BİL3014

Algoritma Analizi

Dr. Öğr. Üyesi Emre DELİBAŞ

Heap ve Heap Sort

Heapsort

- Heap, a data structure
- Max-Heapify procedure
- Building a max-heap
- Heapsort

Quiz Sample

- Is array a data structure?

Is array a data structure?

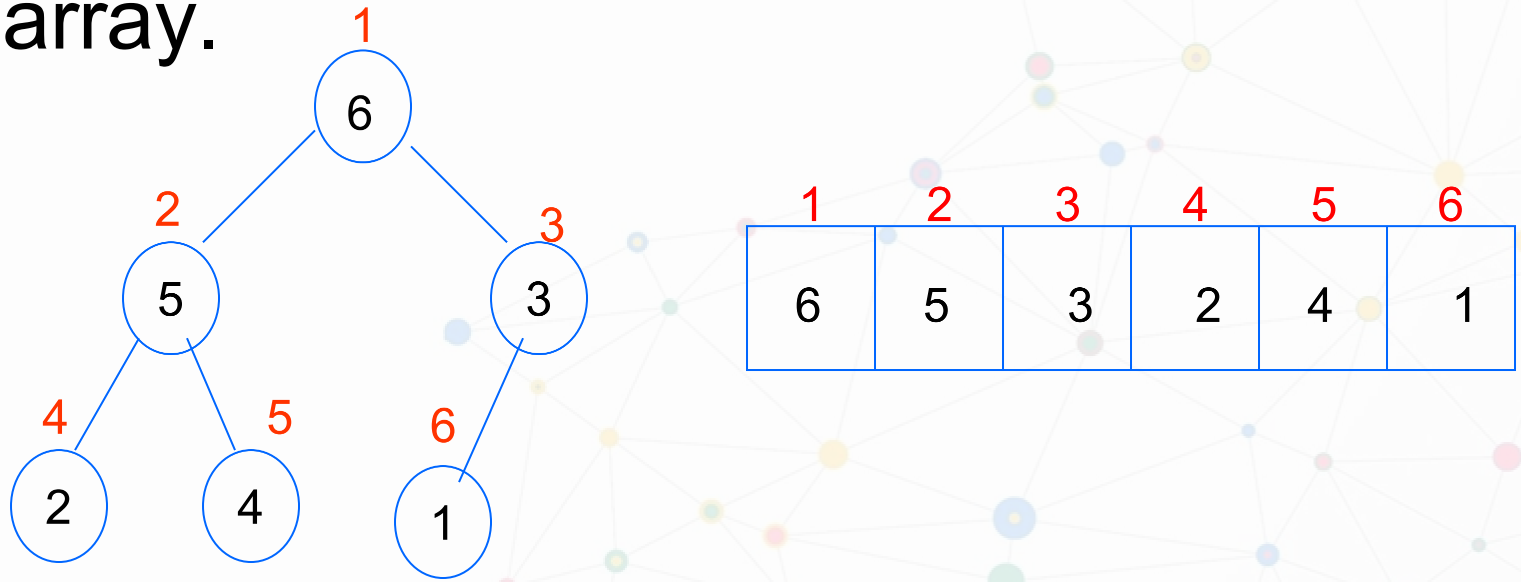
- No!
- A data structure is a data organization associated with a set of operations (standard algorithms) for efficiently using the data.
- What data structures do you know on array?

Is array a data structure?

- No!
- A data structure is a data organization associated with a set of operations (standard algorithms) for efficiently using the data.
- What data structures do you know on array?
- Stack, queue, list, …, heap.

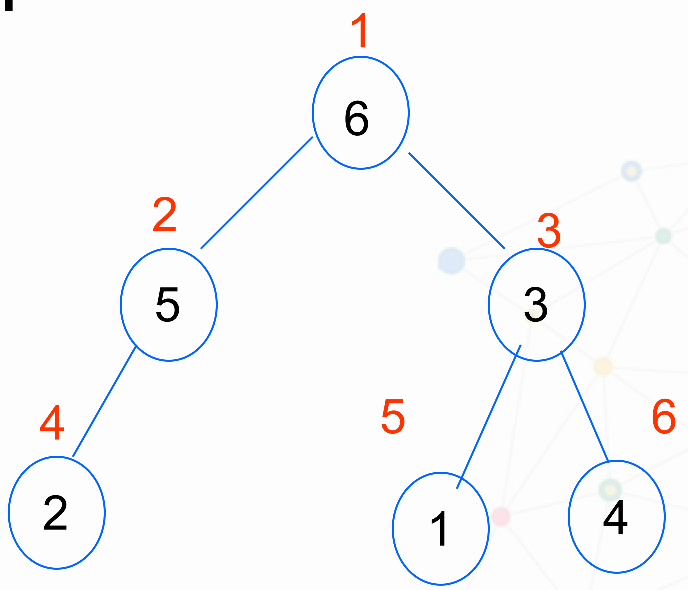
A Data Structure Heap

- A heap is a nearly complete binary tree which can be easily implemented on an array.

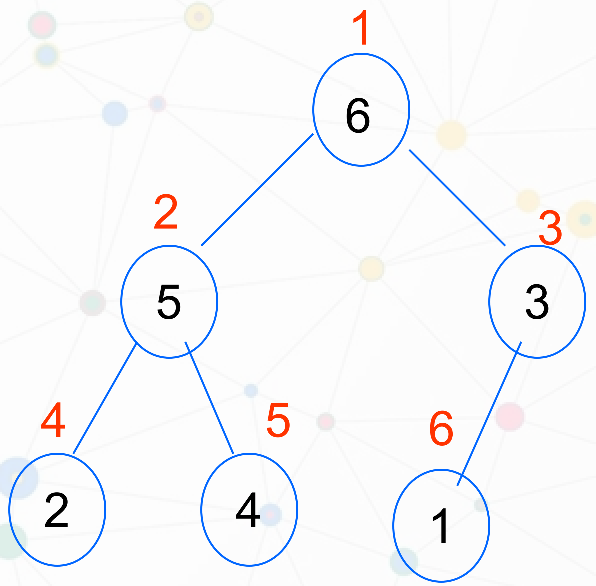


Nearly complete binary tree

- Every level except bottom is complete.
- On the bottom, nodes are placed as left as possible.



No!



Yes!

Algorithms associated with Heap

$$\text{Parent}(i)$$
$$\quad \text{return } \lfloor i/2 \rfloor;$$

$$\text{Left}(i)$$
$$\quad \text{return } 2i;$$

$$\text{Right}(i)$$
$$\quad \text{return } 2i+1;$$

Two Special Heaps

- Max-Heap
- Min-Heap

Max-Heap

In a max-heap, every node i other than the root satisfies the following property:

$$A[\text{Parent}(i)] \geq A[i].$$

Min-Heap

In a min-heap, every node i other than the root satisfies the following property:

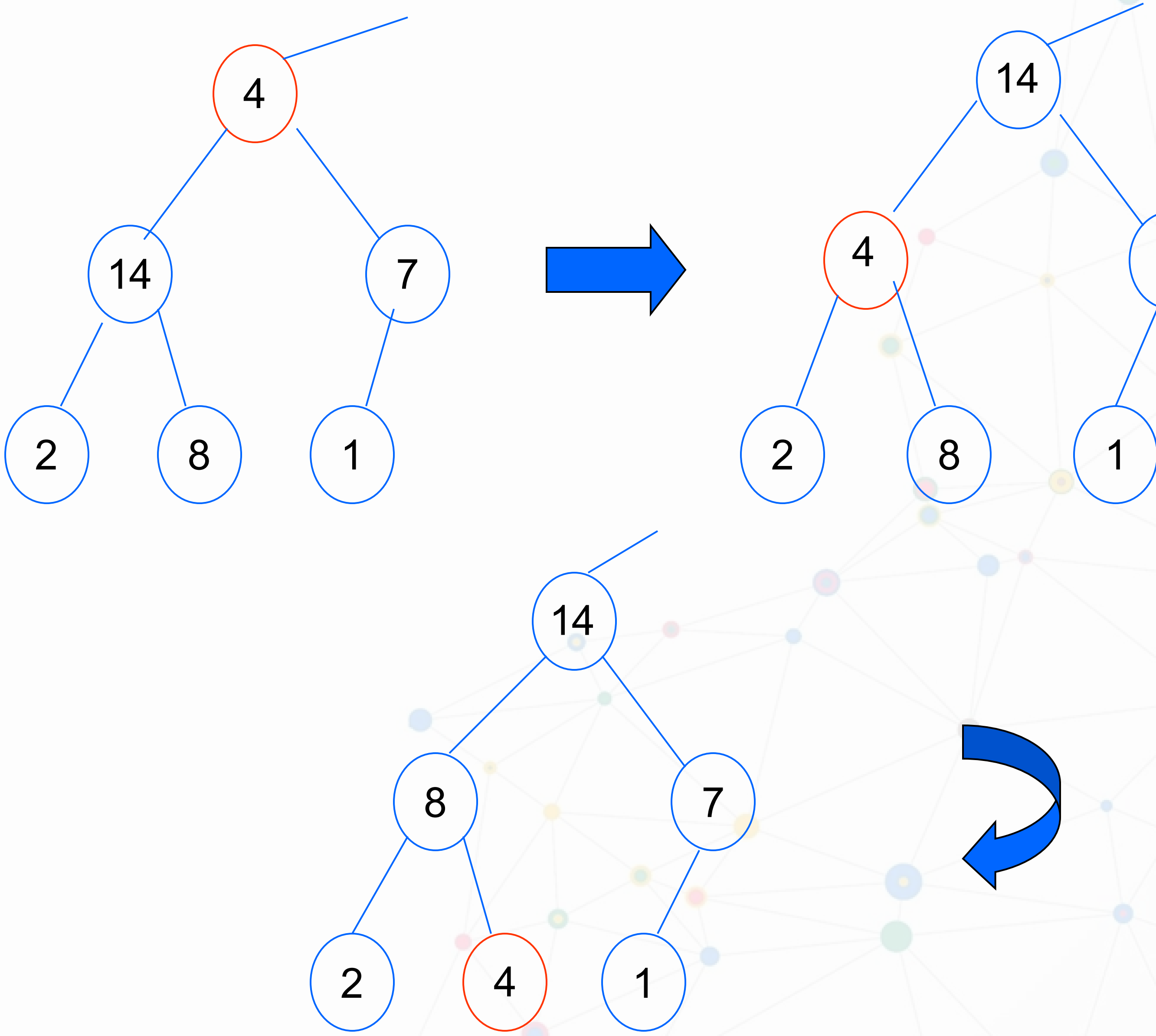
$$A[\text{Parent}(i)] \leq A[i].$$

Algorithms associated with Max-Heap

- Three algorithms associated with heap.
- In addition, it is associated with two more algorithms: Max-Heapify and Build-Max-Heap.

Max-Heapify

- Max-Heapify(A,i) is a subroutine.
- When it is called, two subtrees rooted at Left(i) and Right(i) are max-heaps, but A[i] may not satisfy the max-heap property.
- Max-Heapify(A,i) makes the subtree rooted at A[i] become a max-heap by letting A[i] “float down”.


4
14
7
2
8
1
14
4
7
2
8
1
14
8
7
2
4
1

$$\text{Max-Heapify}\,(A, i)$$

$$l \leftarrow \text{Left}\,(i);$$

$$r \leftarrow \text{Right}\,(i);$$

$$\text{if } l \leq heap-size[A] \text{ and } A[l] > A[i]$$

$$\quad \text{then } largest \leftarrow l$$

$$\quad \text{else } largest \leftarrow i;$$

$$\text{if } r \leq heap-size[A] \text{ and } A[r] > A[largest]$$

$$\quad \text{then } largest \leftarrow r;$$

$$\text{if } largest \neq i$$

$$\quad \text{then begin exchange } A[i] \leftrightarrow A[largest];$$

$$\qquad \text{Max-Heapify}\,(A, largest);$$

$$\quad \text{end-if}$$

Running time

$$height = \lfloor \lg n \rfloor \text{ where } n = heap\text{-}size[A]$$

Hence, Max-Heapify runs in time $O(\lg n)$.

Let $h = height$. Then h is the largest integer

satisfying $n \geq 1 + 2 + \cdots + 2^{h-1} + 1$,

that is, $n \geq 2^h$.

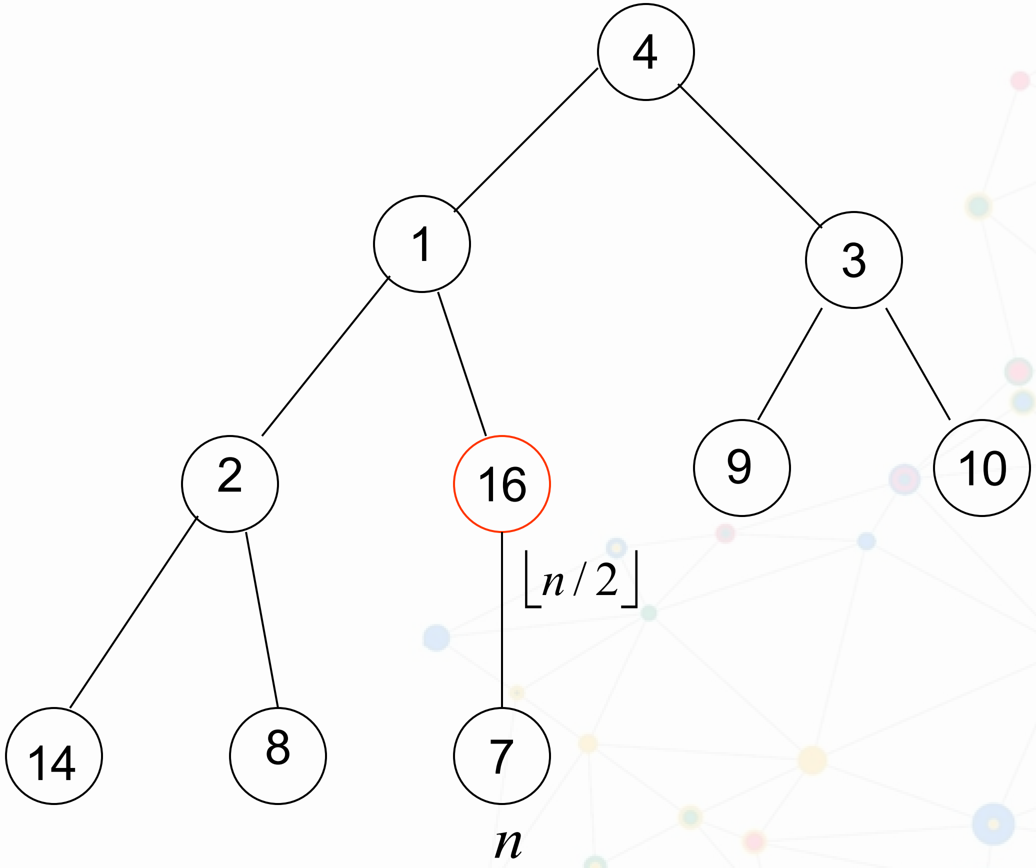
Hence, $h = \lfloor \lg n \rfloor$.

Building a Max-Heap

$$\text{Build-Max-Heap}(A)$$
$$\textit{heap-size}[A] \leftarrow \textit{length}[A];$$
$$\text{for } i \leftarrow \lfloor \textit{length}[A]/2 \rfloor \text{ downto } 1$$
$$\quad \text{do Max-Heapify}(A, i);$$

e.g., 4, 1, 3, 2, 16, 9, 10, 14, 8, 7.

The last location who has a child is $\lfloor n/2 \rfloor$.

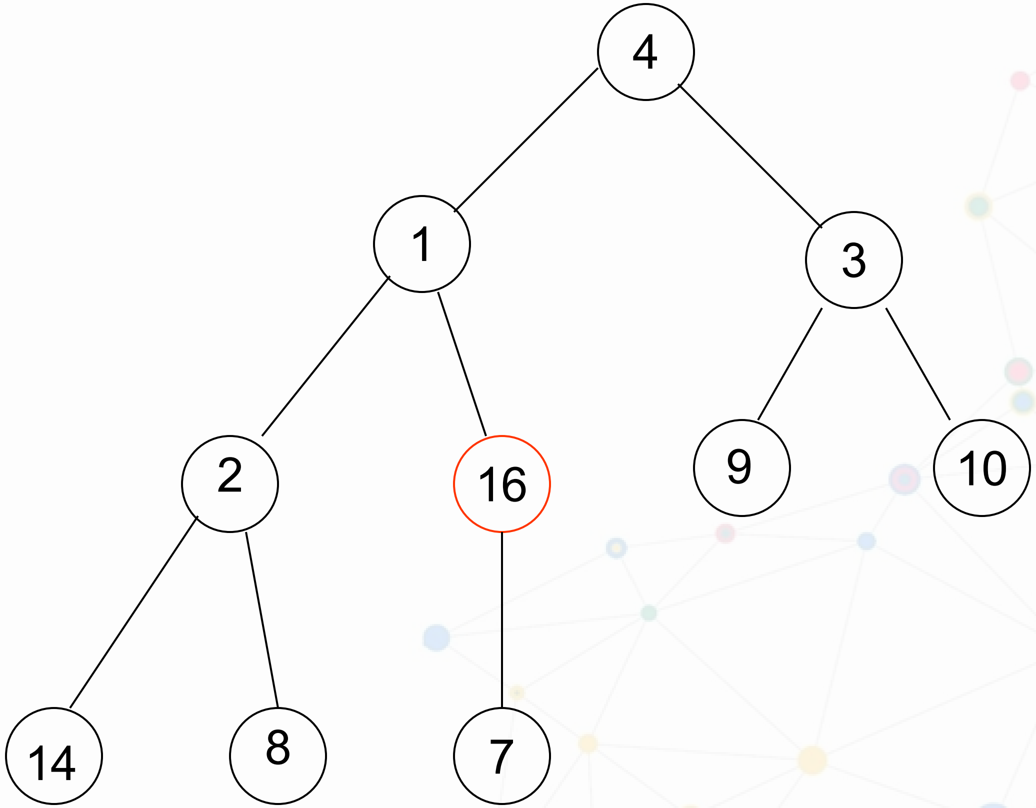


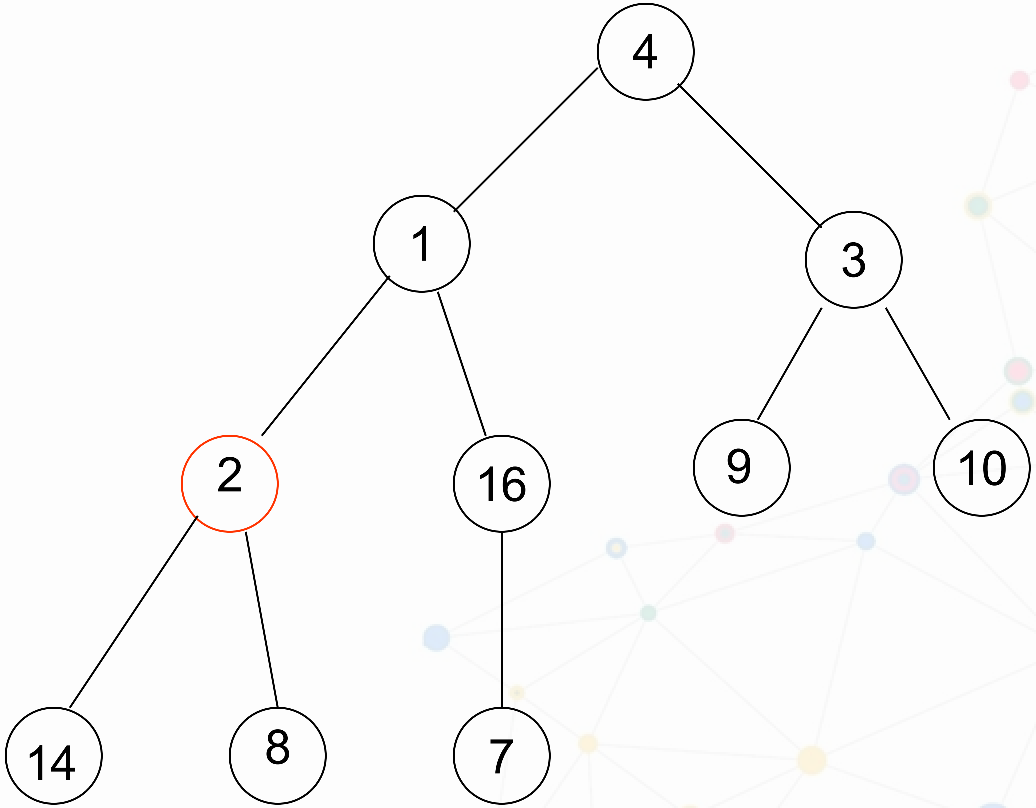
Proof. It is parent of location n

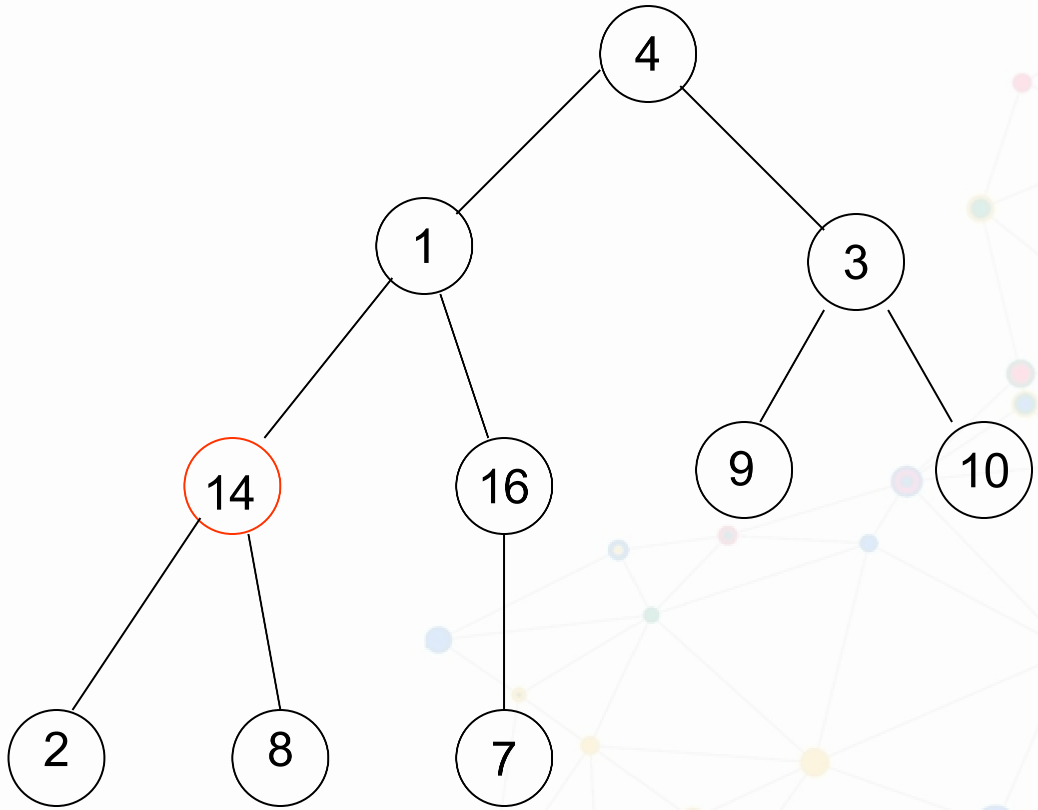
Building a Max-Heap

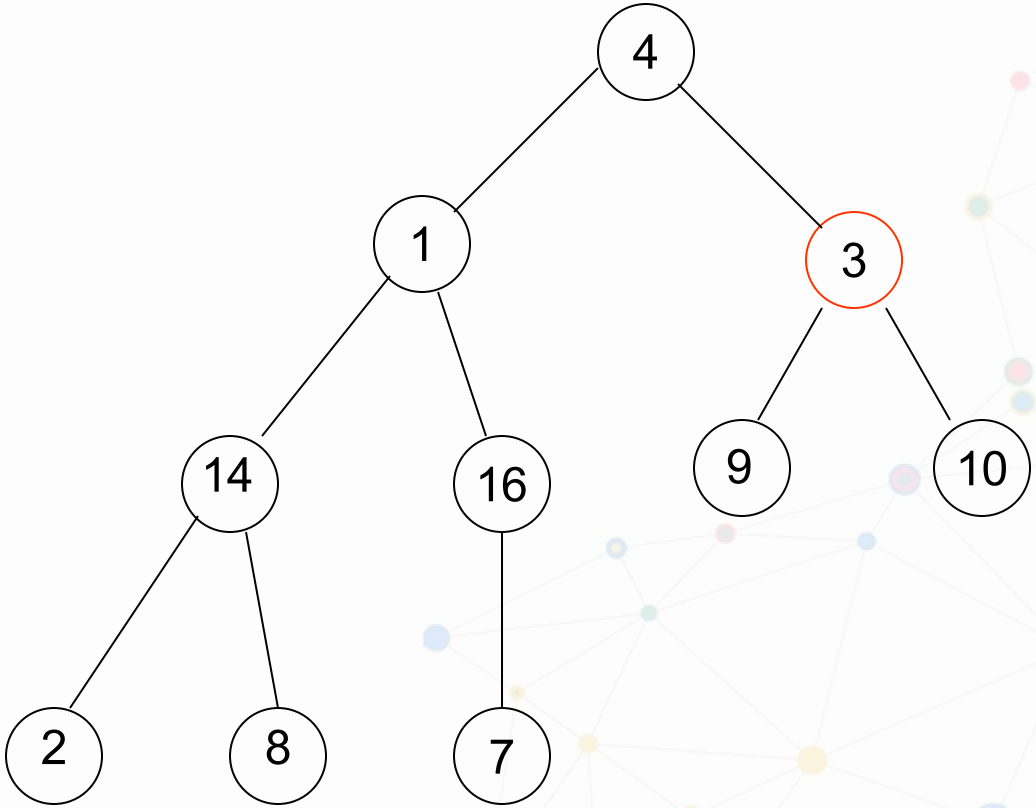
$$\text{Build - Max - Heap}(A)$$
$$heap\text{-}size[A] \leftarrow length[A];$$
$$\text{for } i \leftarrow \lfloor length[A]/2 \rfloor \text{ downto } 1$$
$$\quad \text{do Max - Heapify}(A, i);$$

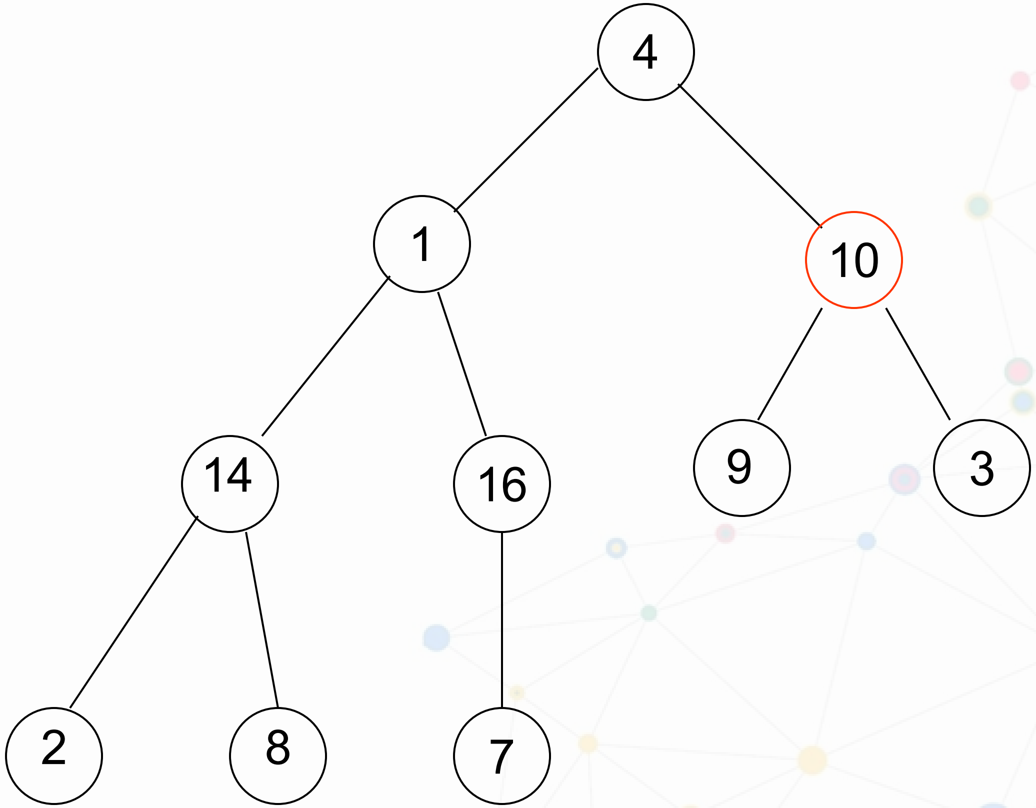
e.g., 4, 1, 3, 2, 16, 9, 10, 14, 8, 7.

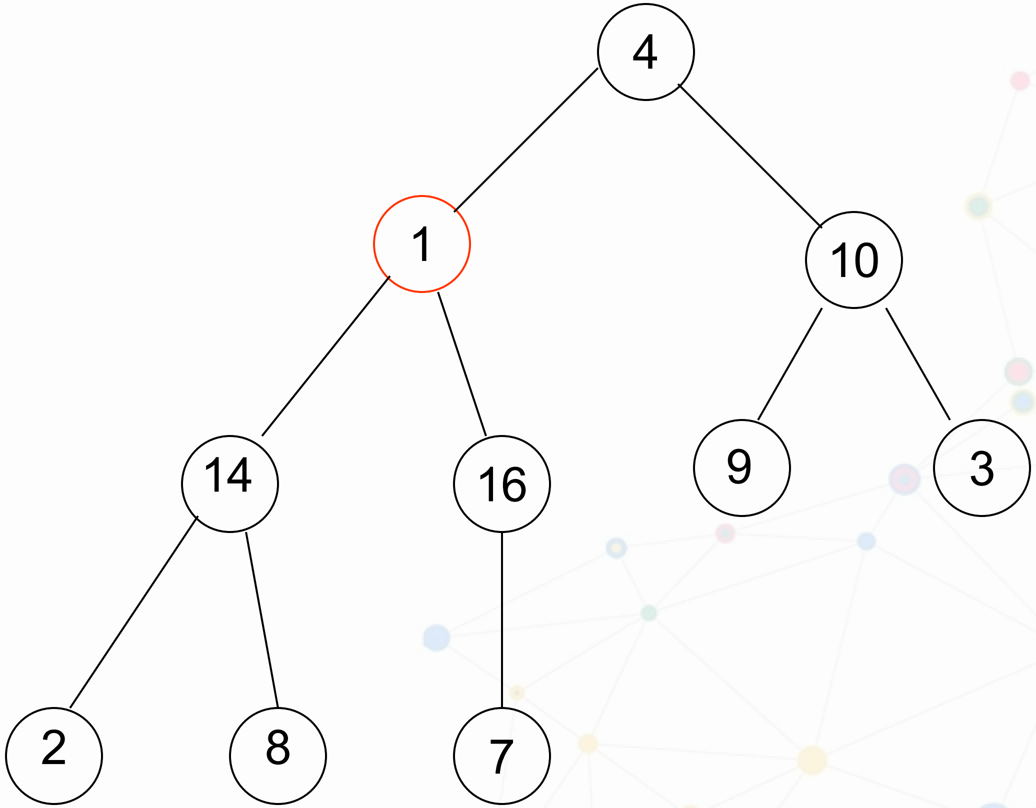

4
1
3
2
16
9
10
14
8
7

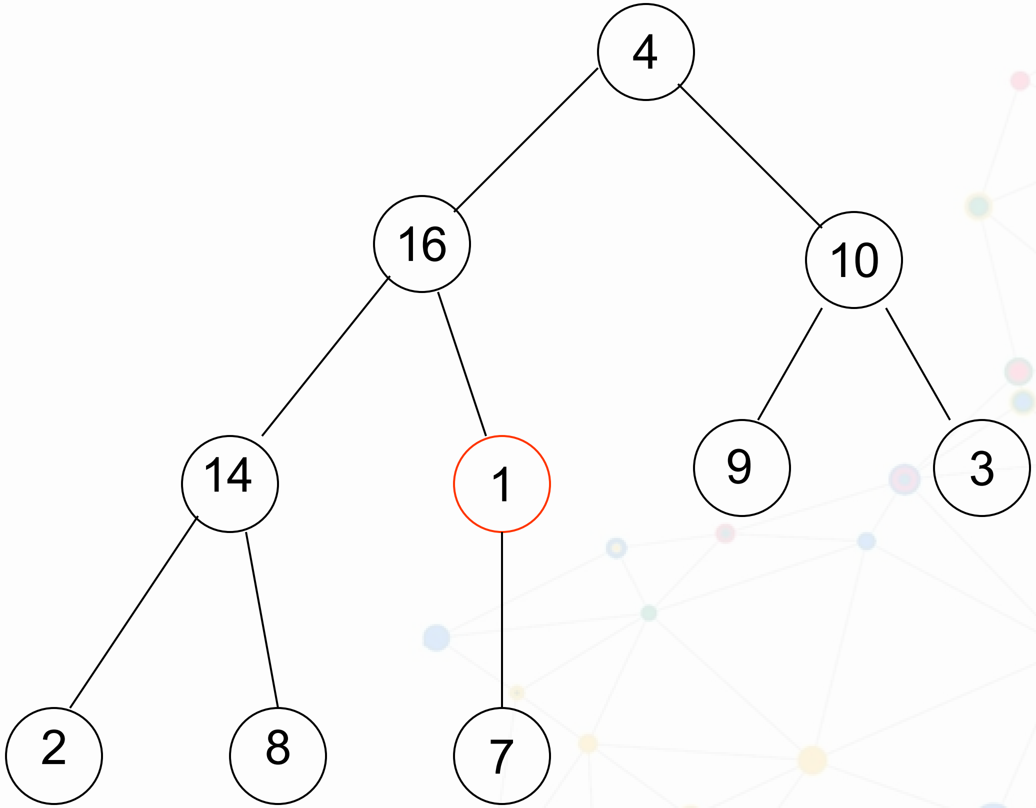

4
1
3
2
16
9
10
14
8
7


4
1
3
14
16
9
10
2
8
7

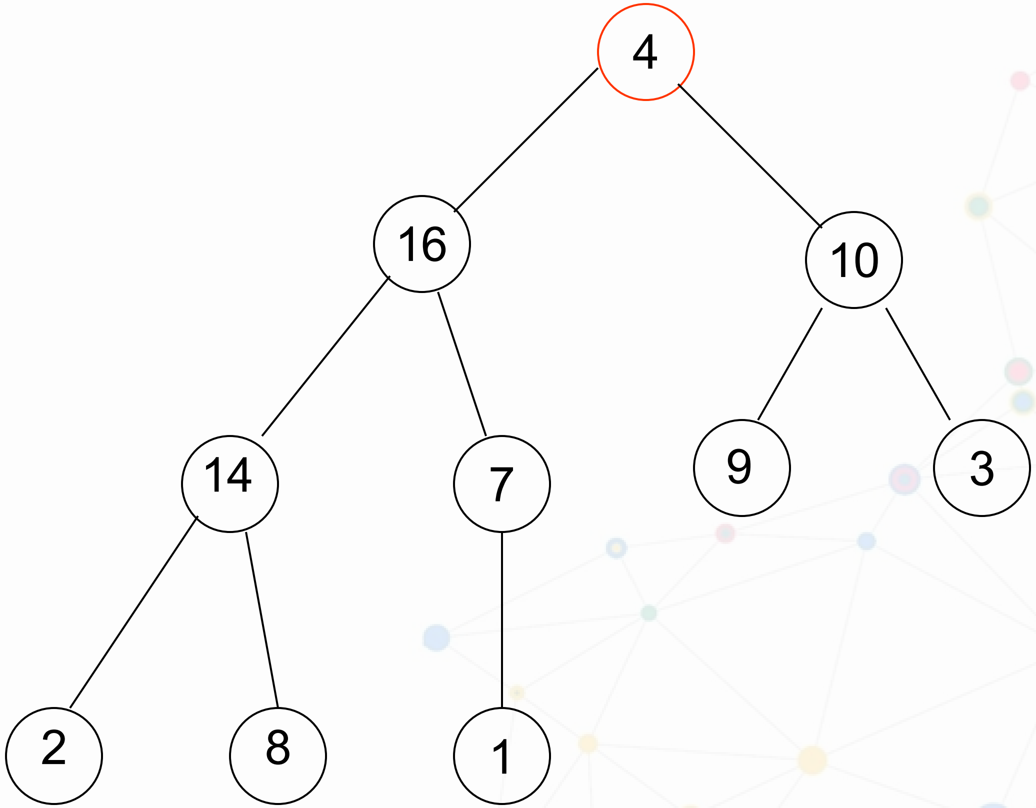

4
1
3
14
16
9
10
2
8
7

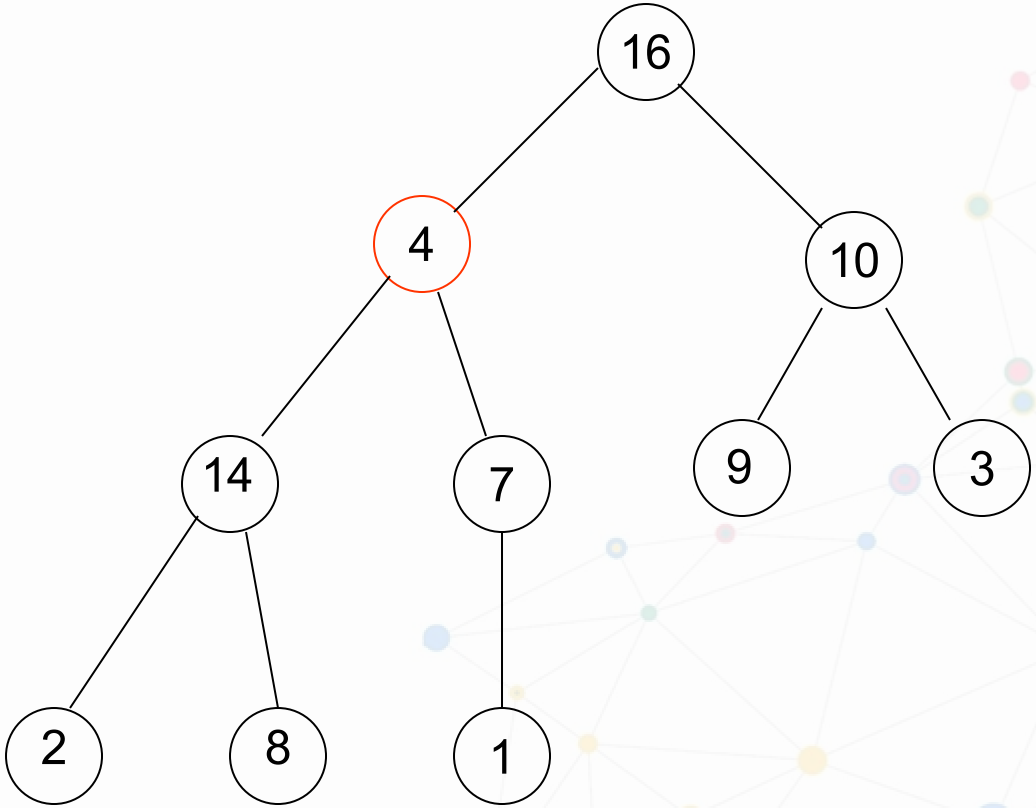

4
1
10
14
16
9
3
2
8
7

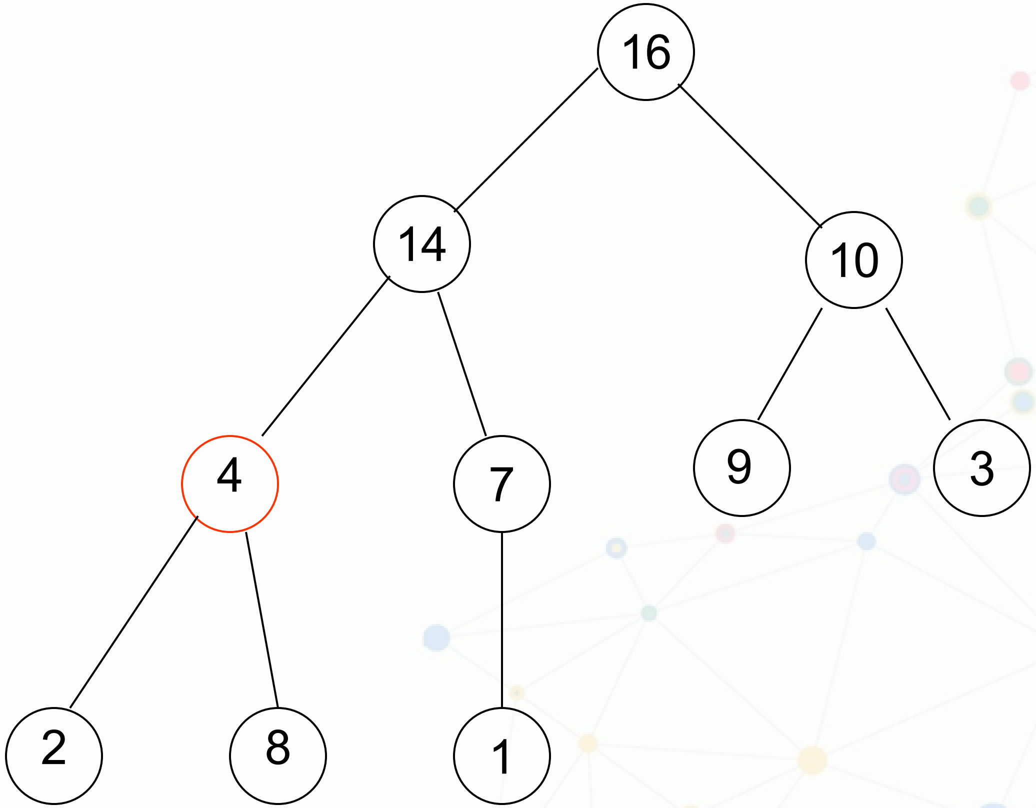

4
1
10
14
16
9
3
2
8
7



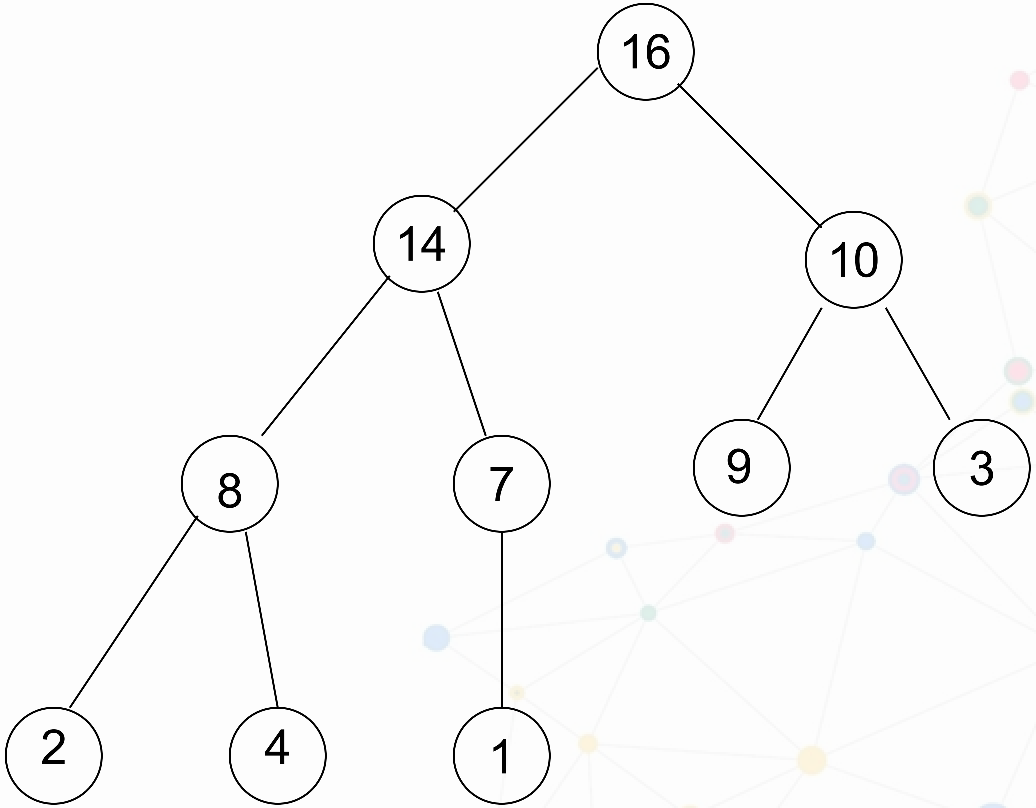
4
16
10
14
1
9
3
2
8
7


4
16
10
14
7
9
3
2
8
1


16
4
10
14
7
9
3
2
8
1



16
14
10
4
7
9
3
2
8
1


16
14
10
8
7
9
3
2
4
1

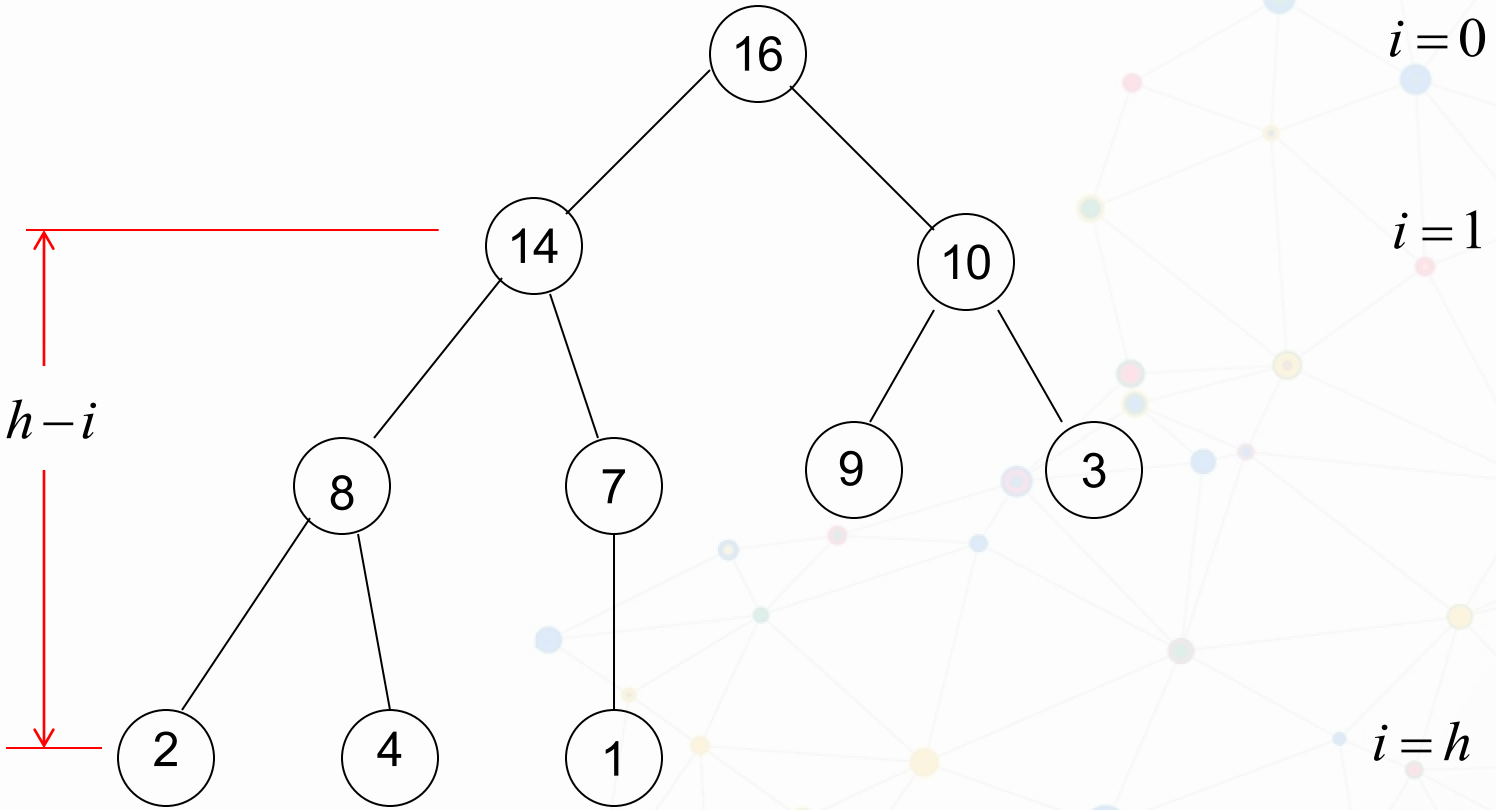
Analysis

$$\text{running time} = \sum_{i=0}^{h} \#node(i) \cdot O(h-i)$$

where $\#node(i)$ is the number of nodes at level i and $h = height(A)$.

$$h = \lfloor \lg n \rfloor \text{ where } n = heap\text{-}size(A)$$

$$\#node(i) \le 2^i$$



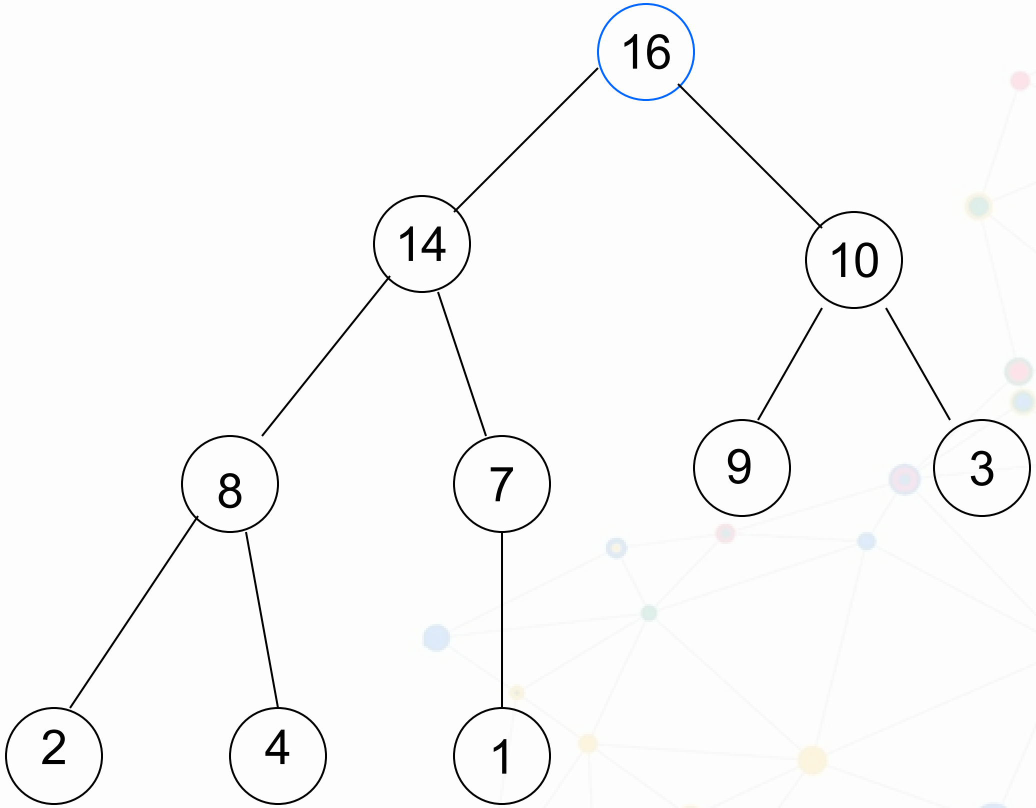
16
14
10
8
7
9
3
2
4
1
$h-i$
$i=0$
$i=1$
$i=h$

Heapsort

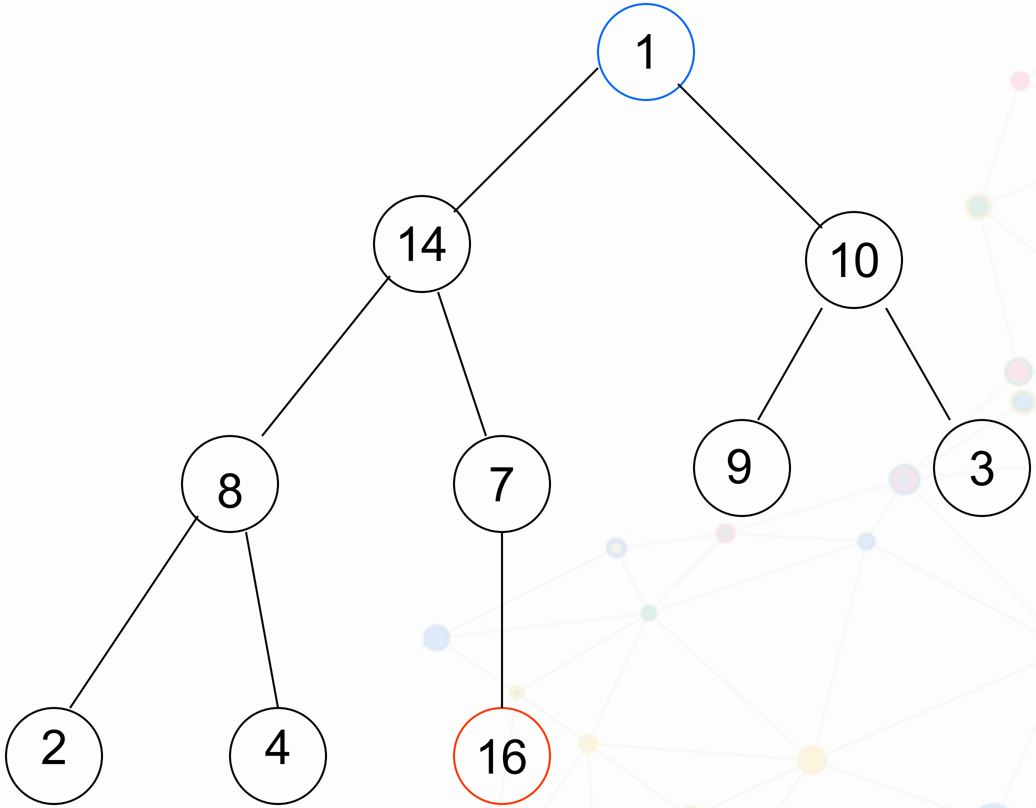
$$
\begin{array}{l}
\text{Heapsort}(A) \\
\quad \text{Buid - Max - Heap}(A); \\
\quad \text{for } i \leftarrow \mathit{length}[A] \text{ downto } 2 \\
\qquad \text{do begin} \\
\qquad\quad \text{exchange } A[1] \leftrightarrow A[i]; \\
\qquad\quad \mathit{heap}\text{-}\mathit{size}[A] \leftarrow \mathit{heap}\text{-}\mathit{size}[A] - 1; \\
\qquad\quad \text{Max - Heapify }(A, 1); \\
\qquad \text{end - for}
\end{array}
$$

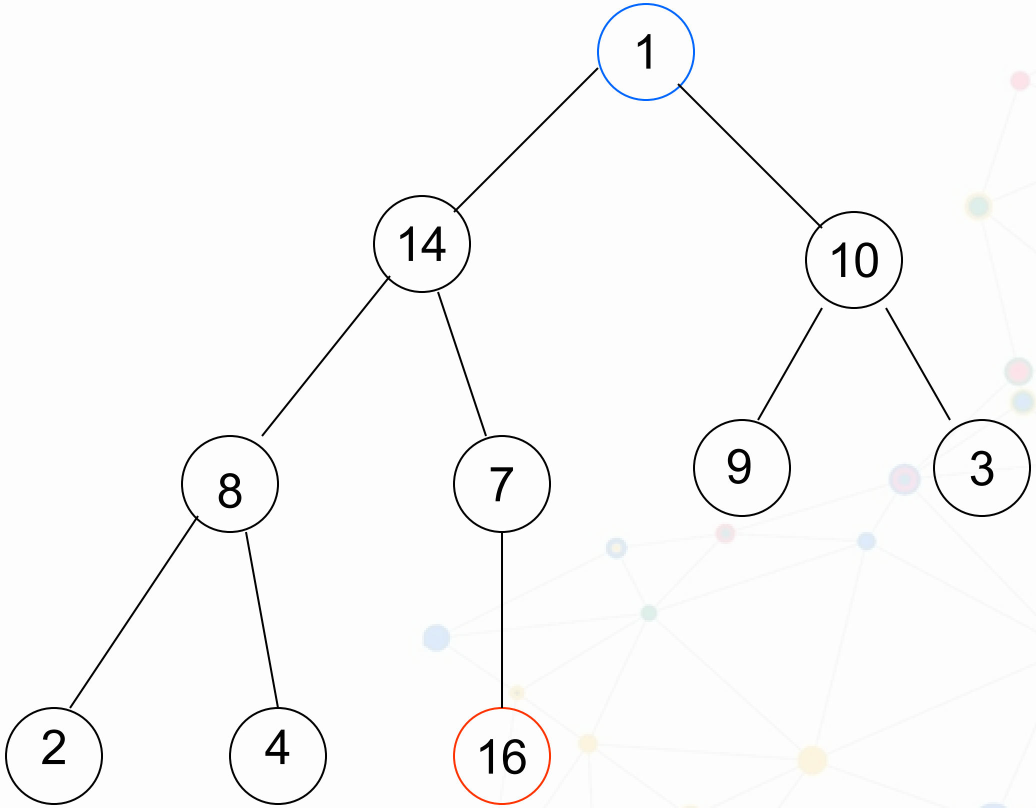
Input: 4, 1, 3, 2, 16, 9, 10, 14, 8, 7.

Build a max-heap

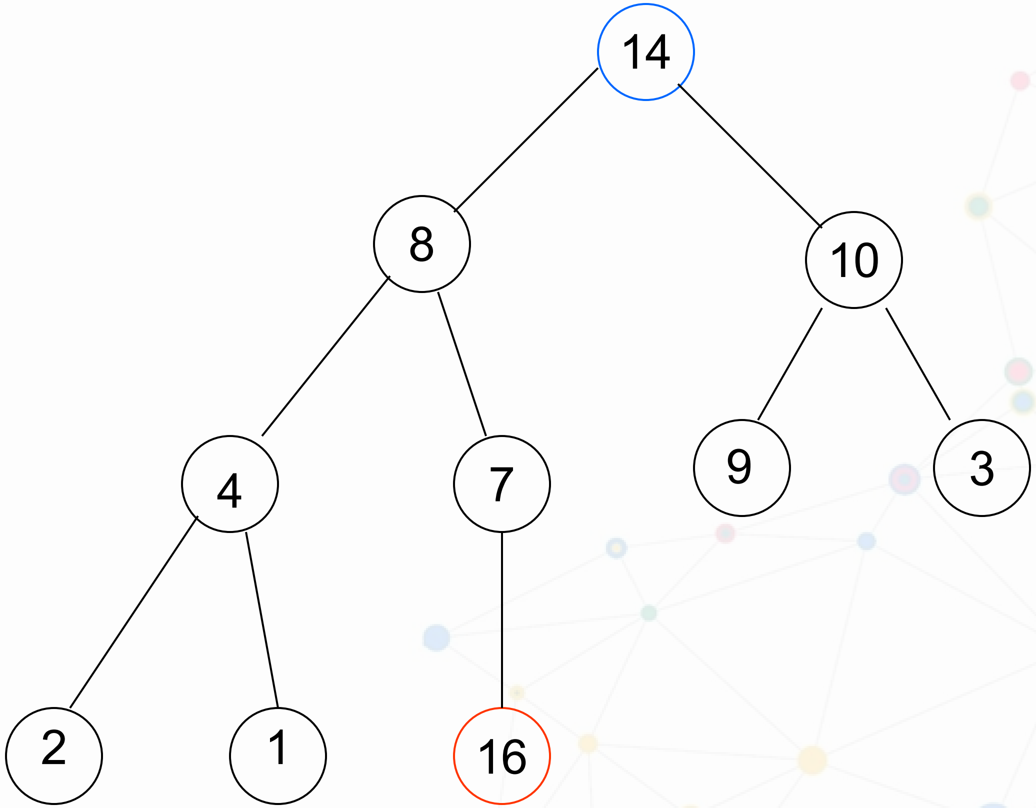


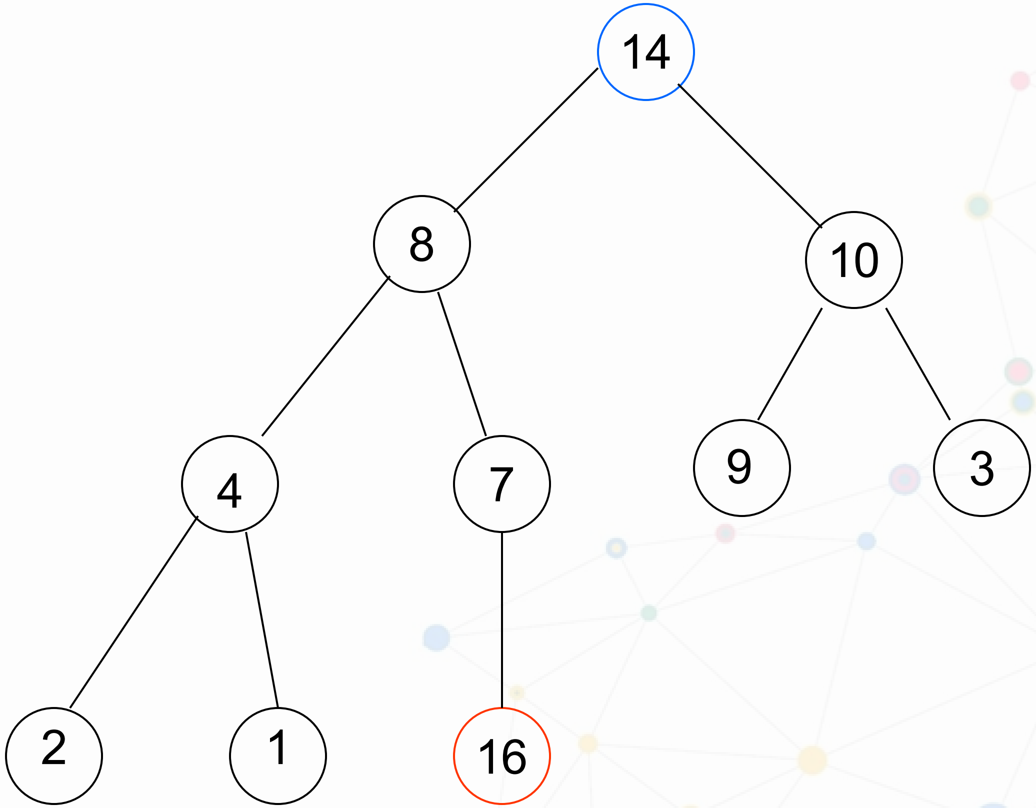
16, 14, 10, 8, 7, 9, 3, 2, 4, 1.


1
14
10
8
7
9
3
2
4
16

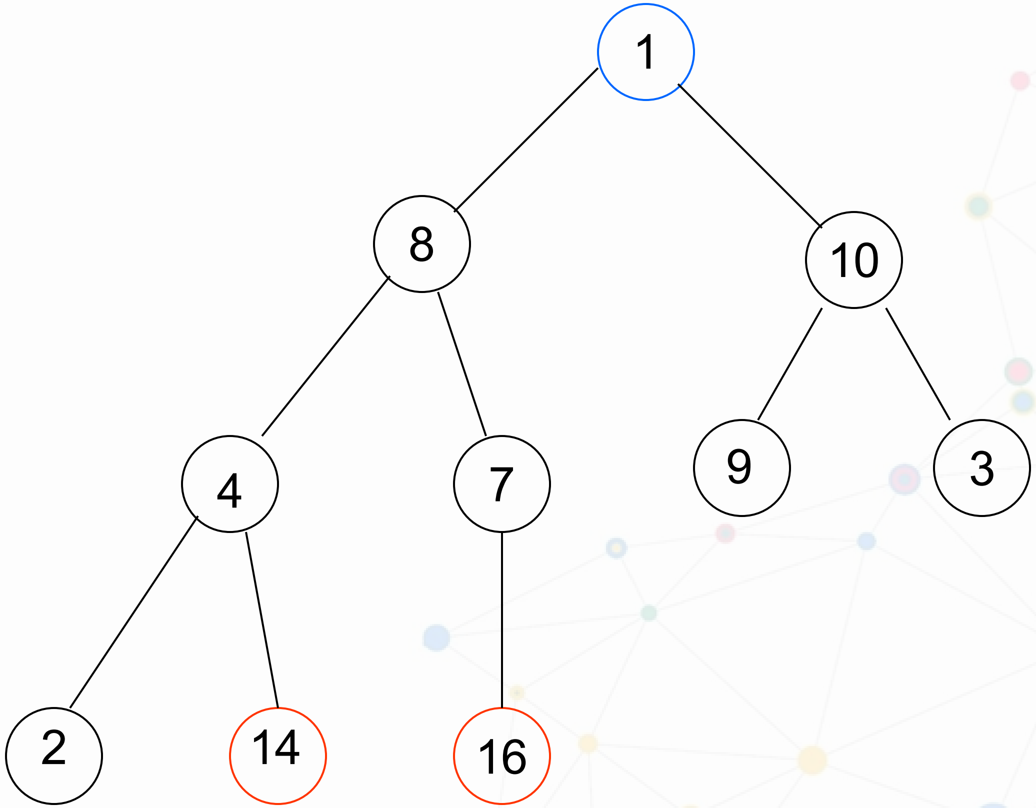


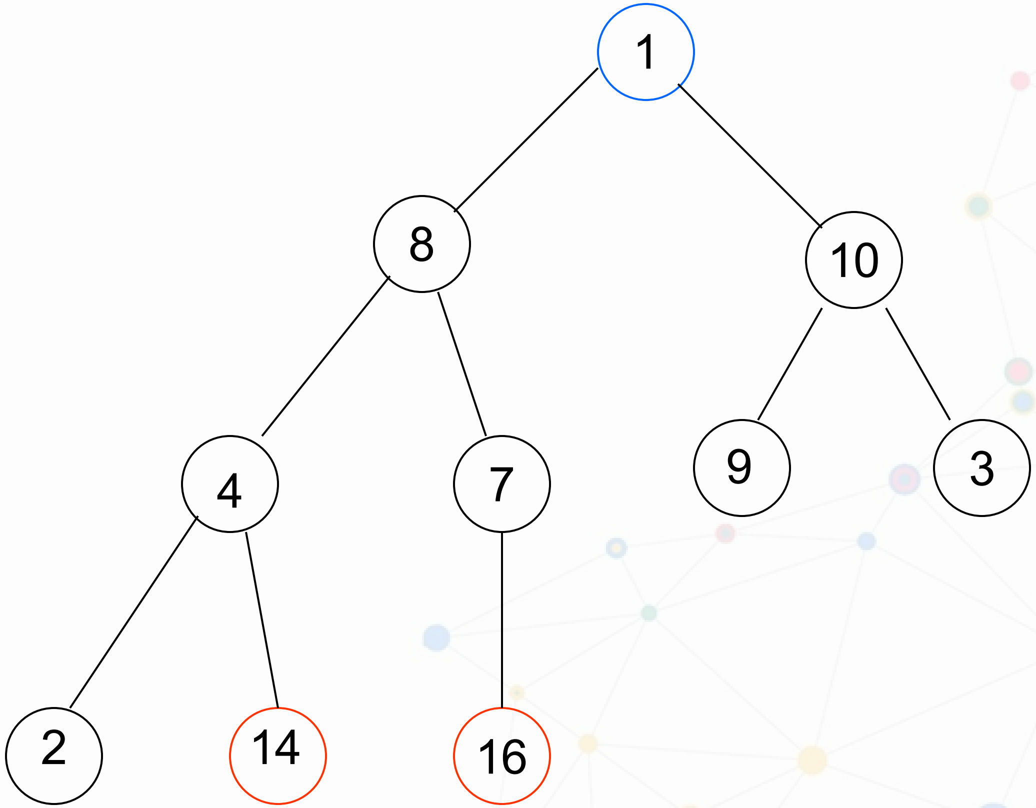
1, 14, 10, 8, 7, 9, 3, 2, 4, 16.


14
8
10
4
7
9
3
2
1
16

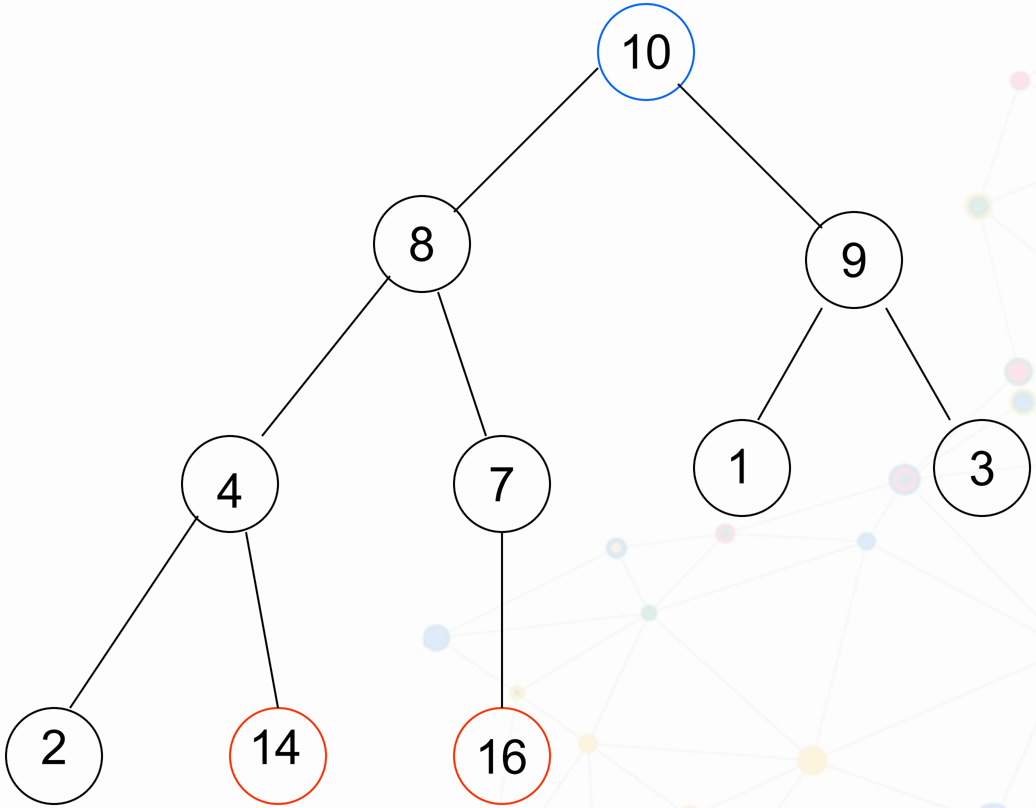


14, 8, 10, 4, 7, 9, 3, 2, 1, 16.

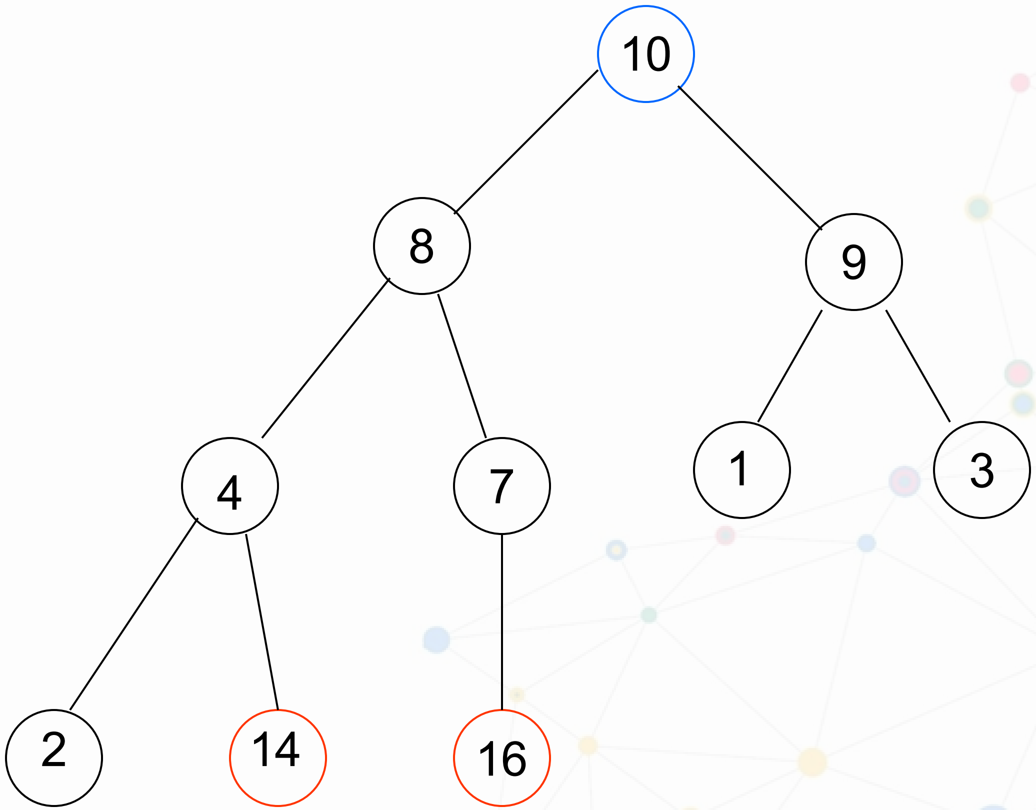

1
8
10
4
7
9
3
2
14
16



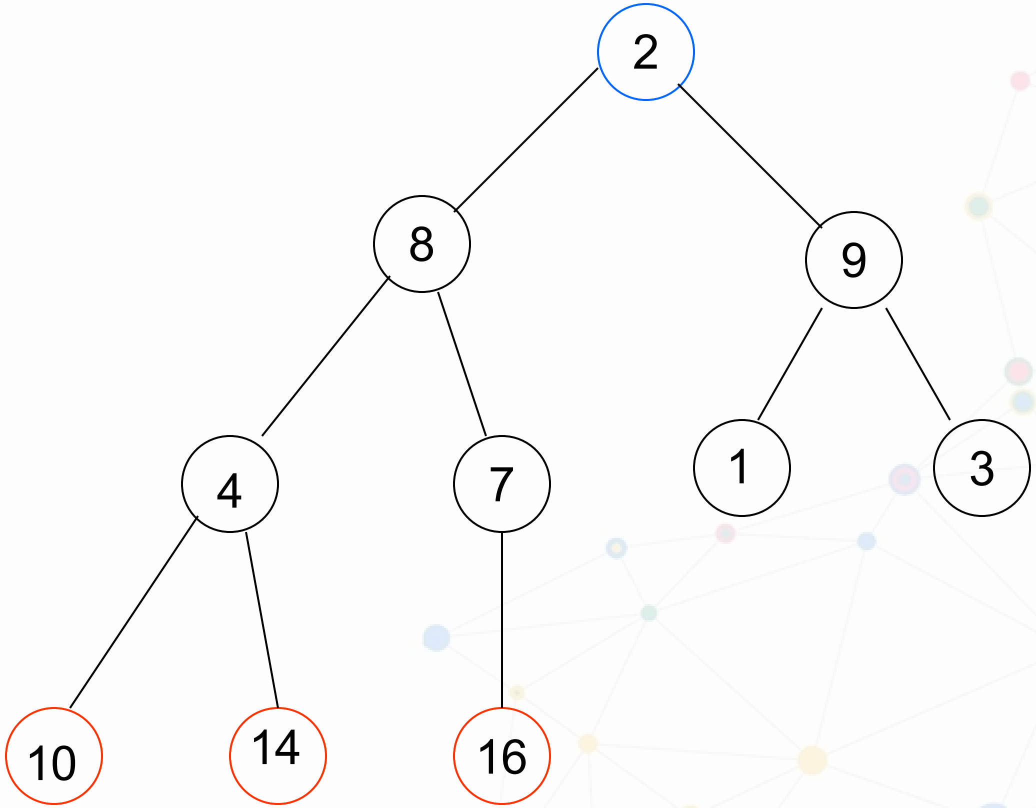
1, 8, 10, 4, 7, 9, 3, 2, 14, 16.

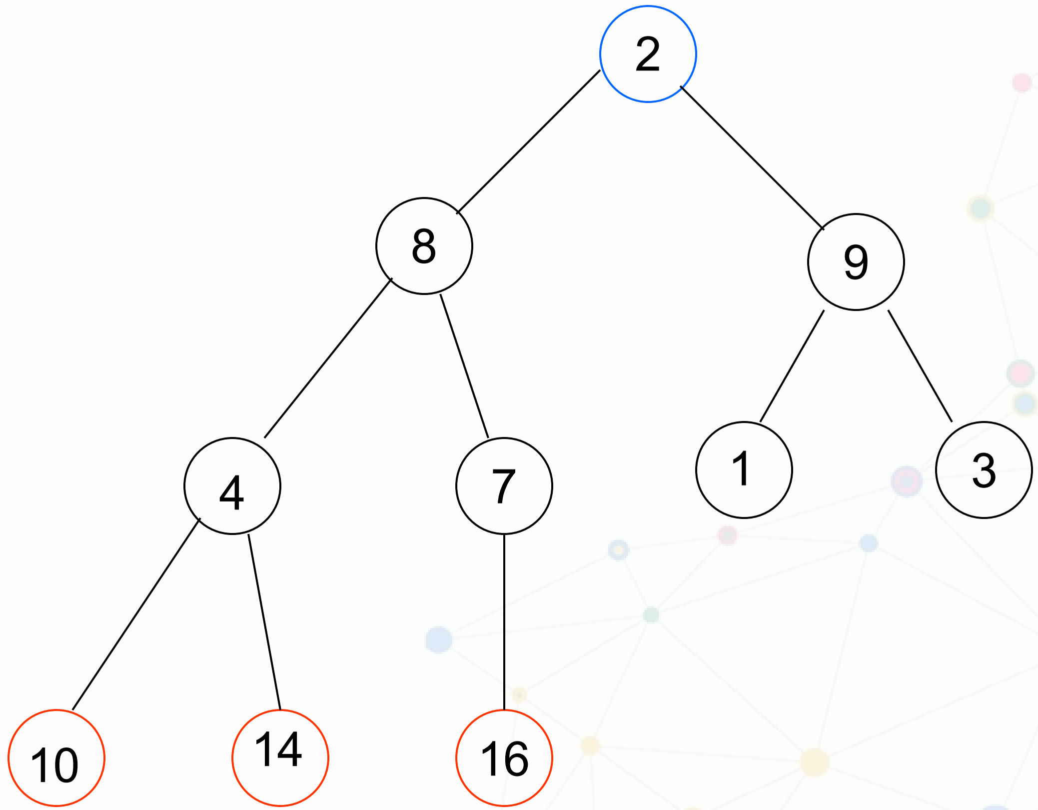


10
8
9
4
7
1
3
2
14
16

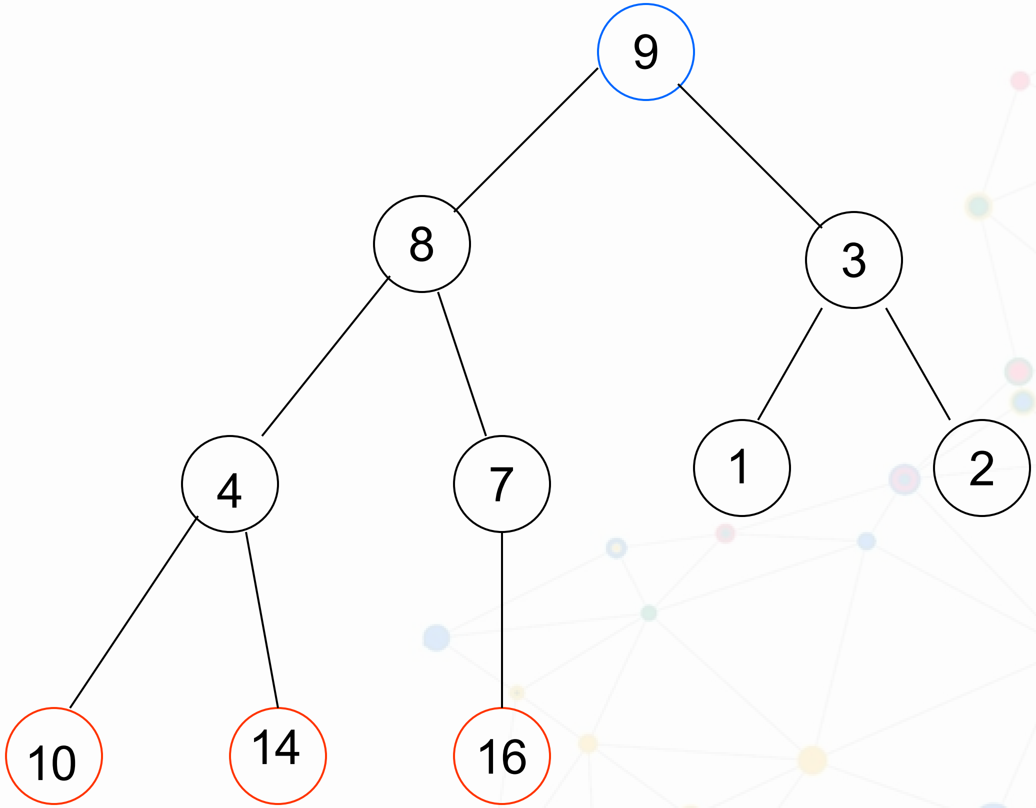


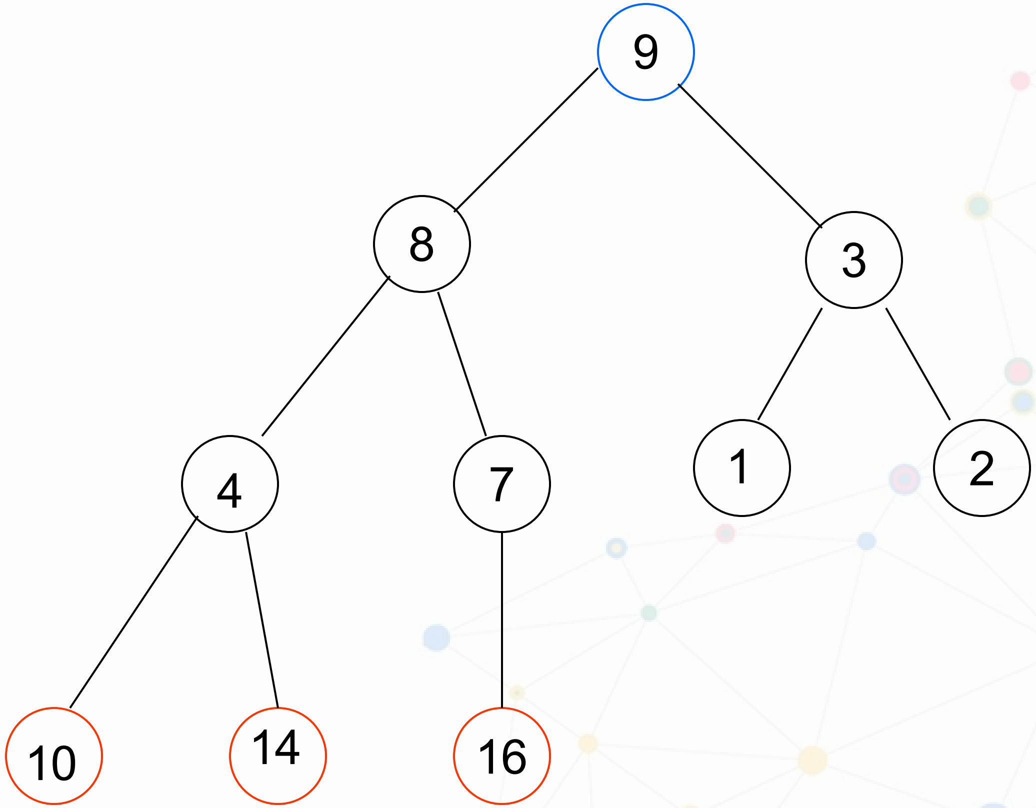
10, 8, 9, 4, 7, 1, 3, 2, 14, 16.


2
8
9
4
7
1
3
10
14
16

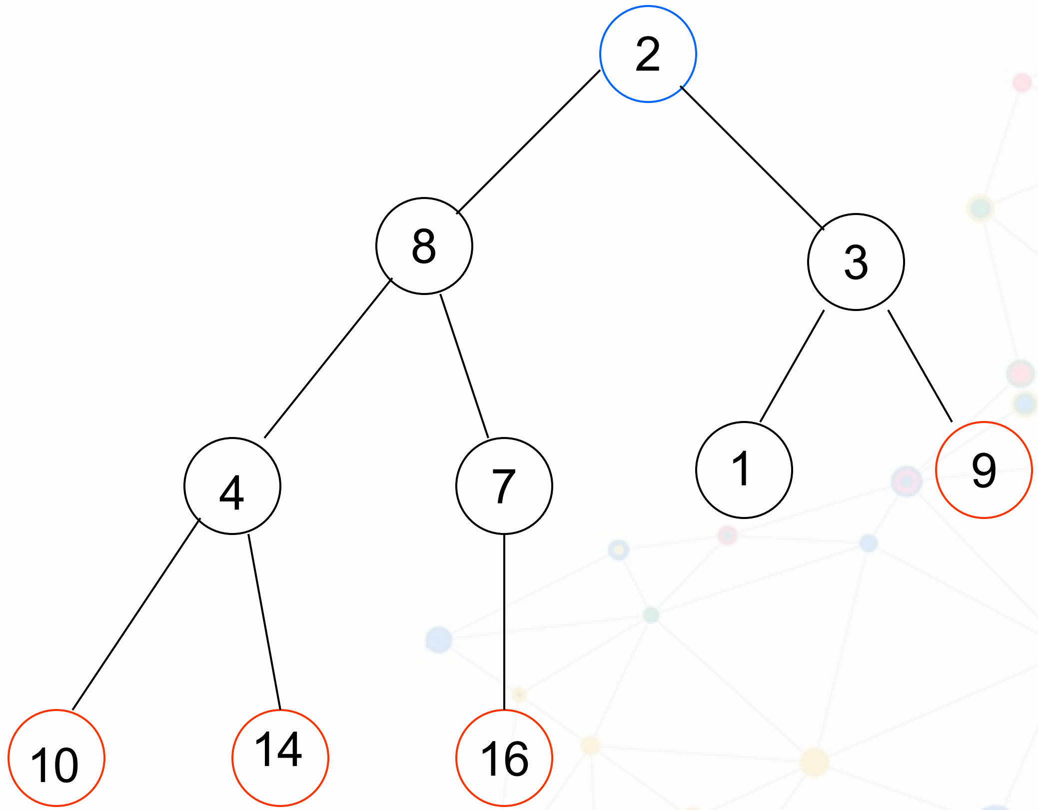


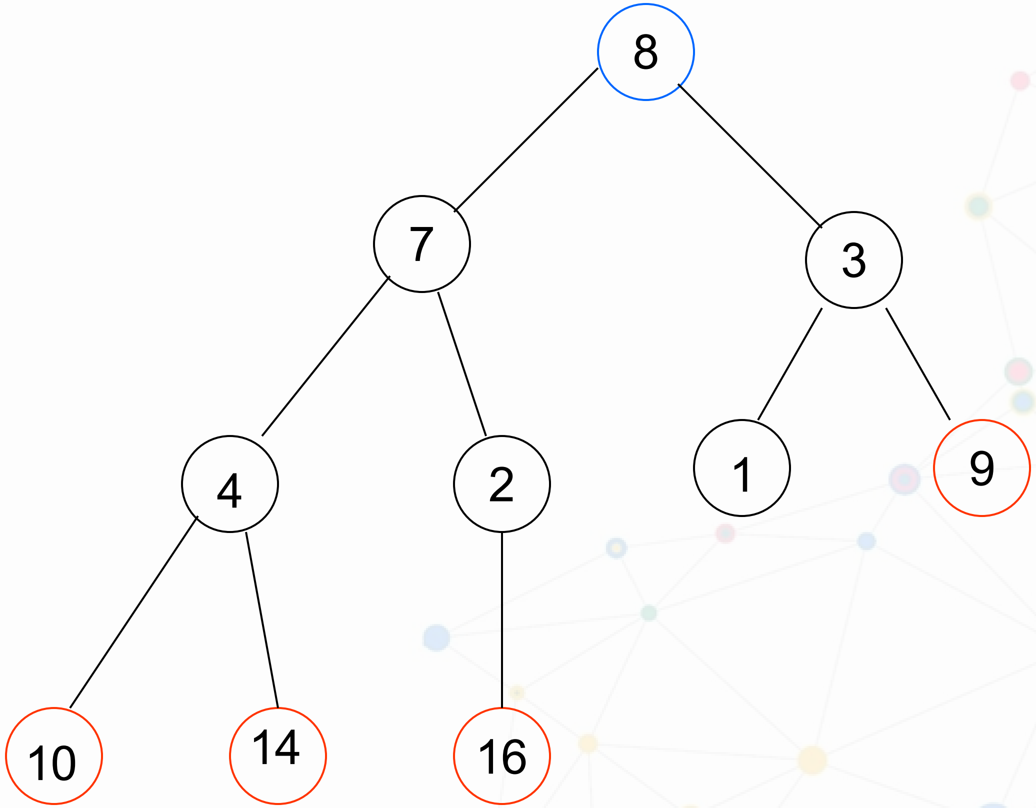
2, 8, 9, 4, 7, 1, 3, 10, 14, 16.

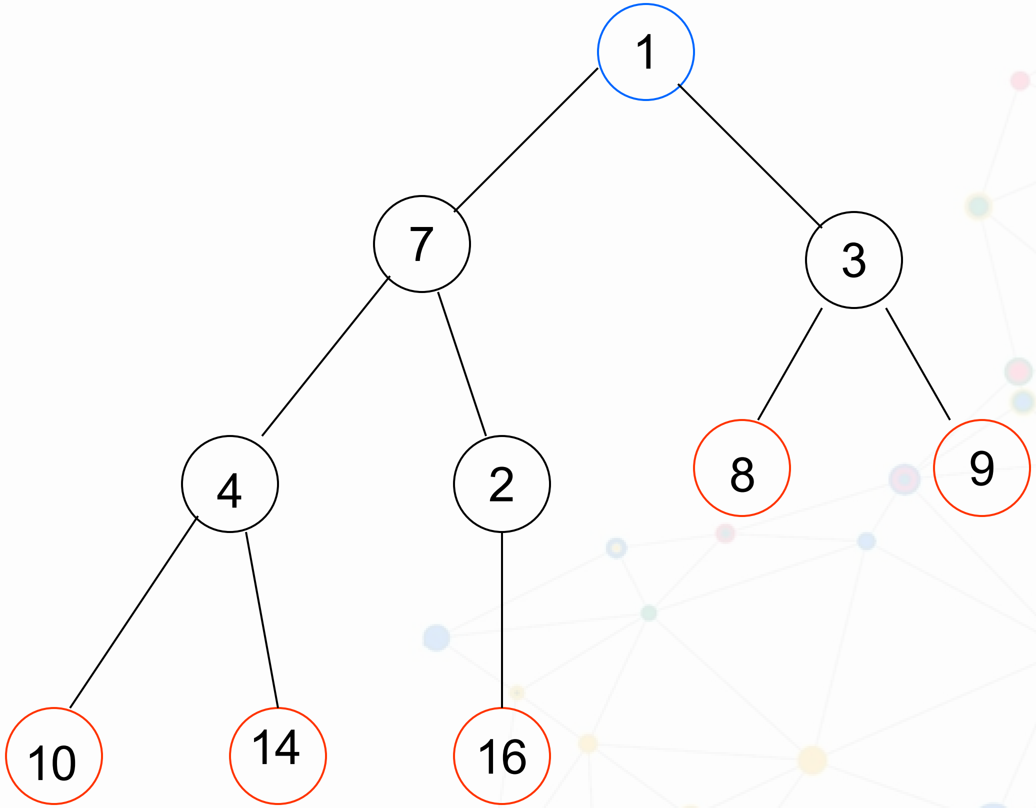

9
8
3
4
7
1
2
10
14
16

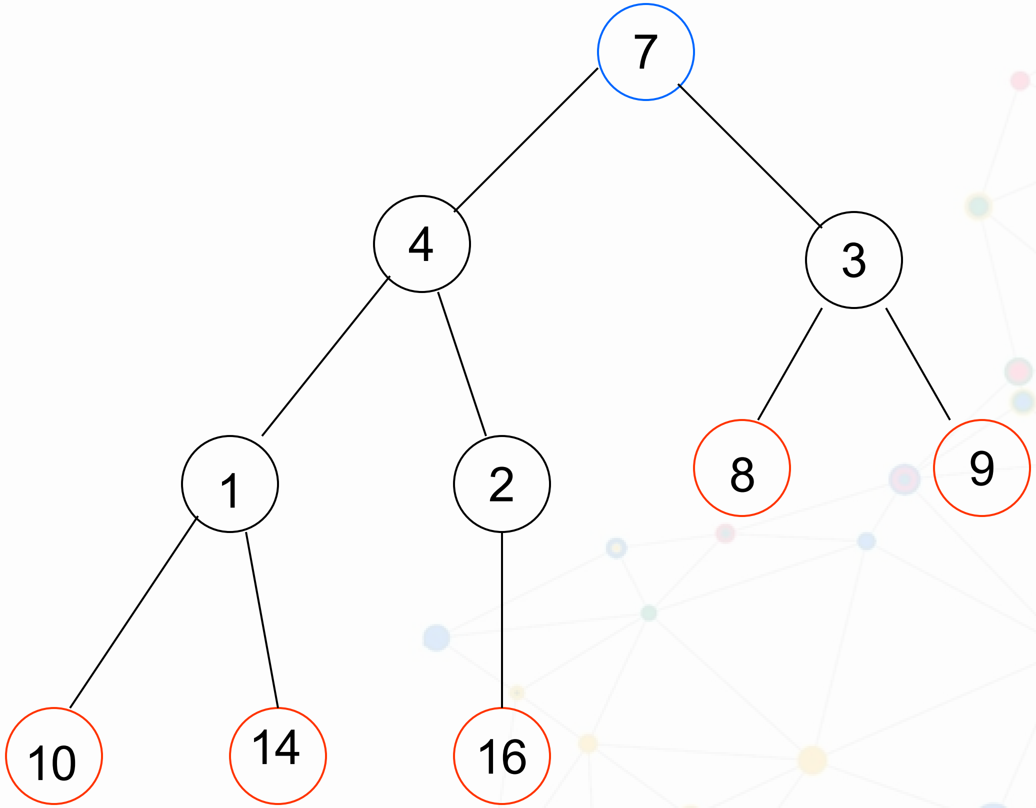


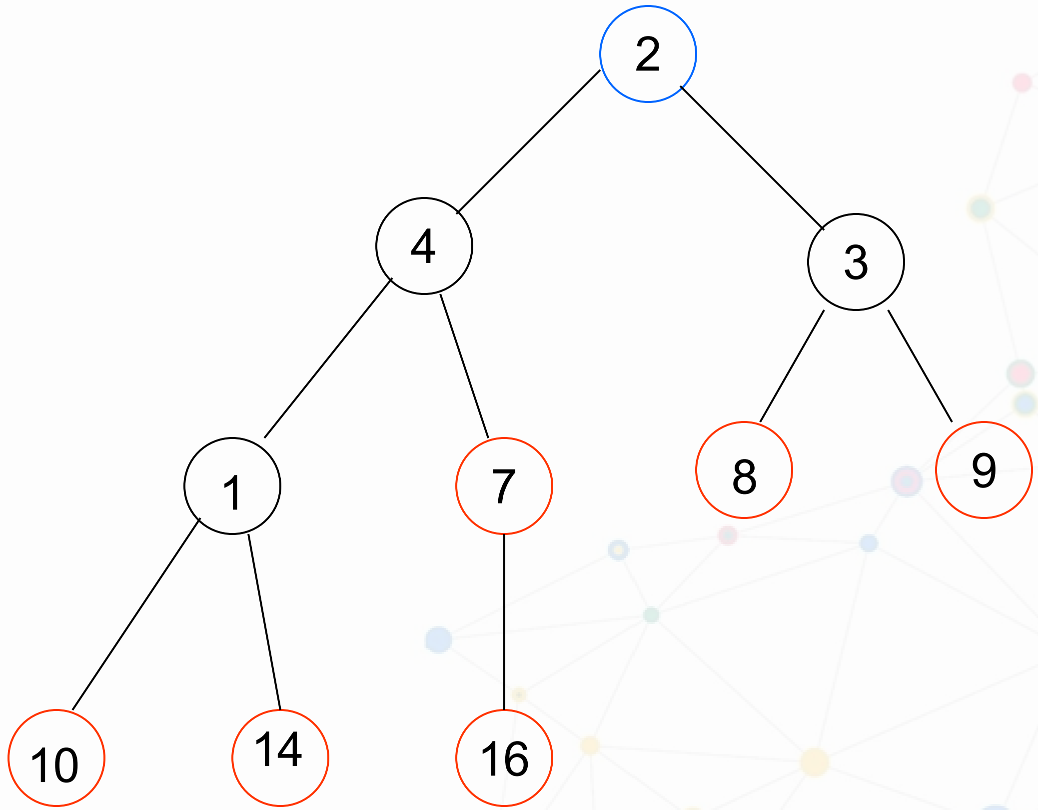
9, 8, 3, 4, 7, 1, 2, 10, 14, 16.

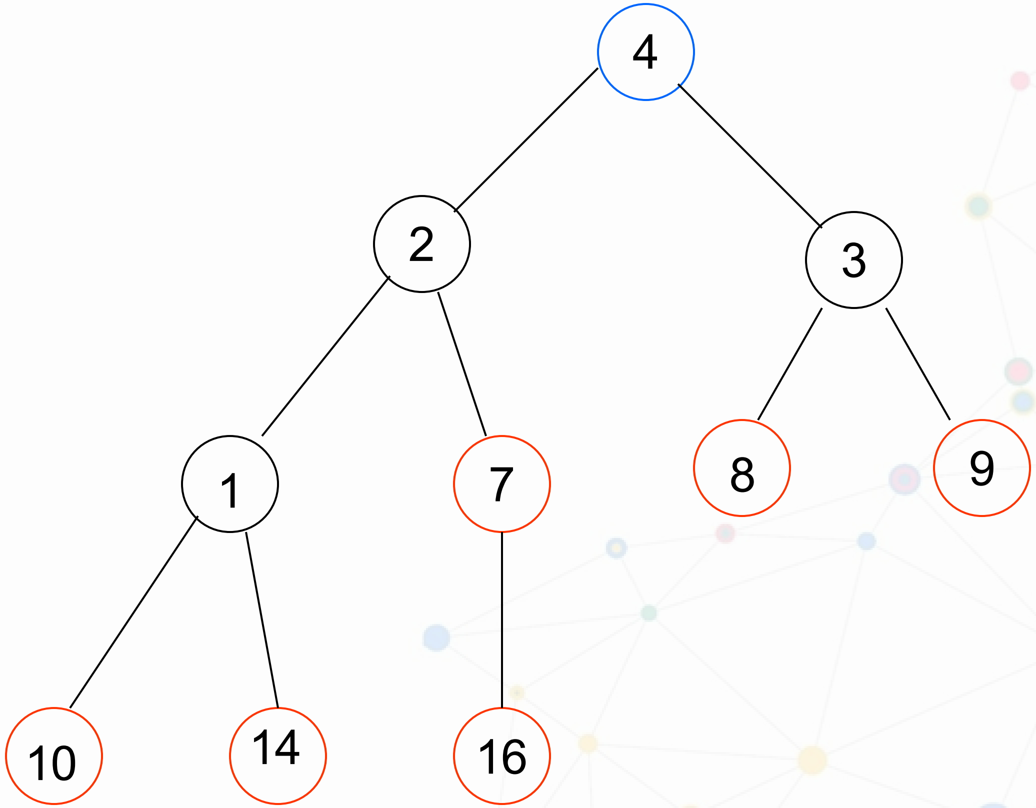

2
8
3
4
7
1
9
10
14
16

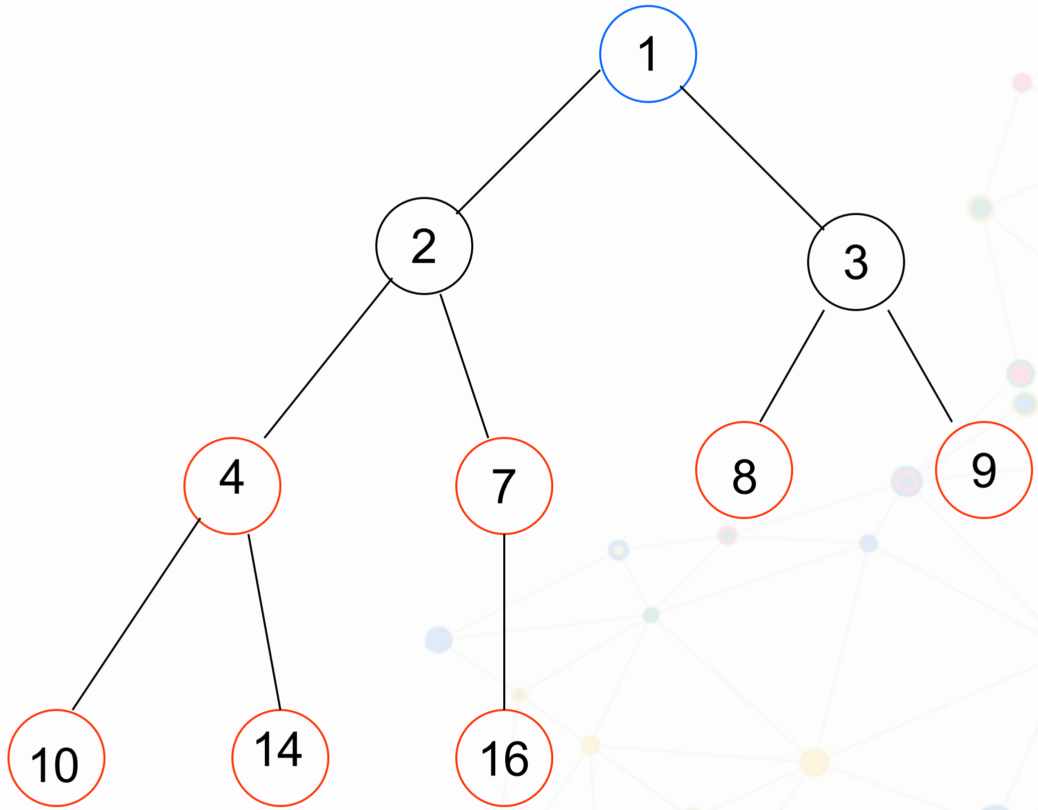

8
7
3
4
2
1
9
10
14
16

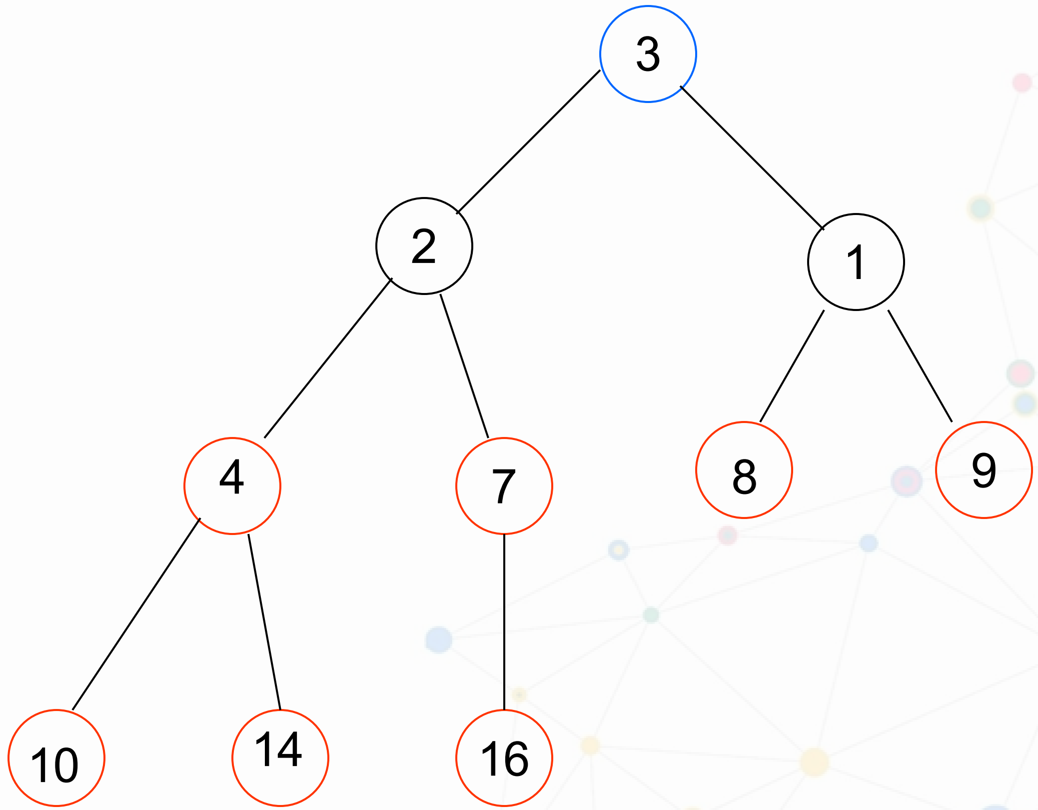

1
7
3
4
2
8
9
10
14
16

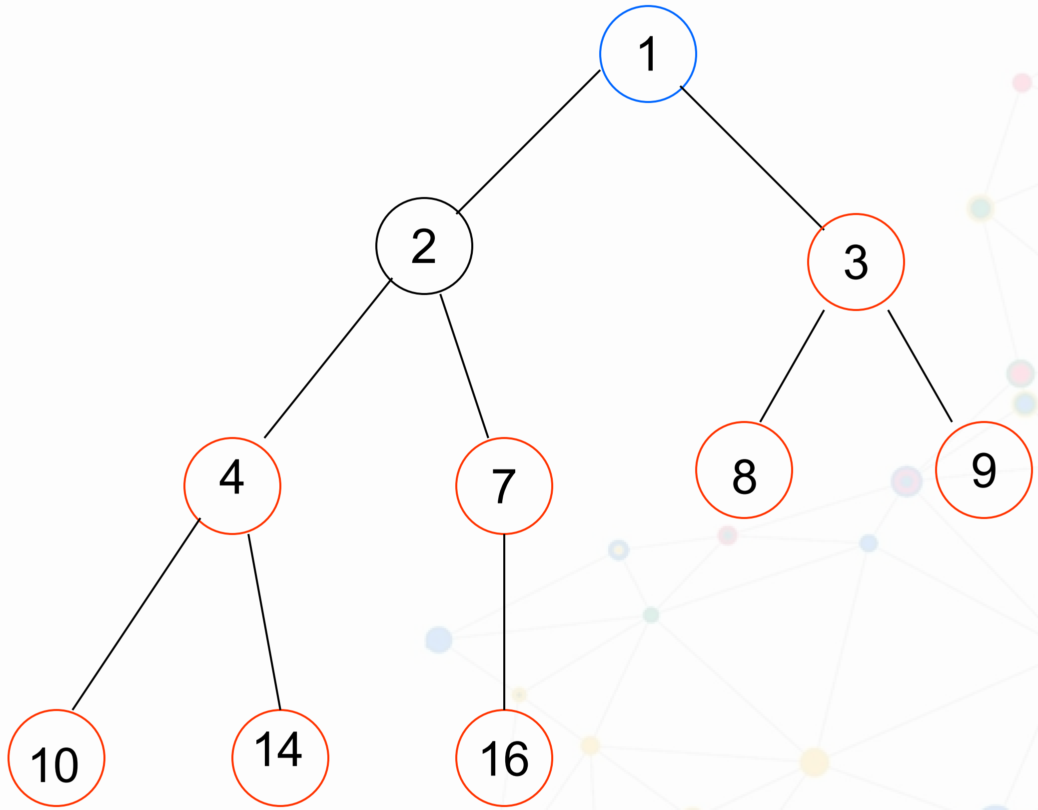

7
4
3
1
2
8
9
10
14
16

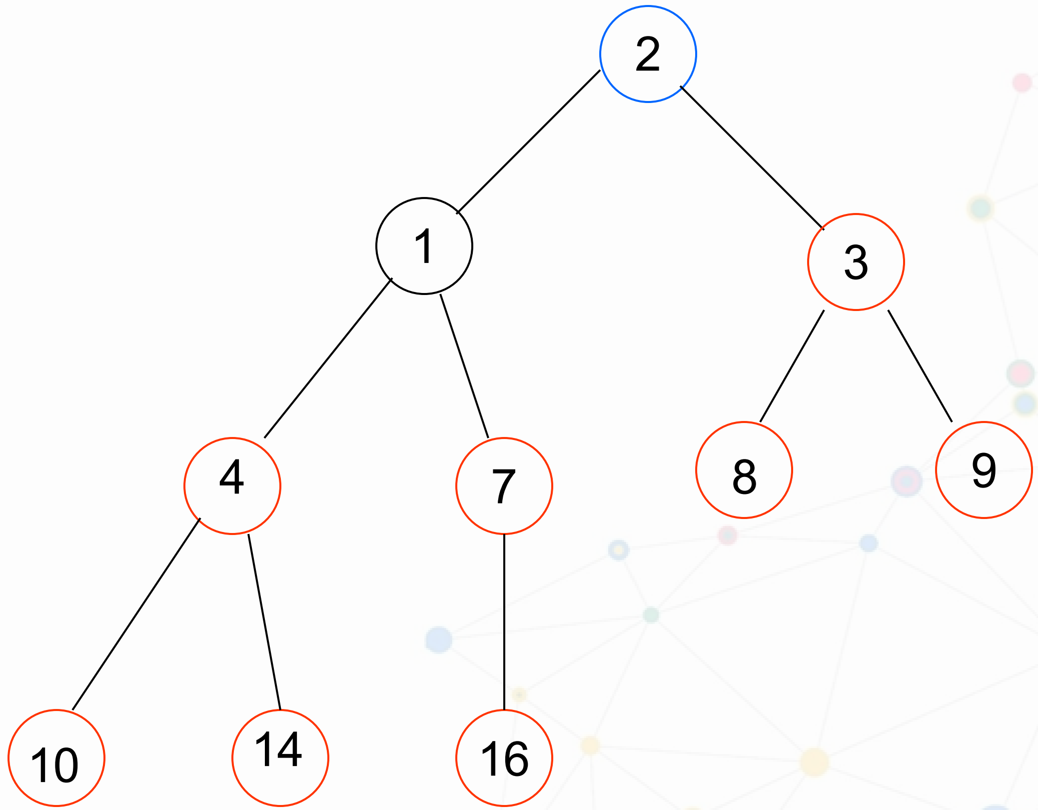

2
4
3
1
7
8
9
10
14
16

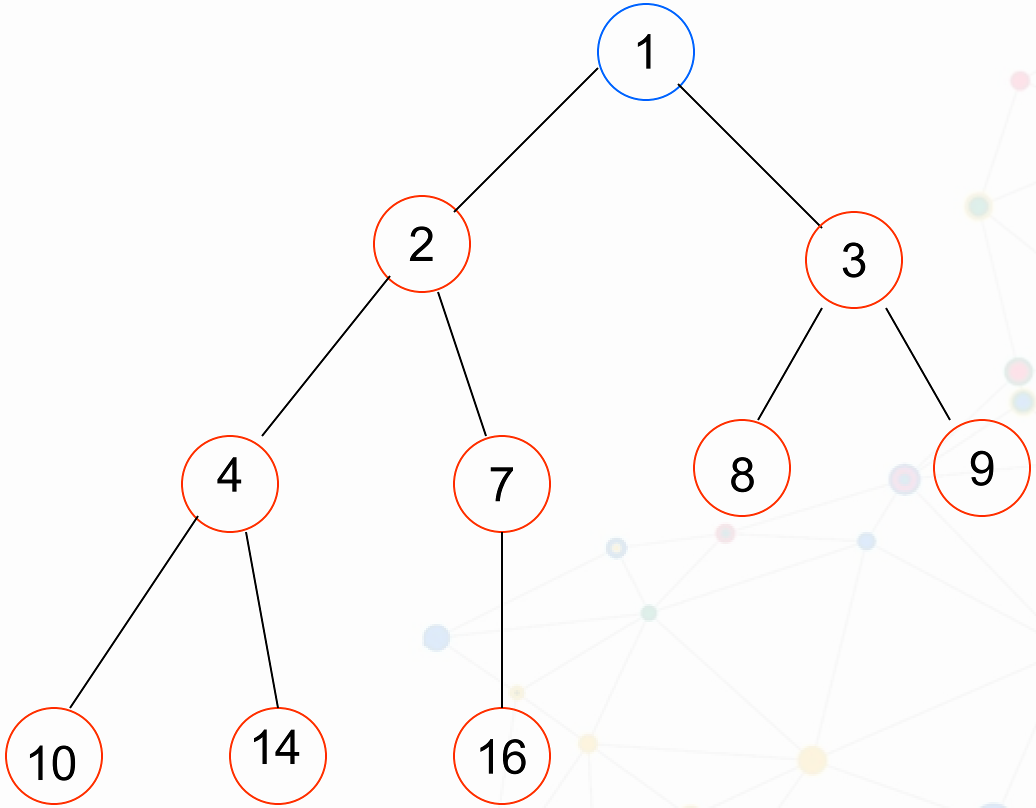

4
2
3
1
7
8
9
10
14
16


1
2
3
4
7
8
9
10
14
16


3
2
1
4
7
8
9
10
14
16


1
2
3
4
7
8
9
10
14
16


2
1
3
4
7
8
9
10
14
16


1
2
3
4
7
8
9
10
14
16

Running Time

$\text{Heapsort}(A)$

$\quad \text{Buid - Max - Heap}(A);$

$\quad \text{for } i \leftarrow \mathit{length}[A] \text{ downto } 2$ O(n)

$\quad\quad \text{do begin}$

$\quad\quad\quad \text{exchange } A[1] \leftrightarrow A[i];$

$\quad\quad\quad \mathit{heap\text{-}size}[A] \leftarrow \mathit{heap\text{-}size}[A] - 1;$

$\quad\quad\quad \text{Max - Heapify}(A, 1);$ O(lg n)

$\quad\quad \text{end - for}$

$O(n \lg n)$