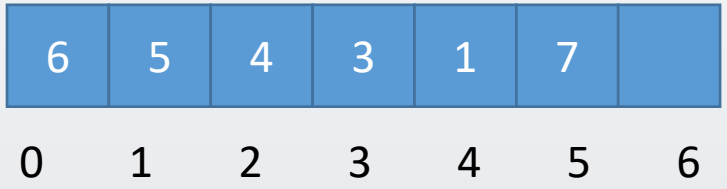
$array$ [3,4,5,6,1,7,8]

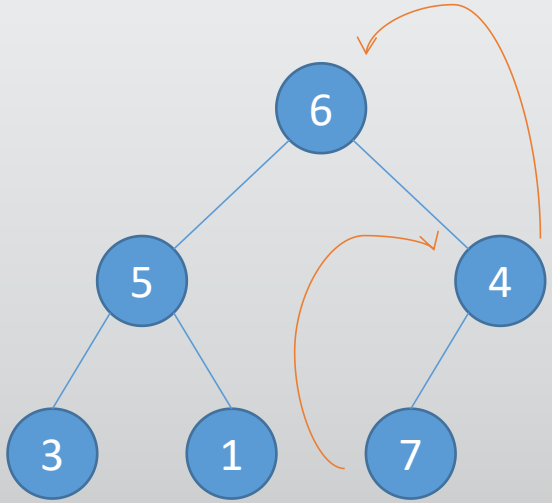




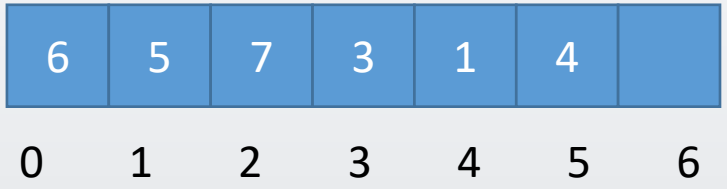
Build Max-heap top-down

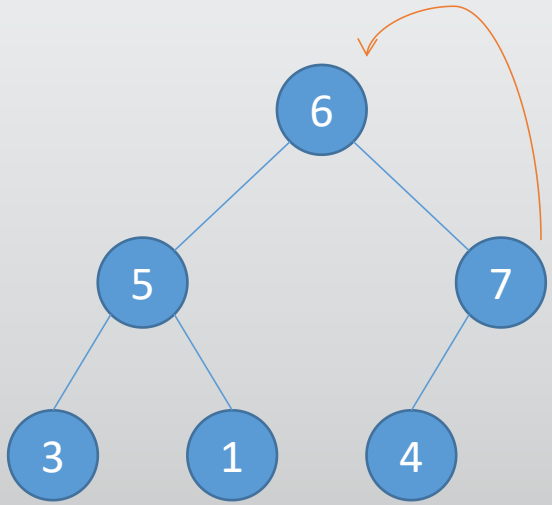
$array$ [3,4,5,6,1,7,8]



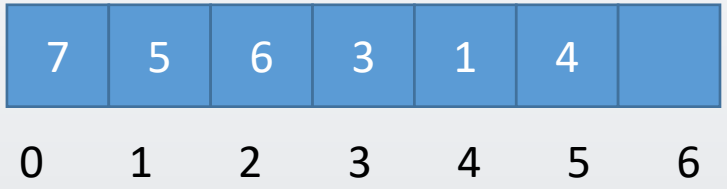


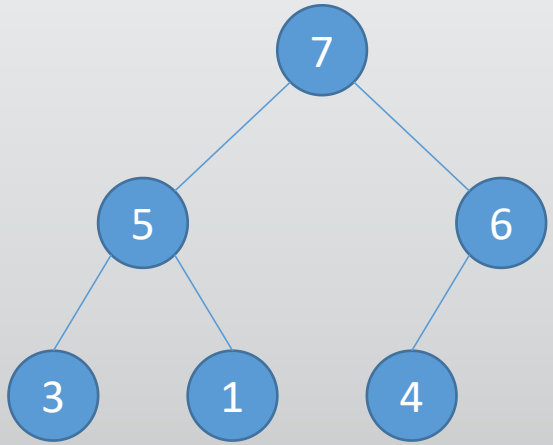
$array$ [3,4,5,6,1,7,8]





$array$ [3,4,5,6,1,7,8]

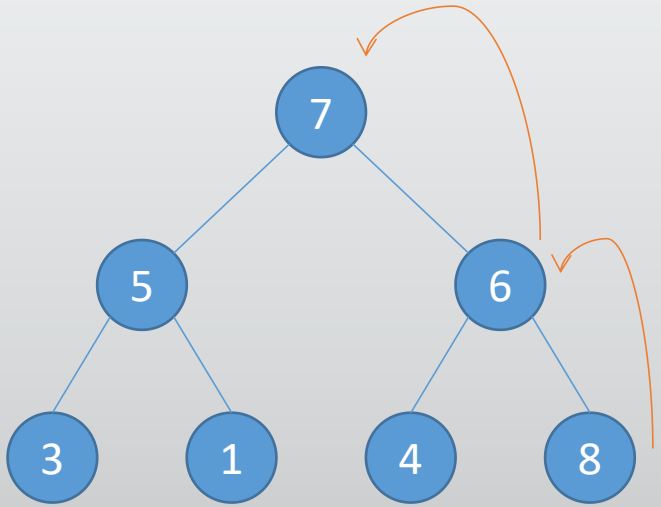




Build Max-heap top-down

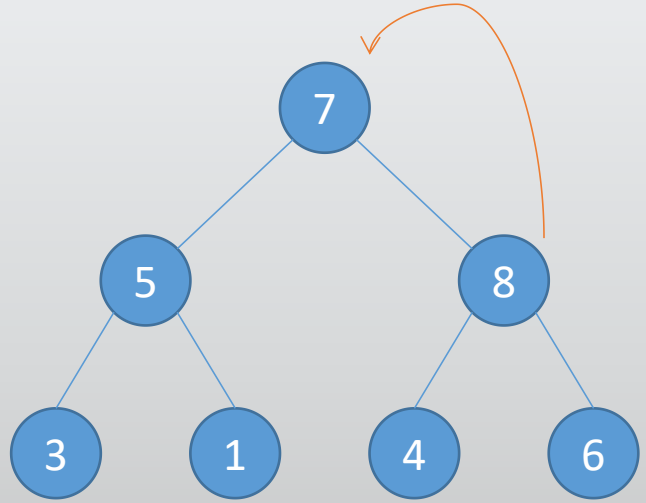
$array$ [3,4,5,6,1,7,8]

7	5	6	3	1	4	8
0	1	2	3	4	5	6



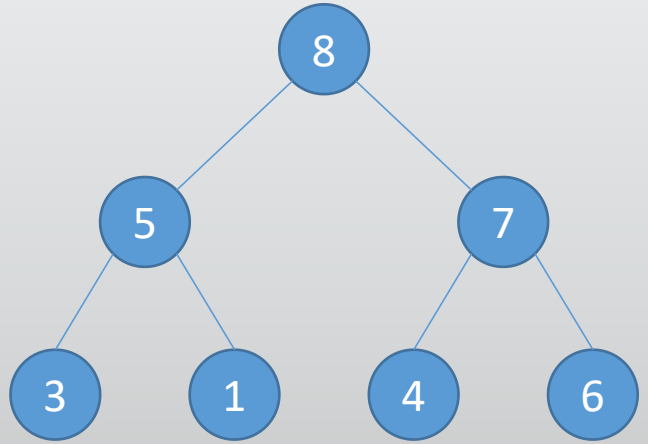
$array$ [3,4,5,6,1,7,8]

7	5	8	3	1	4	6
0	1	2	3	4	5	6



$array$ [3,4,5,6,1,7,8]

8	5	7	3	1	4	6
0	1	2	3	4	5	6



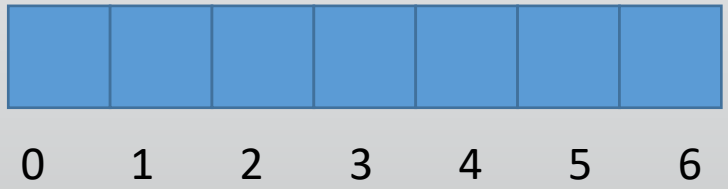
O(NlogN)

Build Max-heap bottom up

- Following a **bottom-up** approach
- Build the complete binary tree
- Do the percolate down of every father element

- Example

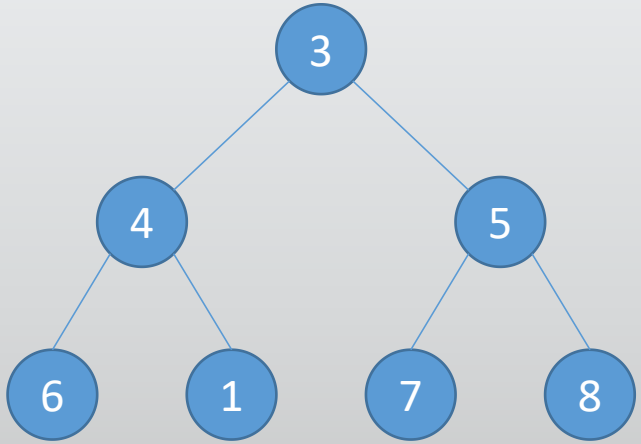
build $array$ $[3,4,5,6,1,7,8]$ to $max - heap$



Build Max-heap bottom up

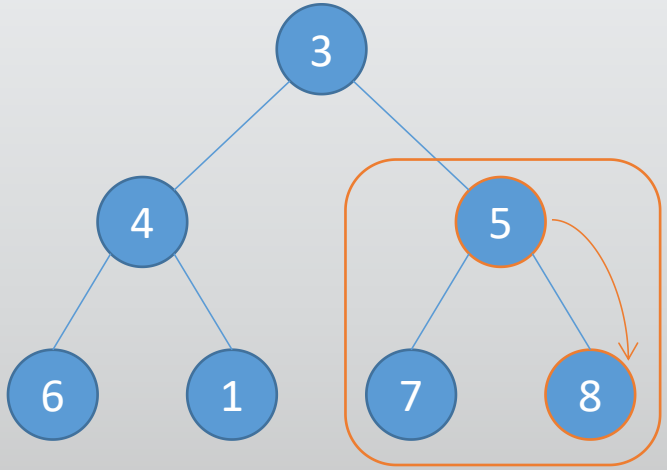
$array$ [3,4,5,6,1,7,8]

3	4	5	6	1	7	8
0	1	2	3	4	5	6



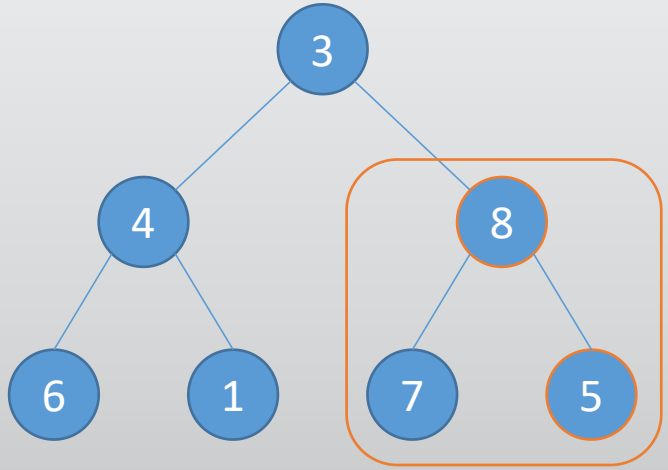
$array$ [3,4,5,6,1,7,8]

3	4	8	6	1	7	5
0	1	2	3	4	5	6



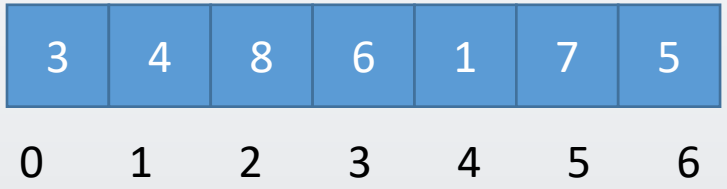
$array$ [3,4,5,6,1,7,8]

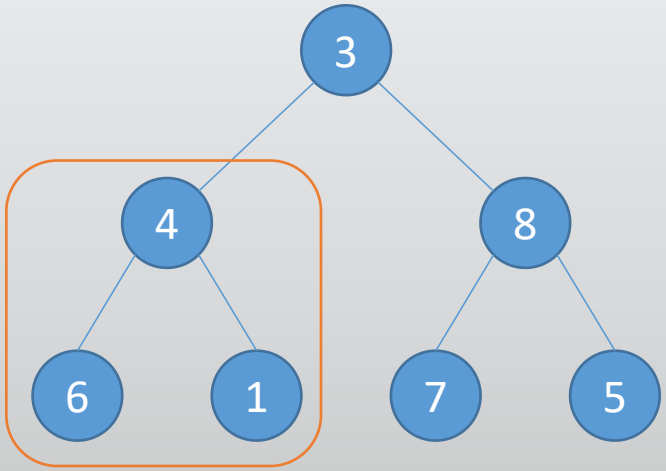
3	4	8	6	1	7	5
0	1	2	3	4	5	6



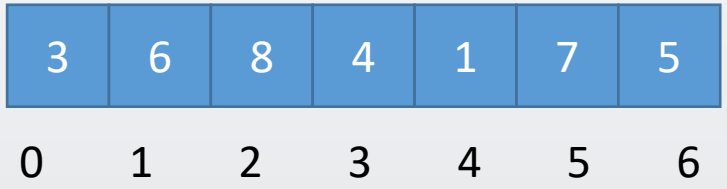
Build Max-heap bottom up

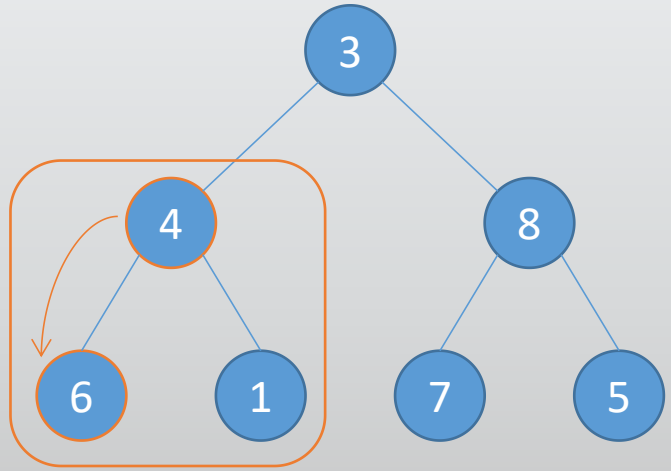
$array$ [3,4,5,6,1,7,8]



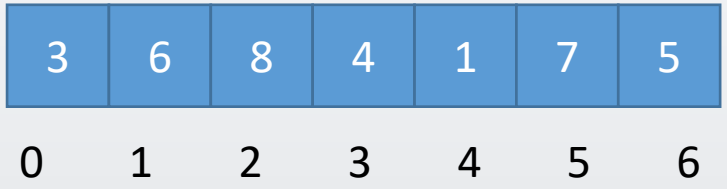


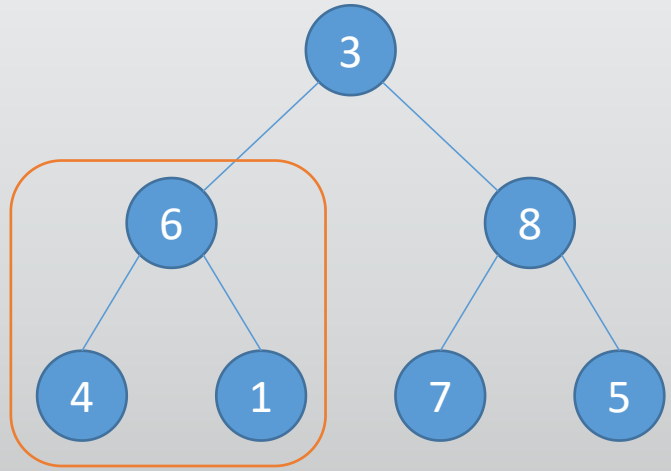
$array$ [3,4,5,6,1,7,8]





$array$ [3,4,5,6,1,7,8]

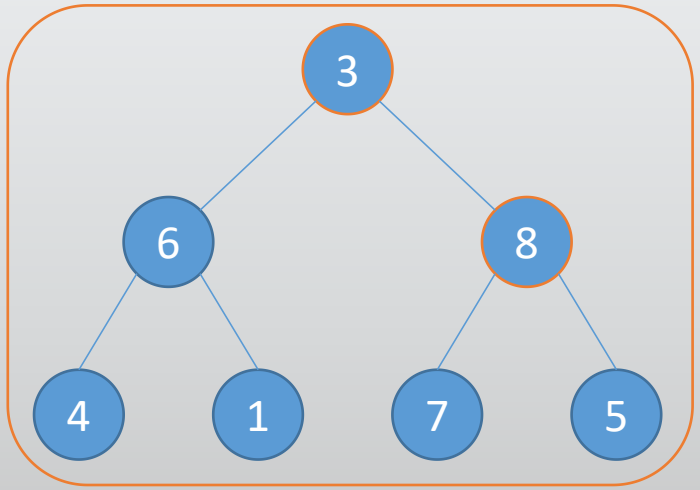




Build Max-heap bottom up

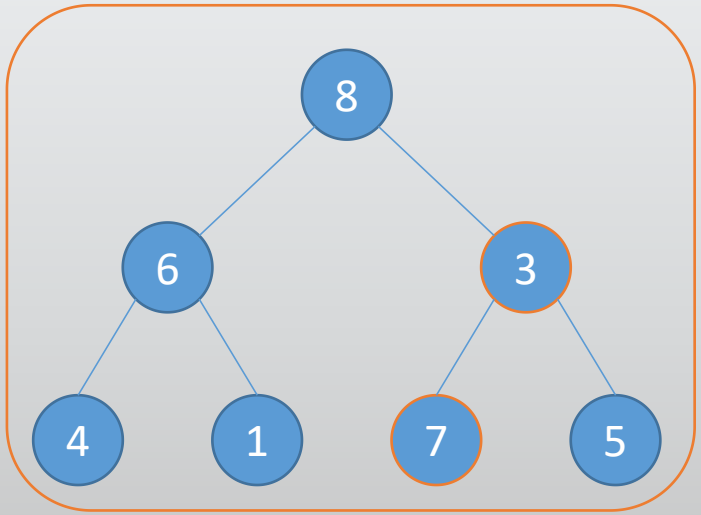
$array$ [3,4,5,6,1,7,8]

3	6	8	4	1	7	5
0	1	2	3	4	5	6



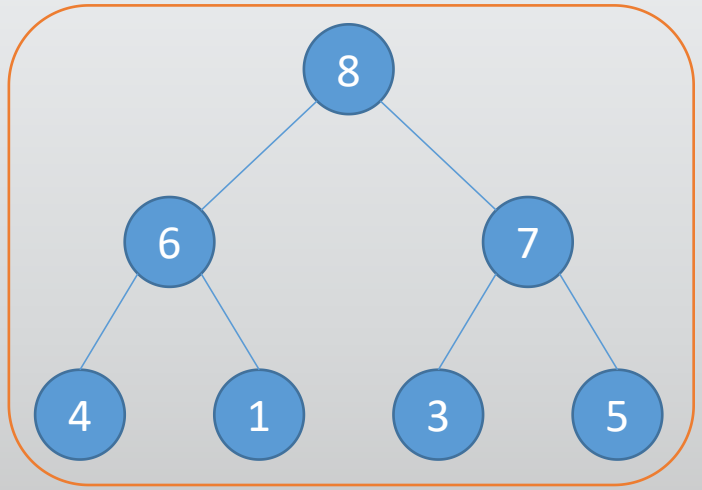
$array$ [3,4,5,6,1,7,8]

8	6	3	4	1	7	5
0	1	2	3	4	5	6



$array$ [3,4,5,6,1,7,8]

8	6	7	4	1	3	5
0	1	2	3	4	5	6



O(N)

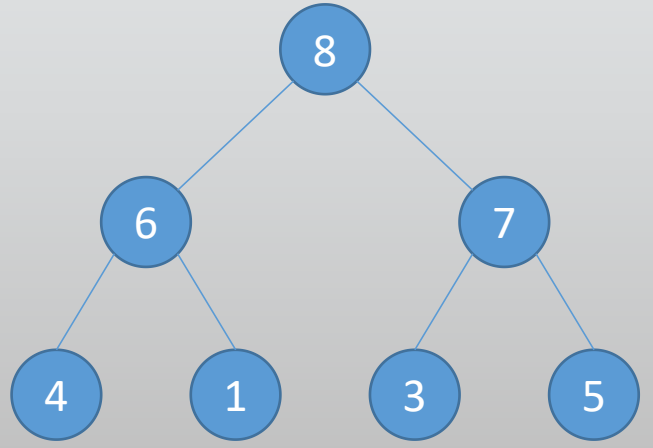
Application: priority queue

- Use Max/Min heap
- Two operation: push and pop
  - push: put element to queue's tail, then percolat up
  - pop: pop the queue's head(root), put the rightest leaf of lowest level, then percolate down

- Example:

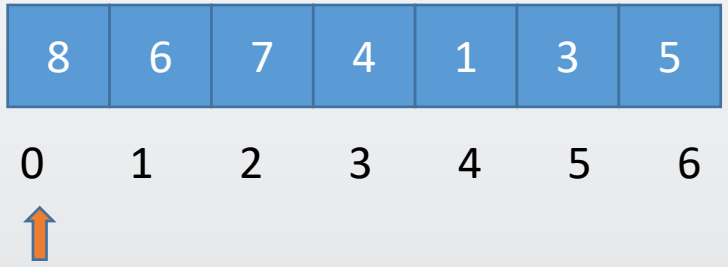
Max-heap to descending array

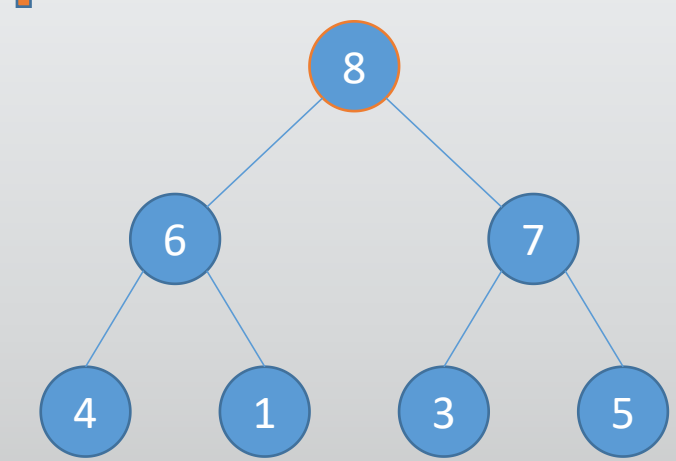
8	6	7	4	1	3	5
0	1	2	3	4	5	6



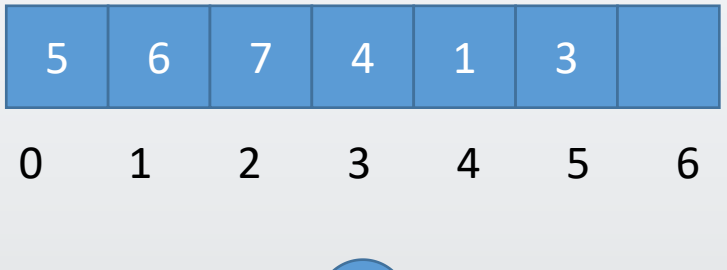
Example: priority queue

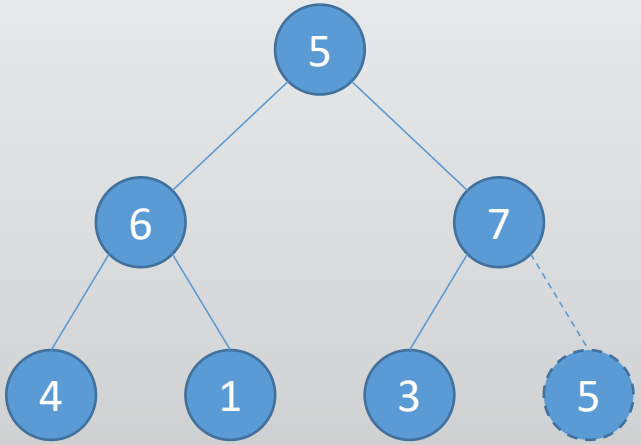
$array\ [8]$



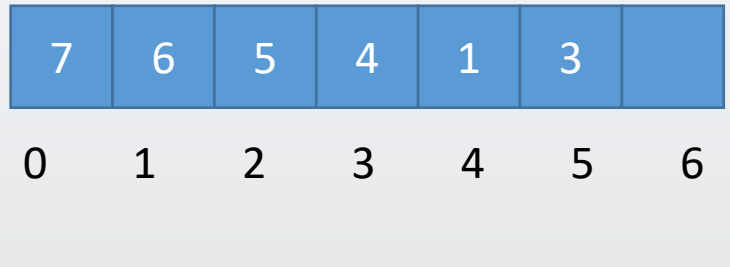


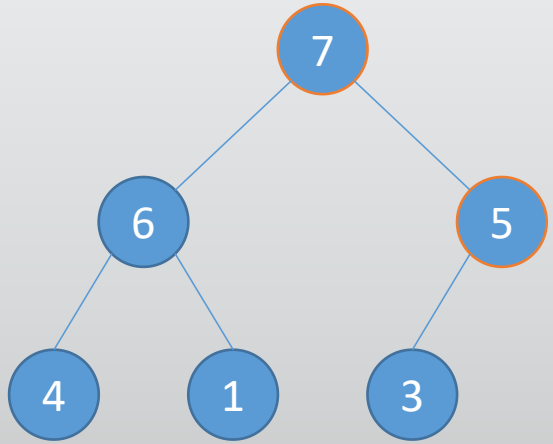
$array\ [8]$





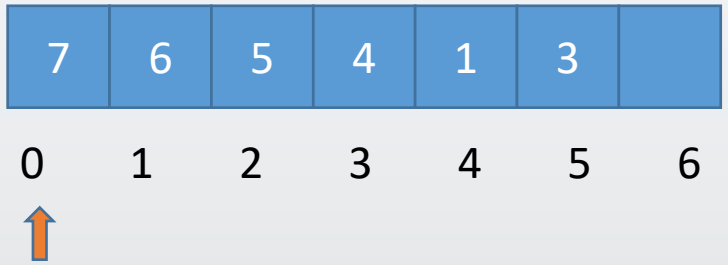
$array\ [8]$

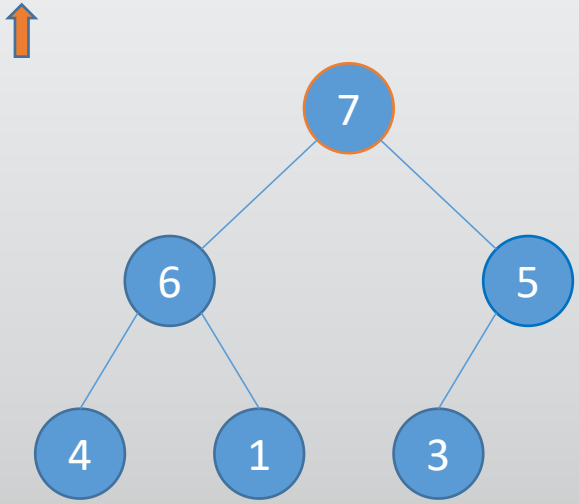




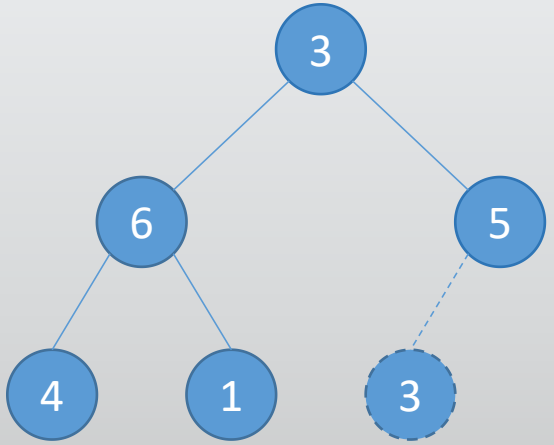
Example: priority queue

$array$ [8,7]

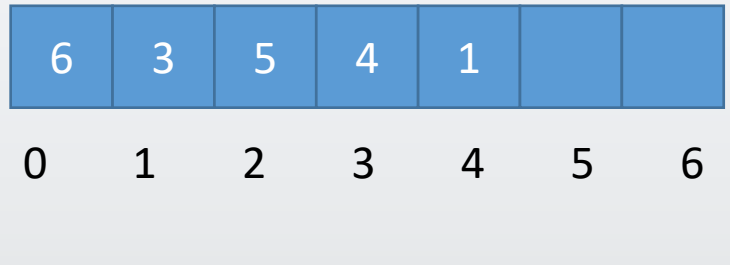


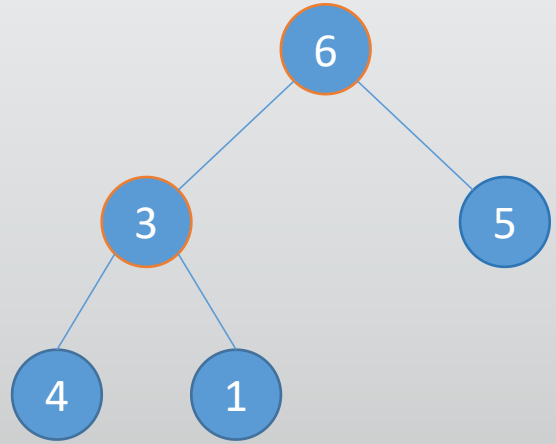


$array$ [8,7]



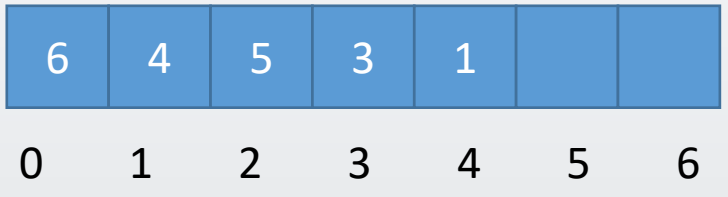
$array$ [8,7]

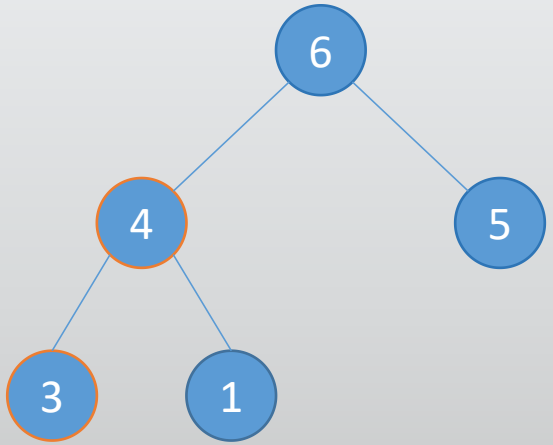




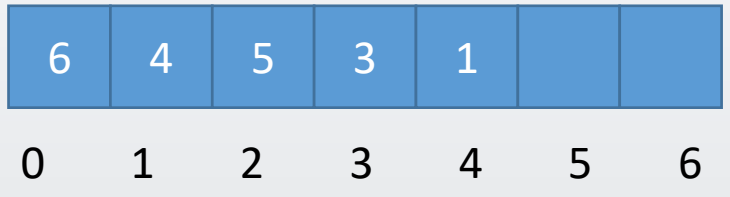
Example: priority queue

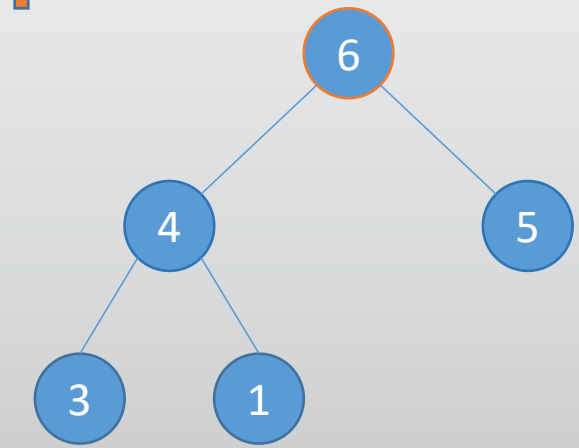
$array$ [8,7]



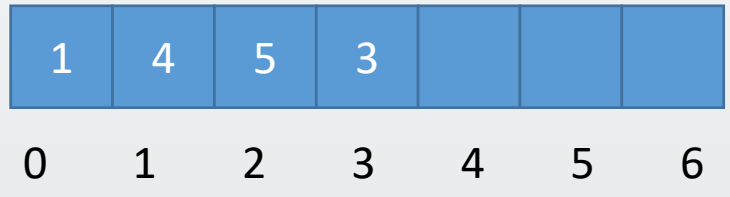


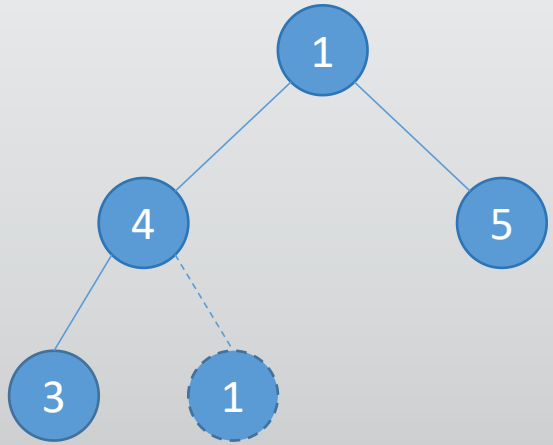
$array$ [8,7,6]





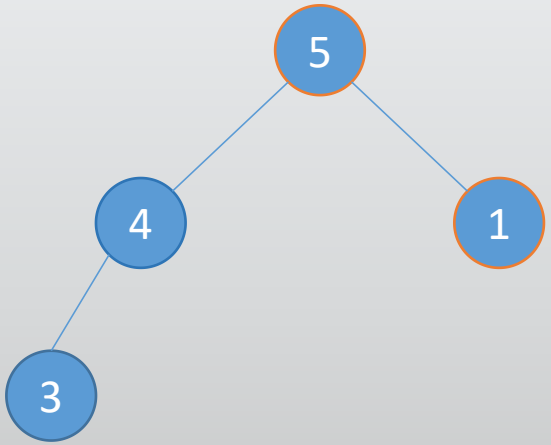
$array$ [8,7,6]



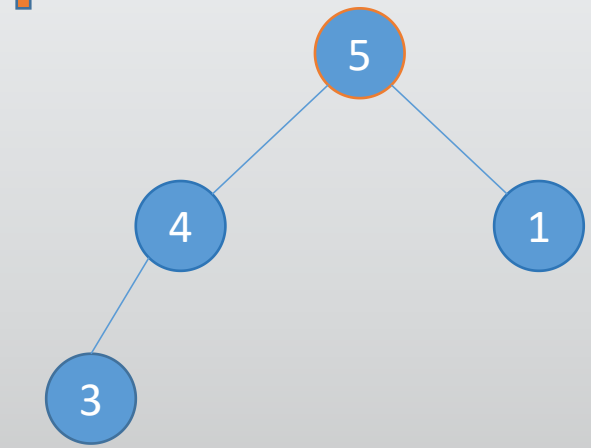


Example: priority queue

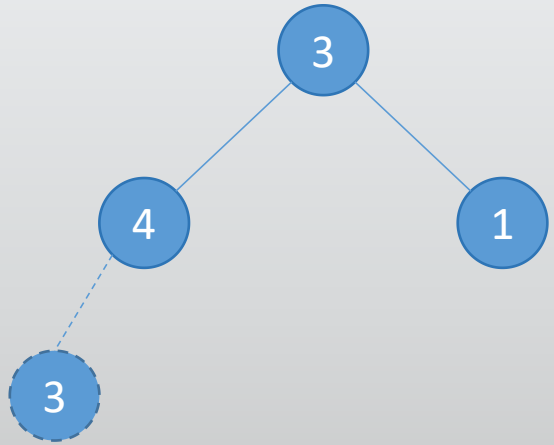
$array$ [8,7,6]



$array$ [8,7,6,5]

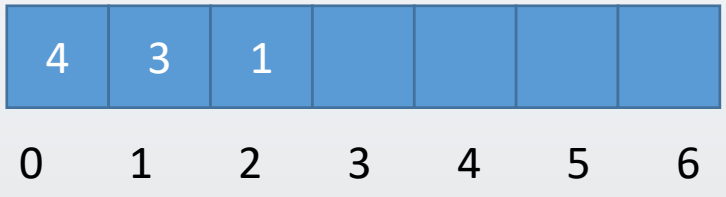


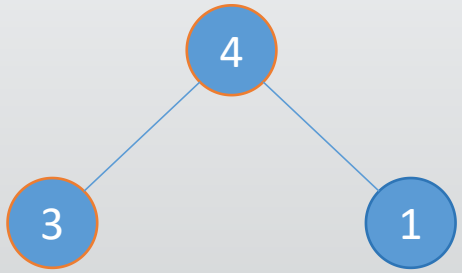
$array$ [8,7,6,5]



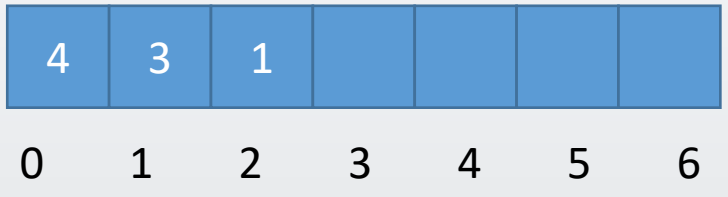
Example: priority queue

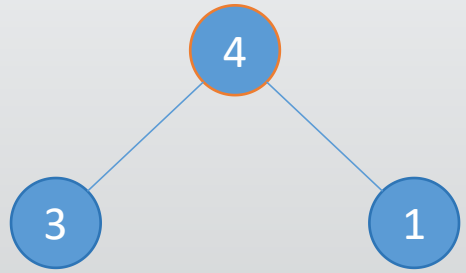
$array$ [8,7,6,5]



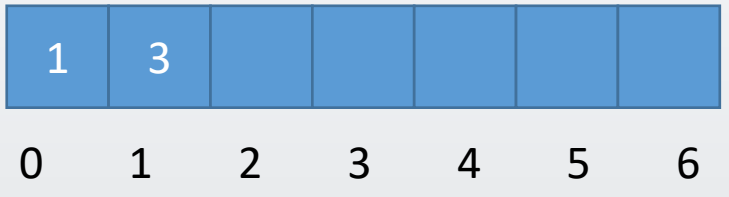


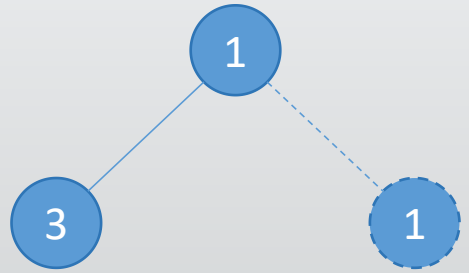
$array$ [8,7,6,5,4]





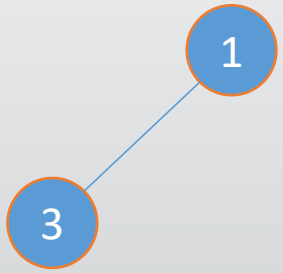
$array$ [8,7,6,5,4]



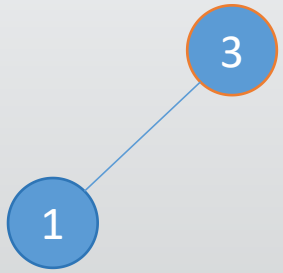


Example: priority queue

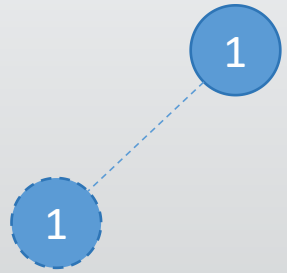
$array$ [8,7,6,5,4]



$array$ [8,7,6,5,4,3]

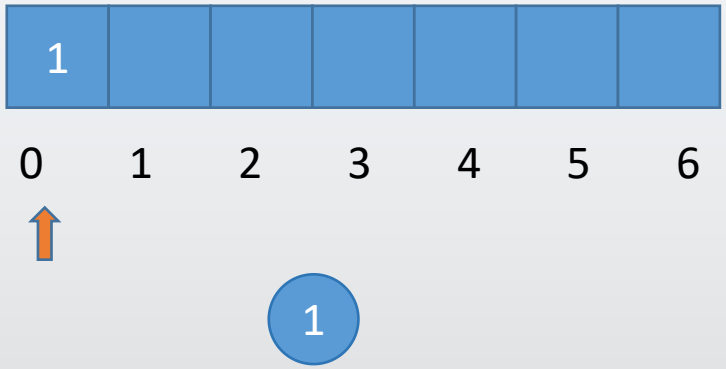


$array$ [8,7,6,5,4,3]



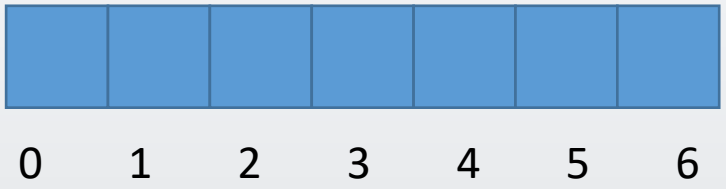
Example: priority queue

$array$ [8,7,6,5,4,3,1]



1

$array$ [8,7,6,5,4,3,1]



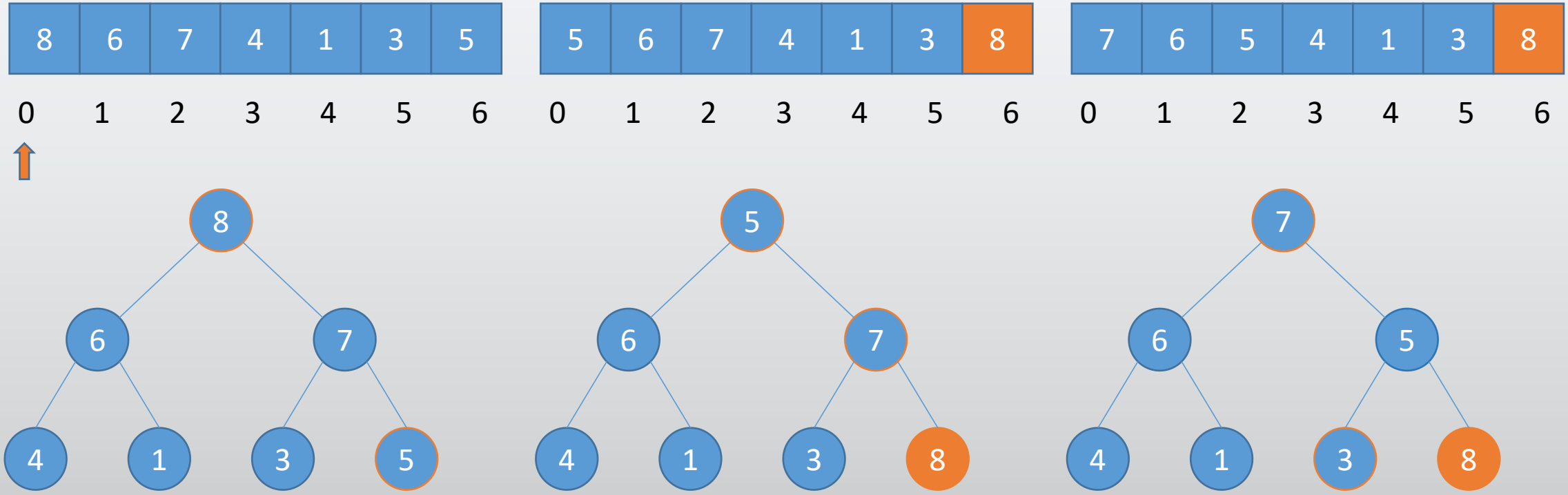
O(NlogN)

Heap sort

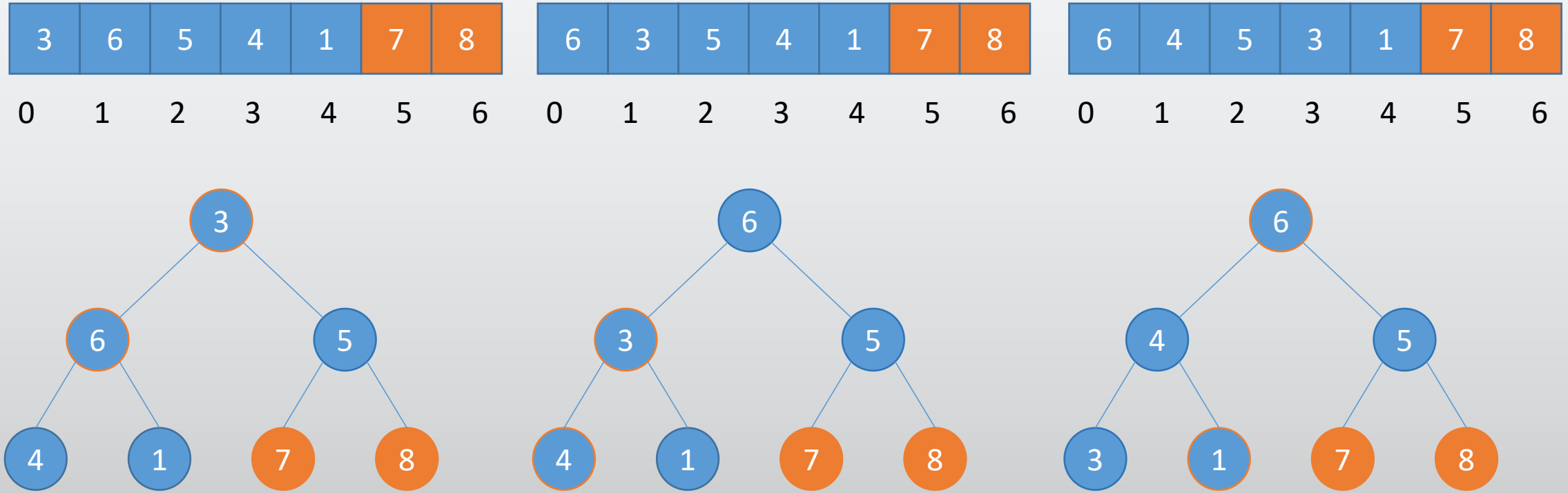
- Lower space complexity than above algrithm
- Same time complexity
- Use heap to store the data

- Example: use max-heap

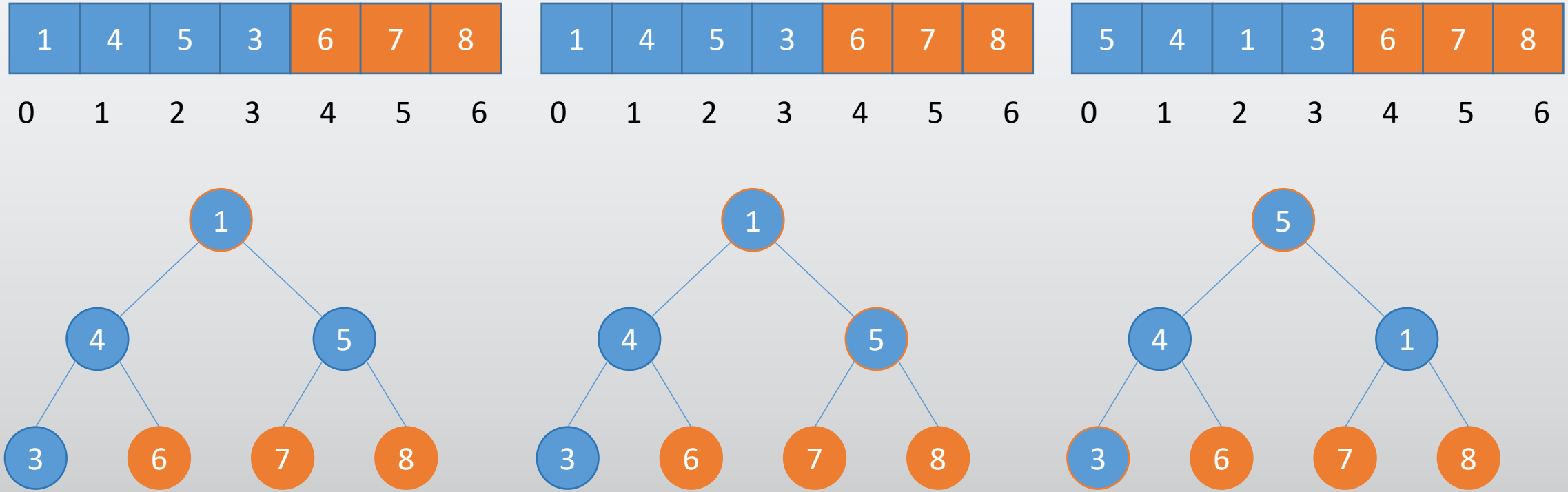
Example: heap sort



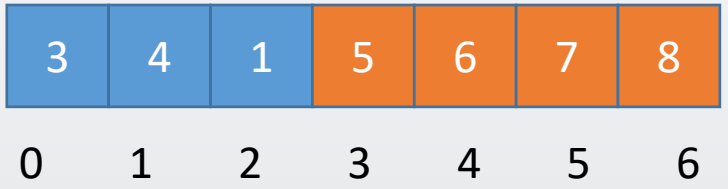
Example: heap sort

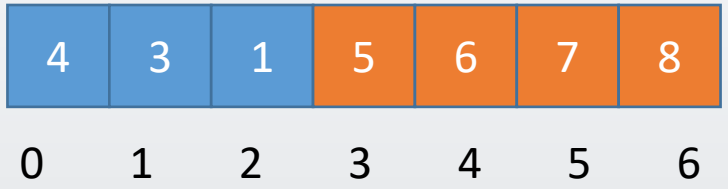


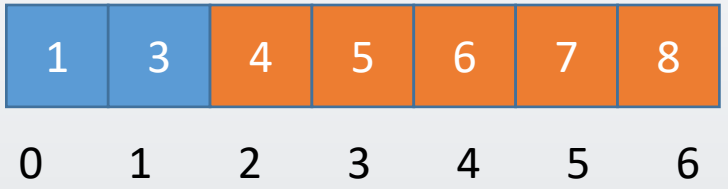
Example: heap sort

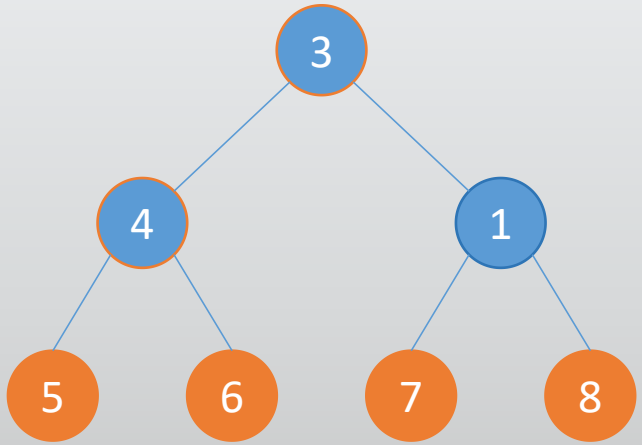


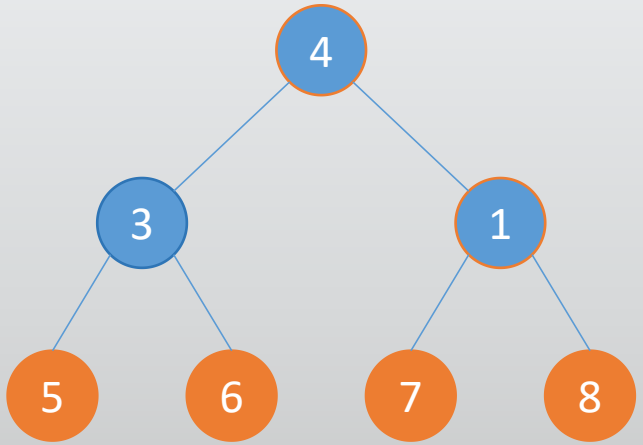
Example: heap sort

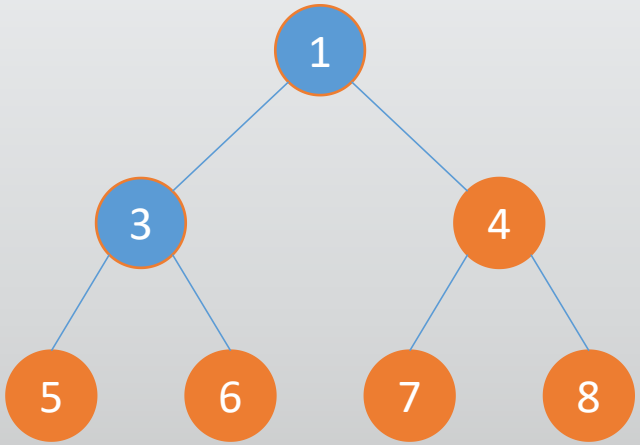




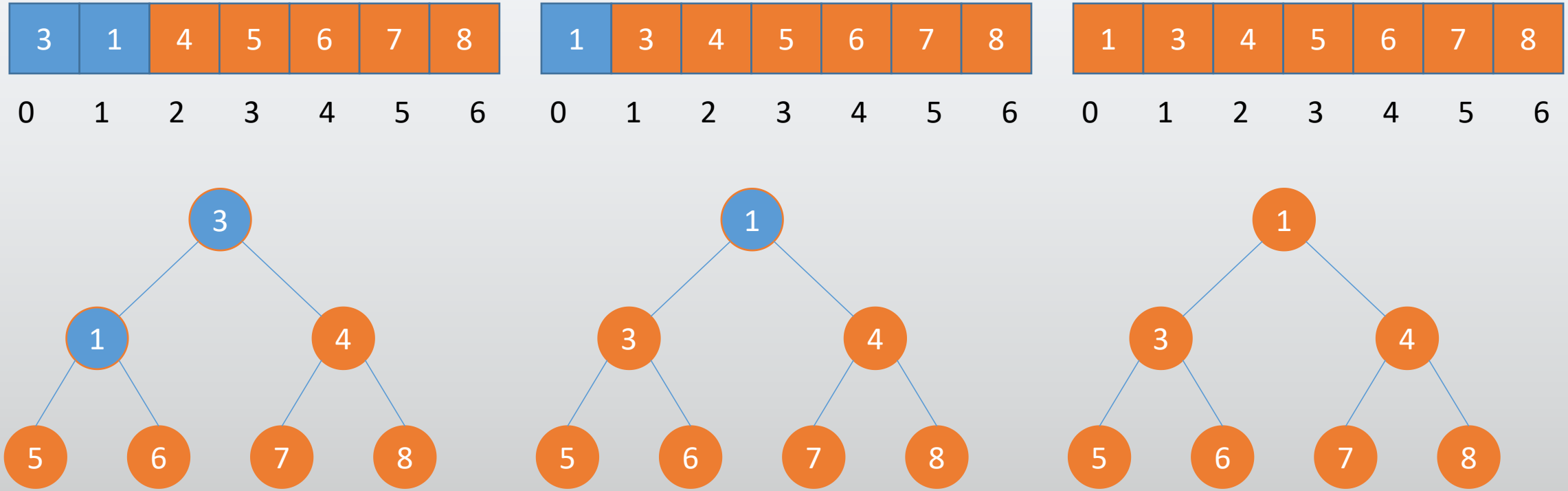








Example: heap sort



O(NlogN)

Thx :-)