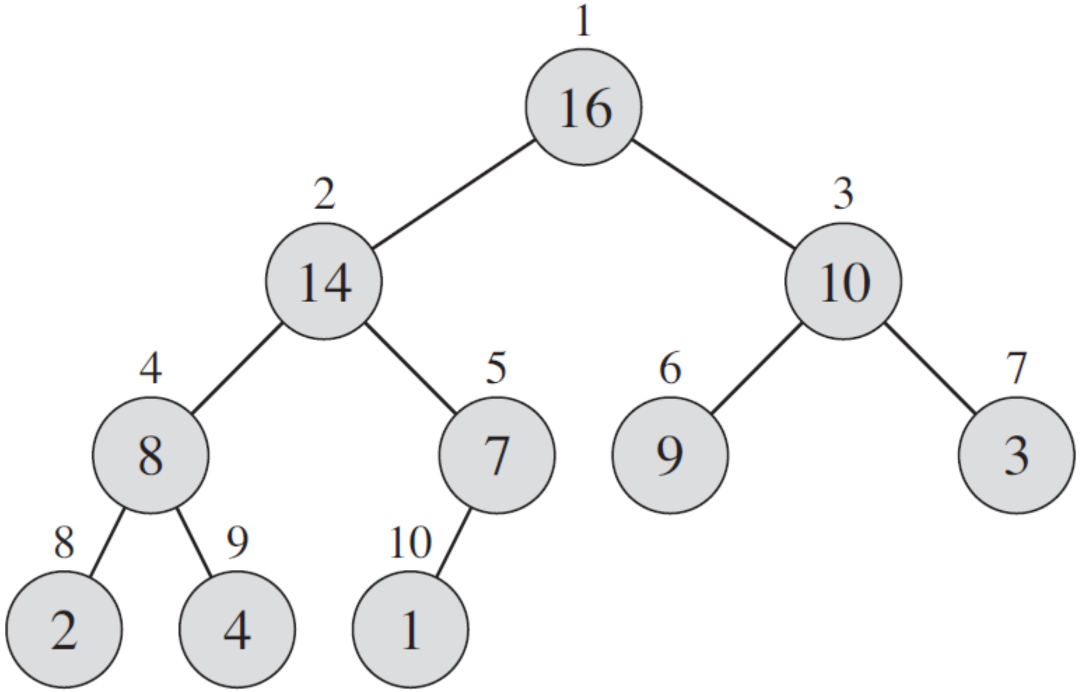
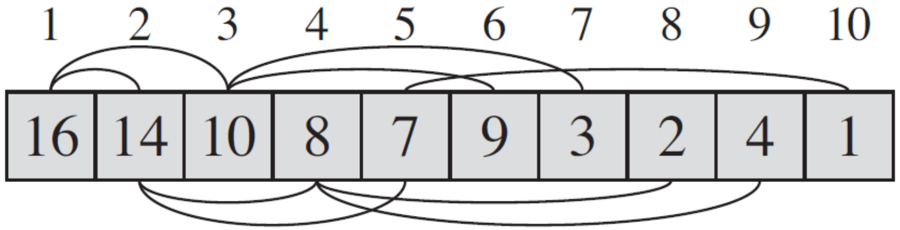
GDI2 – 5 - Heapsort

Heap and ist representation as array



(a)



(b)

PARENT(i)

return $\lfloor i/2 \rfloor$

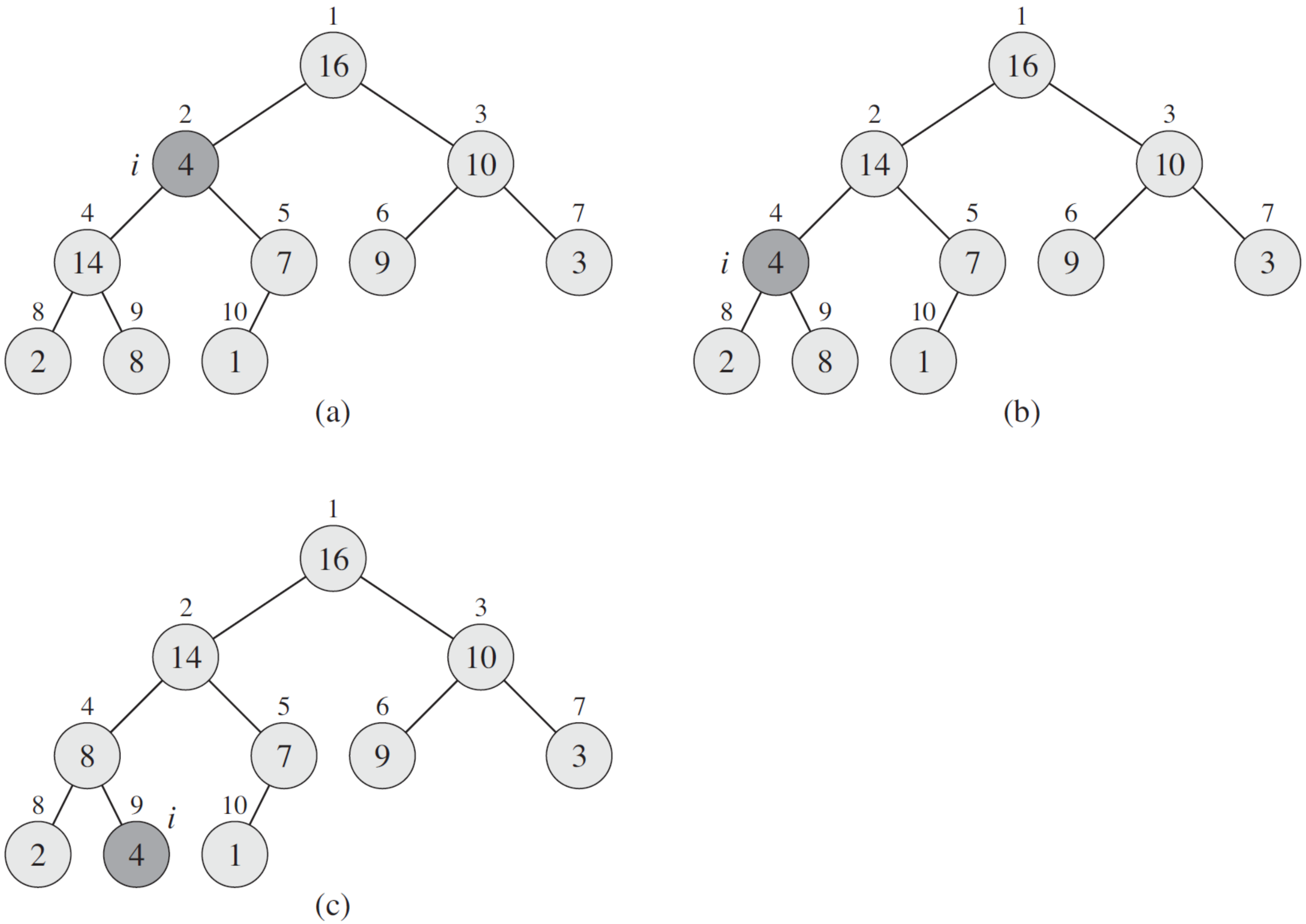
LEFT(i)

return $2i$

RIGHT(i)

return $2i + 1$

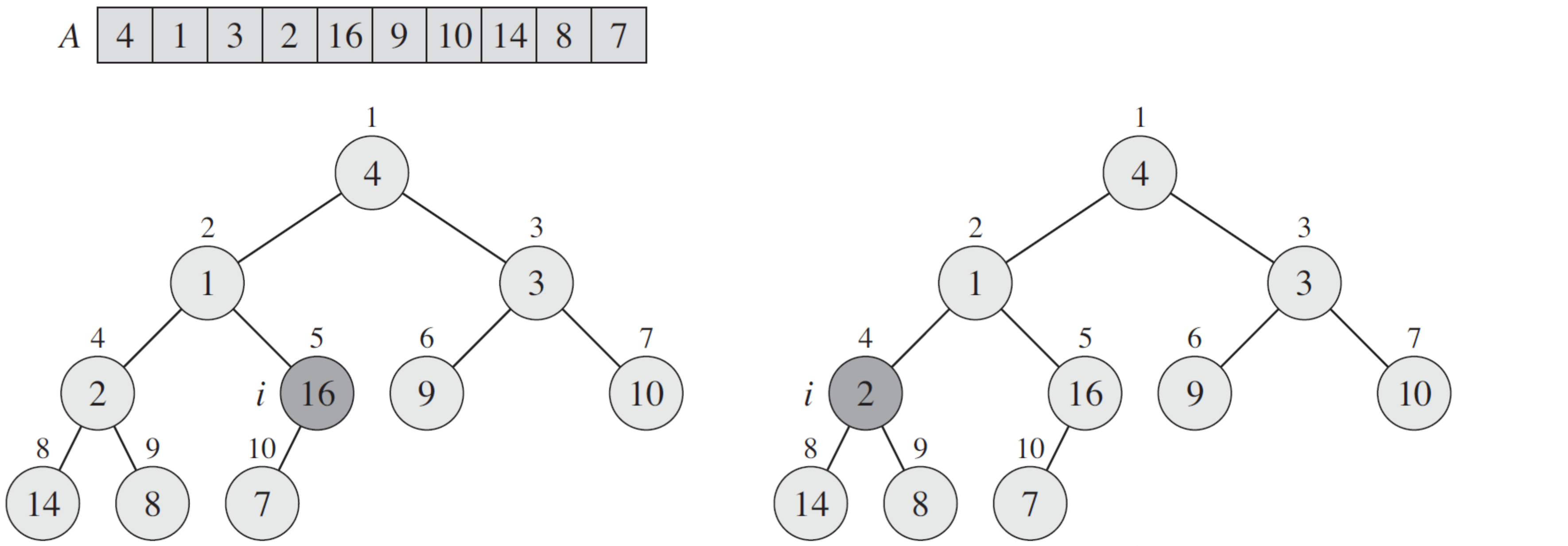
MAX-HEAPIFY



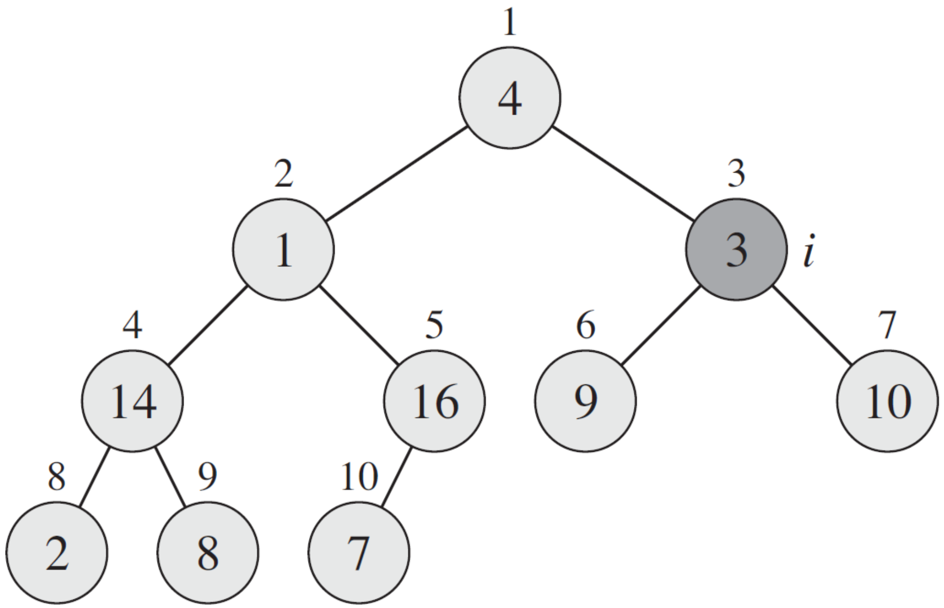
```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i]
    largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
    largest = r
if largest ≠ i
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
```

BUILD-MAX-HEAP(A)

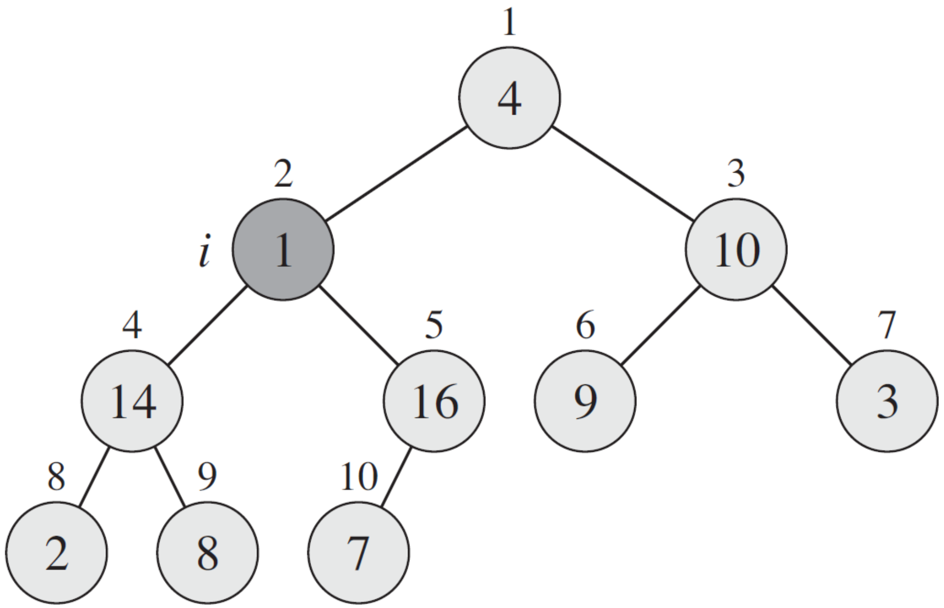
$A.heap\text{-}size = A.length$
for $i = \lfloor A.length/2 \rfloor$ **downto** 1
  MAX-HEAPIFY(A, i)



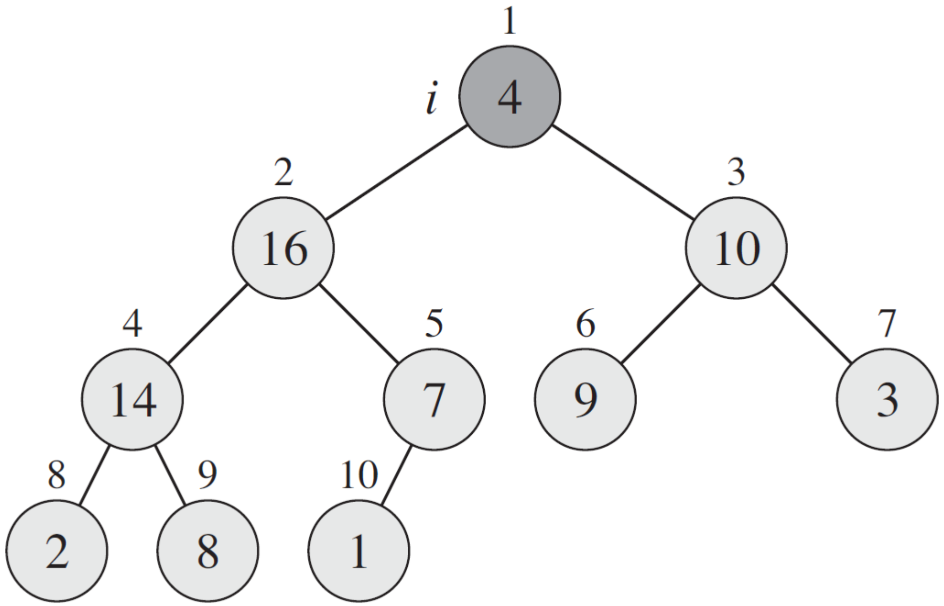
(a) (b)


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

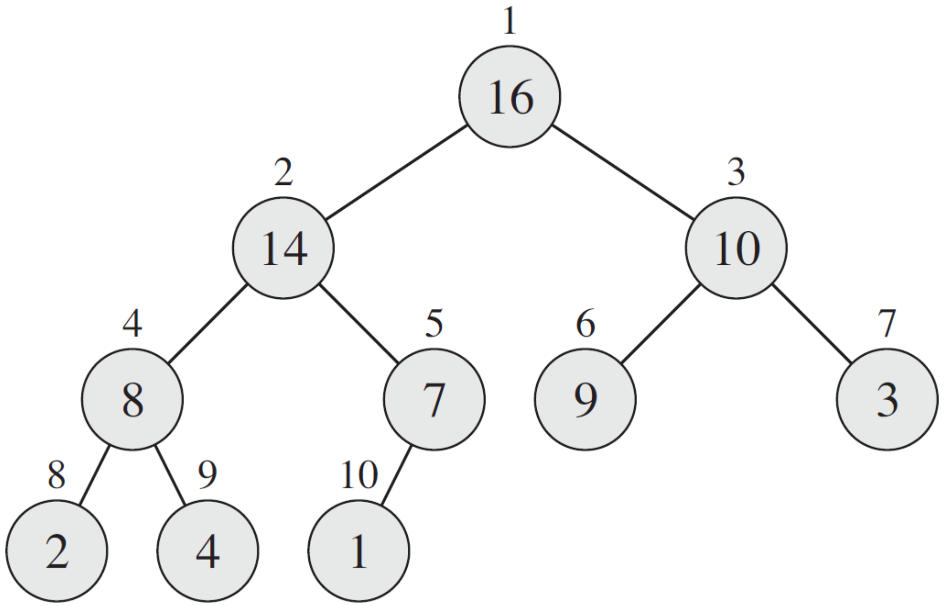
(c)


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

(d)



(e)



(f)

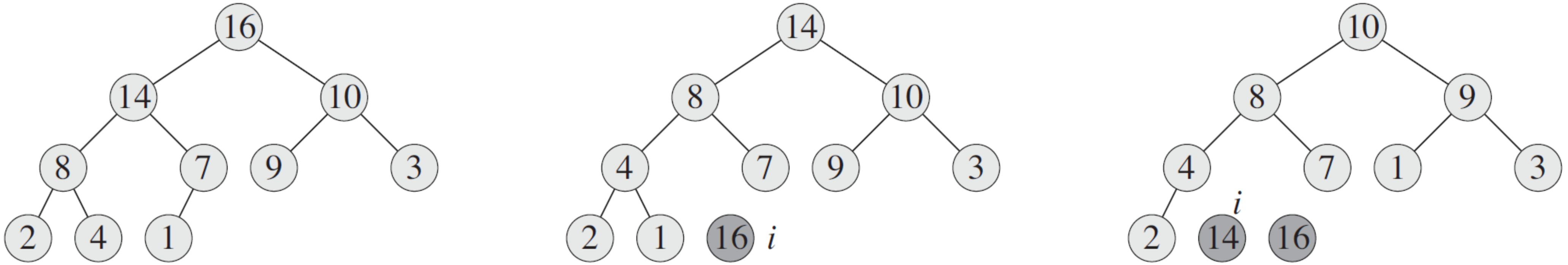
At the start of each iteration of the **for** loop of lines 2–3, each node $i + 1, i + 2, \ldots, n$ is the root of a max-heap.

Initialization: Prior to the first iteration of the loop, $i = \lfloor n/2 \rfloor$. Each node $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, n$ is a leaf and is thus the root of a trivial max-heap.

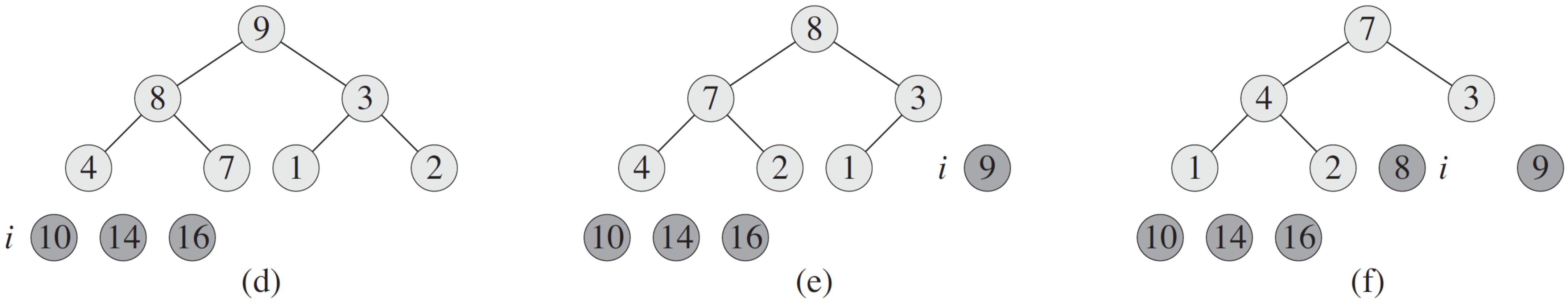
Maintenance: To see that each iteration maintains the loop invariant, observe that the children of node i are numbered higher than i. By the loop invariant, therefore, they are both roots of max-heaps. This is precisely the condition required for the call MAX-HEAPIFY(A, i) to make node i a max-heap root. Moreover, the MAX-HEAPIFY call preserves the property that nodes $i + 1, i + 2, \ldots, n$ are all roots of max-heaps. Decrementing i in the **for** loop update reestablishes the loop invariant for the next iteration.

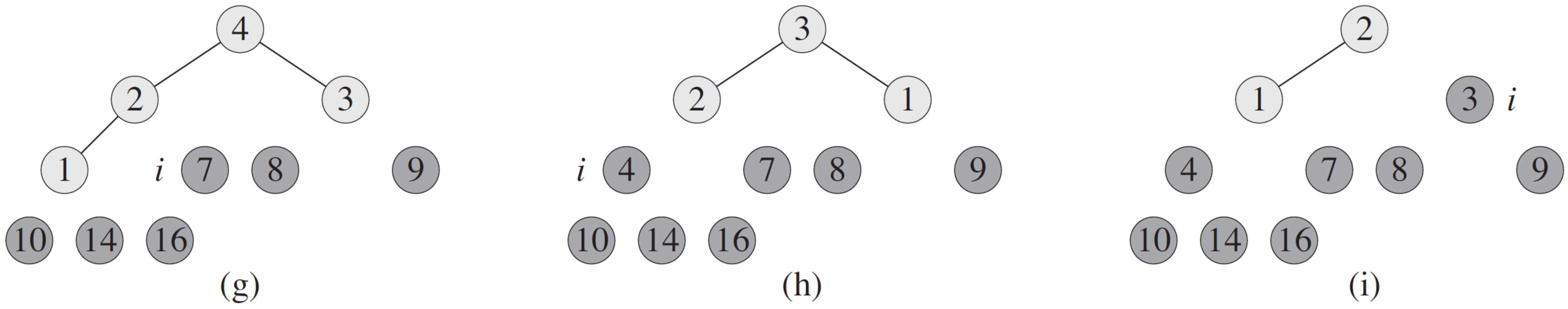
Termination: At termination, $i = 0$. By the loop invariant, each node $1, 2, \ldots, n$ is the root of a max-heap. In particular, node 1 is.

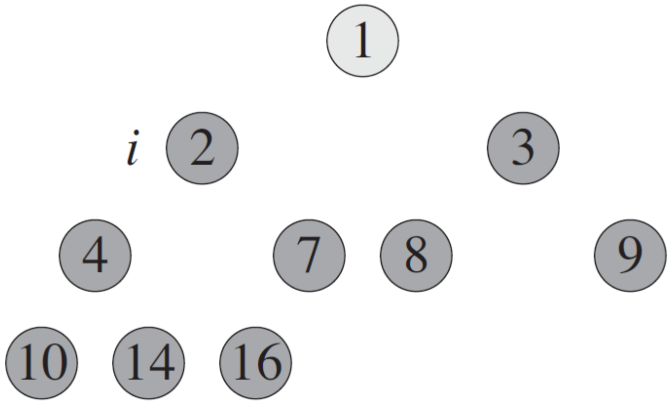
```
HEAPSORT(A)
BUILD-MAX-HEAP(A)
for i = A.length downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```



(a) (b) (c)


9
8
3
4
7
1
2
i
10
14
16
(d)
8
7
3
4
2
1
i
9
10
14
16
(e)
7
4
3
1
2
8
i
9
10
14
16
(f)


4
2
3
1
i
7
8
9
10
14
16
(g)
3
2
1
i
4
7
8
9
10
14
16
(h)
2
1
3
i
4
7
8
9
10
14
16
(i)



(j)

(k)