Heap

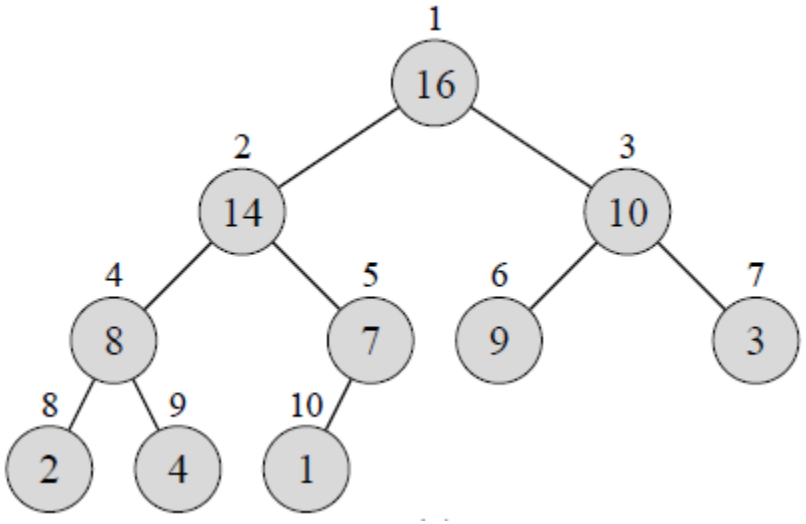
Prepared by:
Morteza Keshtkaran

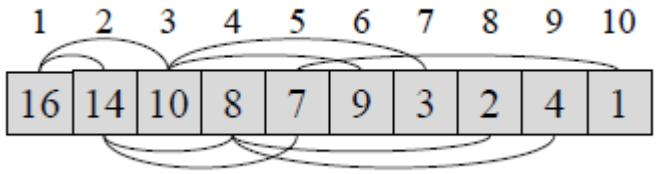
References:

CLRS: Chapter 6

Ghodsi: Chapter 4 (Section 6)

Max-Heap





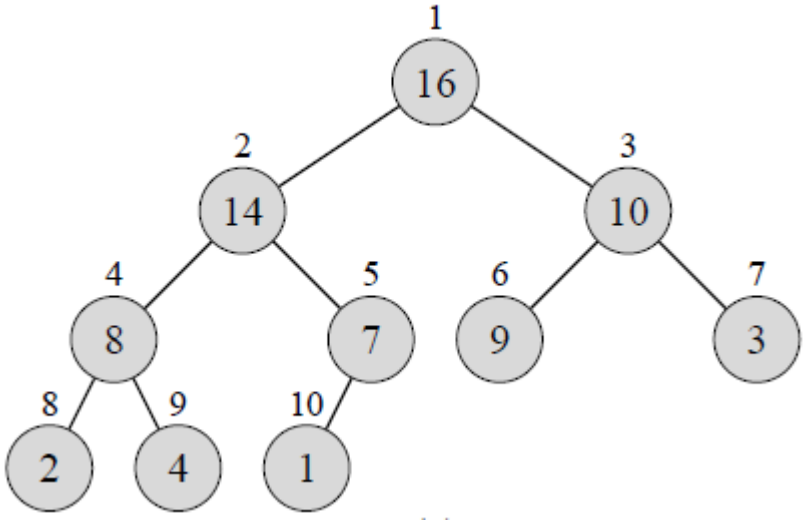
max-heap property

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

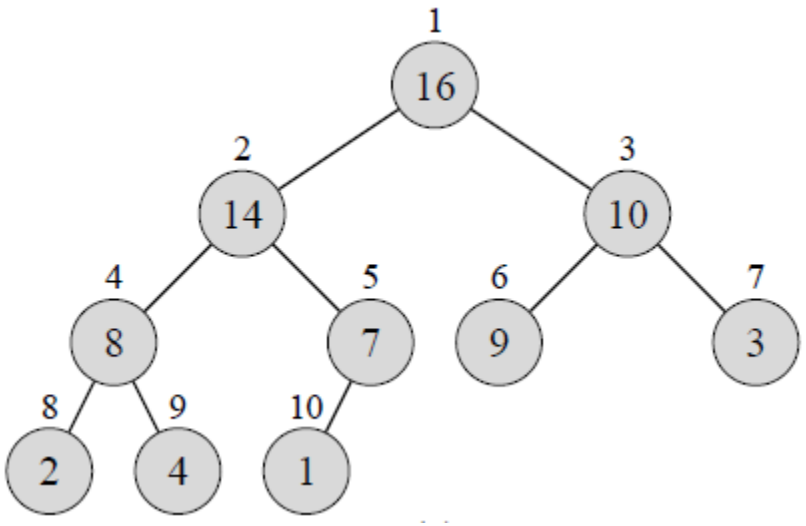
```
PARENT(i)
    return ⌊i/2⌋

LEFT(i)
    return 2i

RIGHT(i)
    return 2i + 1
```

Height

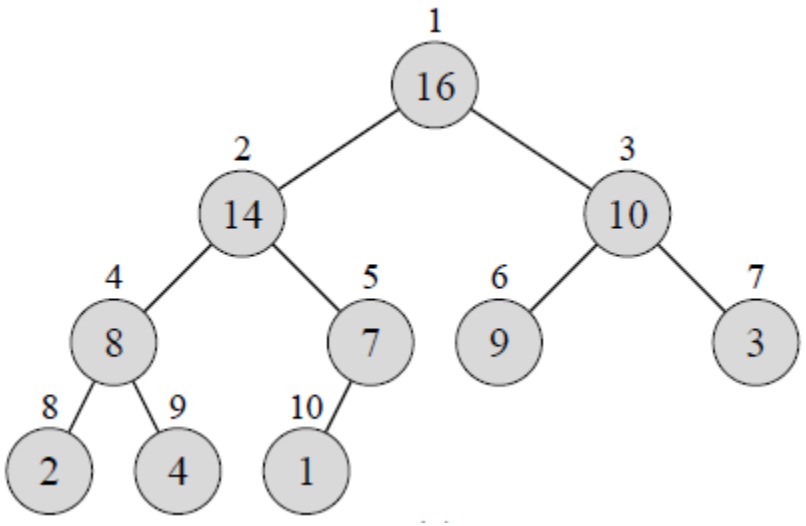
$$2^h \leq n \leq 2^{h+1} - 1$$
$$\Rightarrow$$
$$\lg(n+1) - 1 \leq h \leq \lg(n)$$



Insert

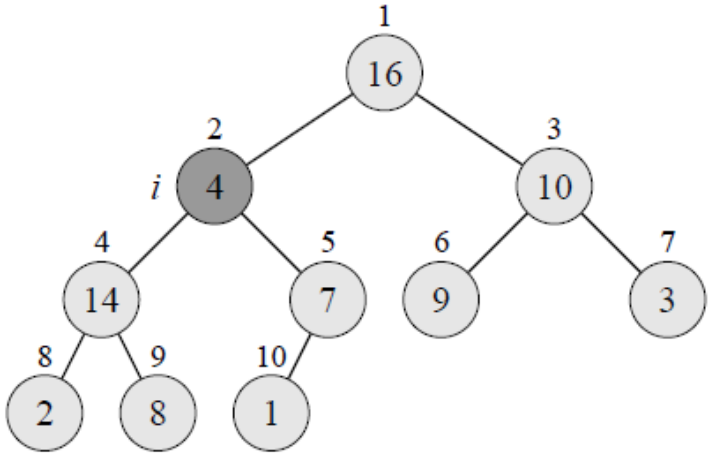


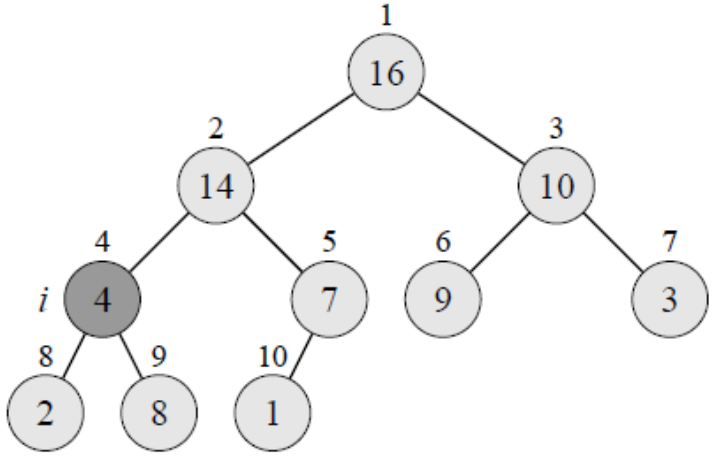
Delete Max

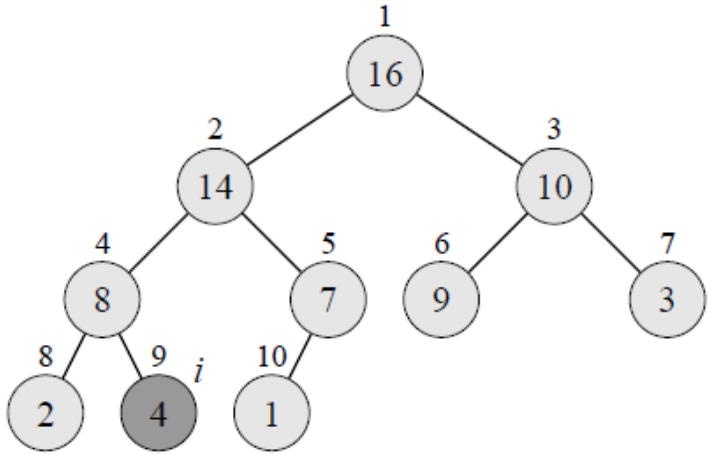


Max-Heapify

```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i]
    largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
    largest = r
if largest ≠ i
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
```

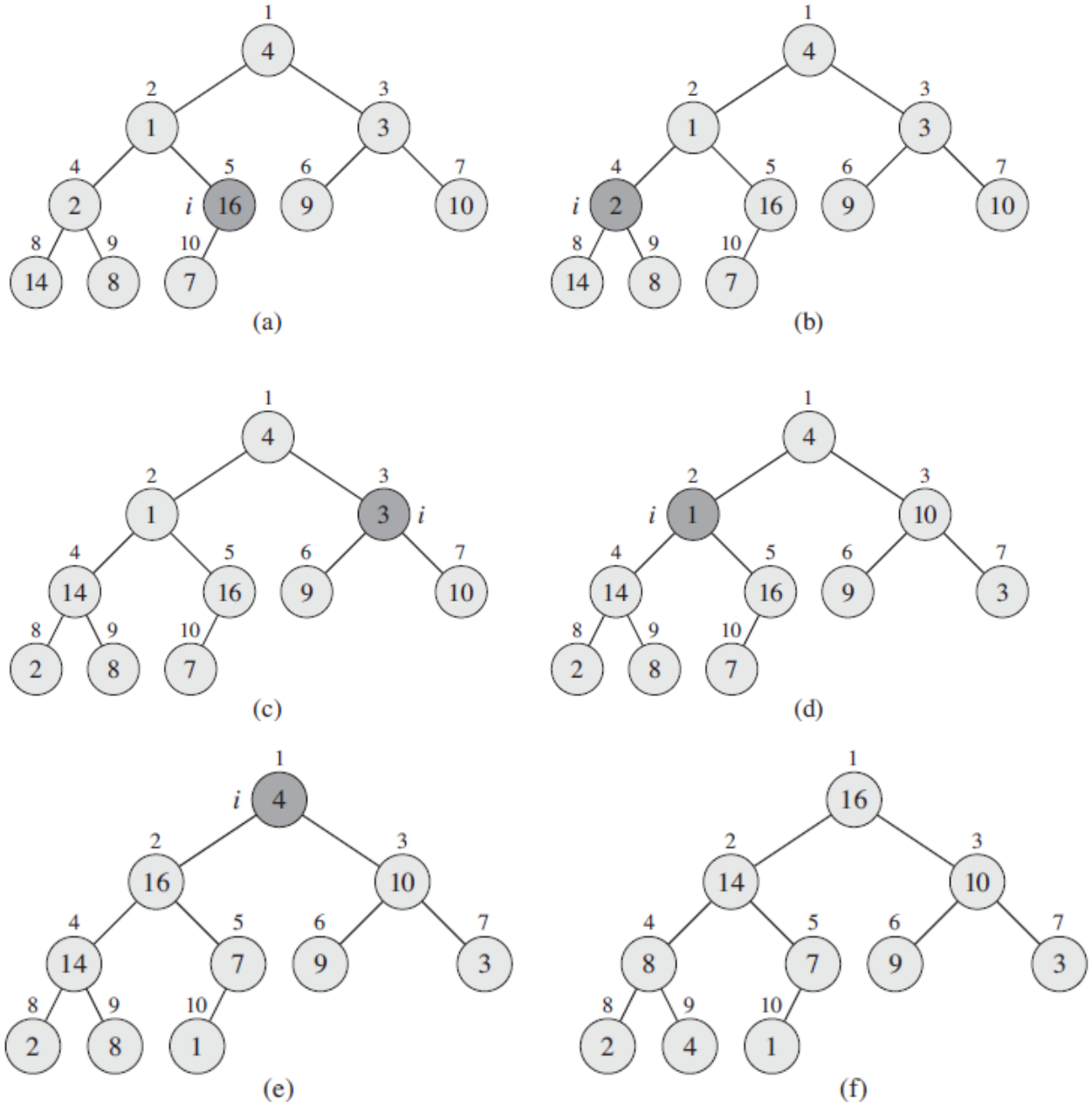






Building a heap

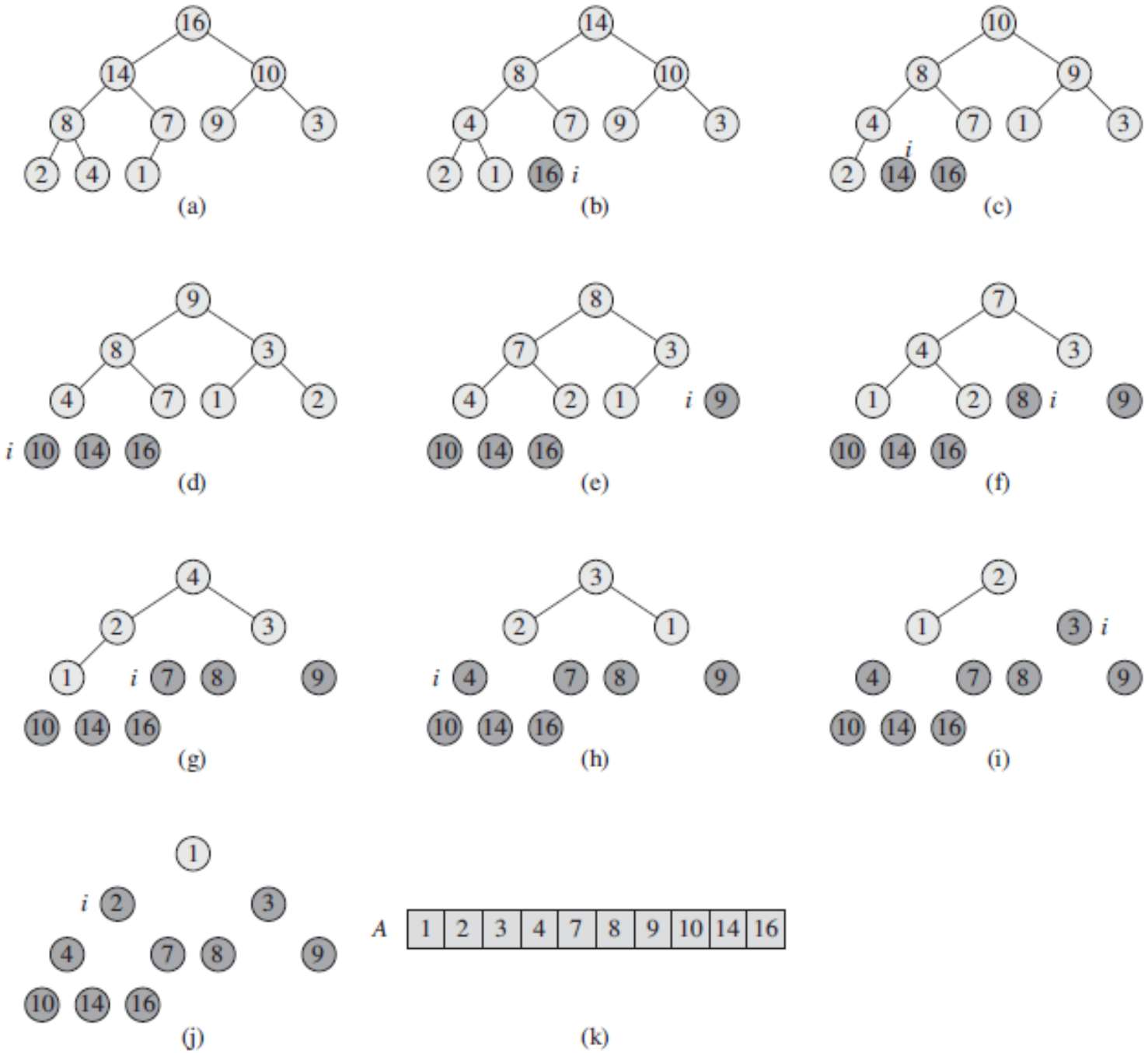
A									
4	1	3	2	16	9	10	14	8	7




```
BUILD-MAX-HEAP(A)
A.heap-size = A.length
for i = ⌊A.length/2⌋ downto 1
    MAX-HEAPIFY(A, i)
```

The heapsort algorithm

```
HEAPSORT(A)
BUILD-MAX-HEAP(A)
for i = A.length downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size - 1
    MAX-HEAPIFY(A, 1)
```


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
A
1 2 3 4 7 8 9 10 14 16

Heap Sort

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $\mathbf{O}(nlgn)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7
4	2	3	1	16	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
4	2	3	1	16	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
4	2	3	1	16	9	10	14	8	7
4	16	3	1	2	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
4	2	3	1	16	9	10	14	8	7
4	16	3	1	2	9	10	14	8	7
16	4	3	1	2	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
16	4	3	1	2	9	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
16	4	3	1	2	9	10	14	8	7
16	4	9	1	2	3	10	14	8	7

Heap Sort: Build Heap (Top Down) $O(nlgn)$

Insert in the same way

1	2	3	4	5	6	7	8	9	10
16	4	9	1	2	3	10	14	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7
4	1	3	14	16	9	10	2	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	1	3	14	16	9	10	2	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	1	3	14	16	9	10	2	8	7
4	1	10	14	16	9	3	2	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	1	10	14	16	9	3	2	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	1	10	14	16	9	3	2	8	7
4	16	10	14	1	9	3	2	8	7

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
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

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	16	10	14	7	9	3	2	8	1

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
4	16	10	14	7	9	3	2	8	1
16	4	10	14	7	9	3	2	8	1

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
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

Heap Sort: Build Heap (Bottom Up) $\theta(n)$

1	2	3	4	5	6	7	8	9	10
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

Heap Sort: Delete (n-1 times) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

Heap Sort: Delete (n-1 times) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

Heap Sort: Delete (n-1 times) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
1	14	10	8	7	9	3	2	4	16

Heap Sort: Delete (n-1 times) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
1	14	10	8	7	9	3	2	4	16
14	1	10	8	7	9	3	2	4	16

Heap Sort: Delete (n-1 times) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
1	14	10	8	7	9	3	2	4	16
14	1	10	8	7	9	3	2	4	16
14	8	10	1	7	9	3	2	4	16

Heap Sort: Delete (n-1 times) $O(nlgn)$

1	2	3	4	5	6	7	8	9	10
1	14	10	8	7	9	3	2	4	16
14	1	10	8	7	9	3	2	4	16
14	8	10	1	7	9	3	2	4	16
14	8	10	4	7	9	3	2	4	16

Heap Sort: Delete (n-1 times) $O(nlgn)$

Delete in the same way

1	2	3	4	5	6	7	8	9	10
14	8	10	4	7	9	3	2	4	16