Heap Sort

Heaps, Heap property, Building a heap, Heap sort algorithm

Rashed Hassan Siam

Department of CSE

Eastern University

Introduction to Heaps

- Heapsort is a sorting algorithm with a running time of O(n log n).

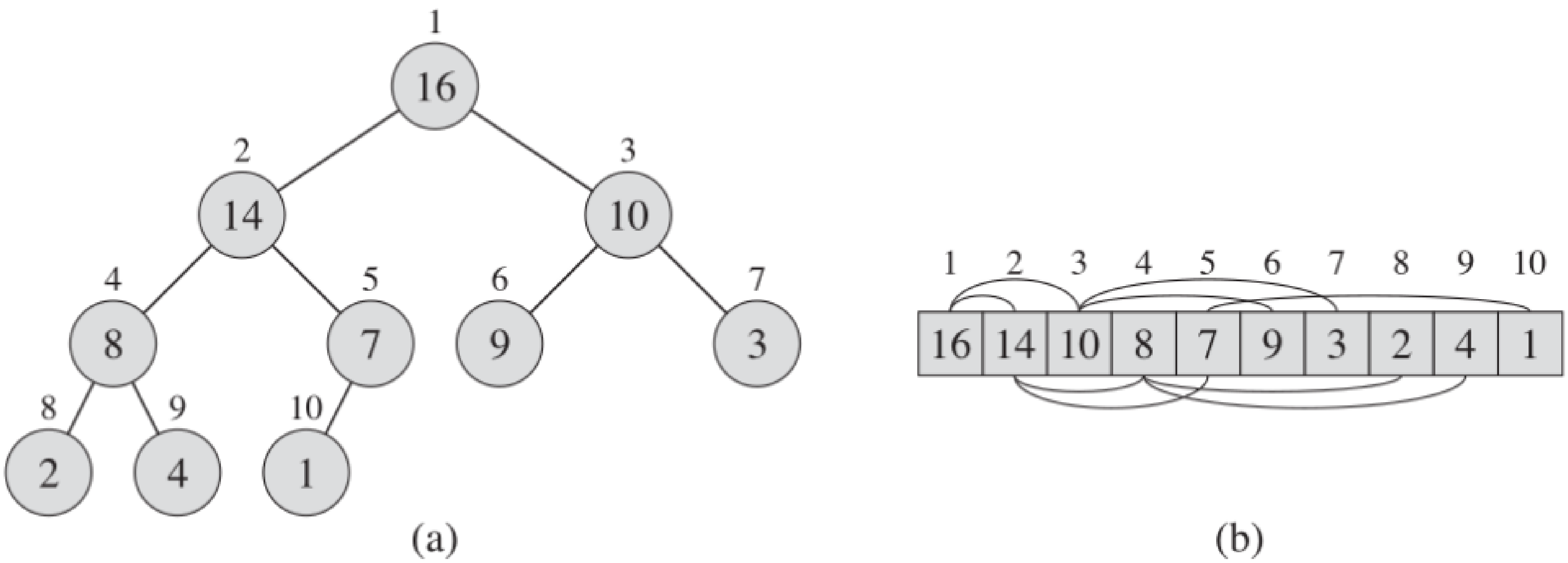
- Like insertion sort, it sorts **in place**. No auxiliary memory space is needed. So, space complexity is O(1).

- It achieves this using a specific data structure called a "heap".

The Heap Data Structure

- The (binary) heap data structure is an array object that can be viewed as a nearly complete binary tree.
  - Each node in the tree corresponds to an element in the array.
  - The tree is completely filled on all levels, except possibly the lowest level, which is filled from the left.
  - The array object, A, has two attributes: A.length (the number of elements in the array) and A.heap-size (how many elements are currently valid elements of the heap).
  - The root of the tree is A[1].

The Heap Data Structure



A max-heap viewed as (a) a binary tree and (b) an array.

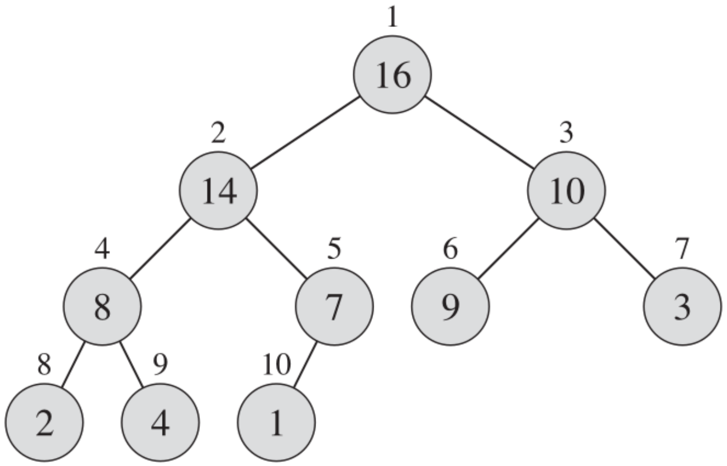
The Heap Data Structure

- Given the index i of a node, we can compute the indices of its parent and children:

PARENT(i)
1 **return** $\lfloor i/2 \rfloor$

LEFT(i)
1 **return** $2i$

RIGHT(i)
1 **return** $2i + 1$



- A heap of n elements has a height of Θ(log n).

The Heap Property

- There are two kinds of binary heaps: max-heaps and min-heaps. In both cases, the values in the nodes must satisfy a specific "heap property".
  - Max-Heaps
    - In a max-heap, the value of a node is at most the value of its parent. This is the max-heap property: $A[PARENT(i)] \geq A[i]$ for every node i other than the root.
    - This means the largest element in a max-heap is stored at the root ($A[1]$). The heapsort algorithm uses max-heaps.

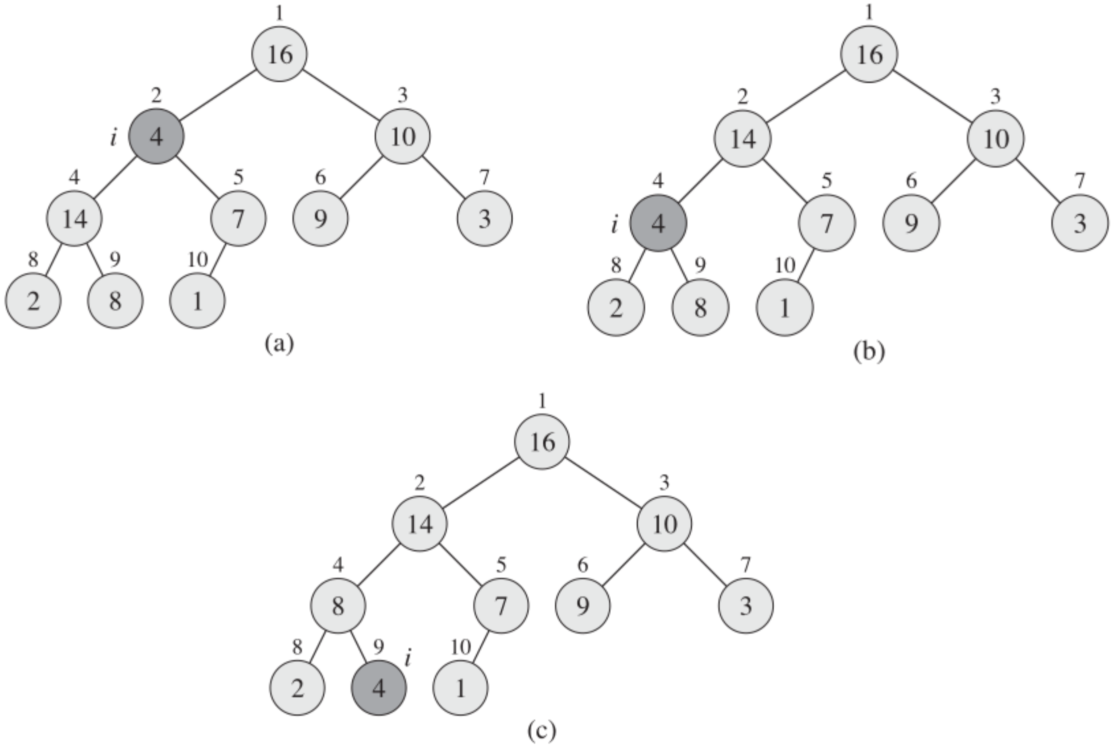
The Heap Property

- There are two kinds of binary heaps: max-heaps and min-heaps. In both cases, the values in the nodes must satisfy a specific "heap property".
  - Min-Heaps
    - A min-heap is organized in the opposite way. The min-heap property is: A[PARENT(i)] ≤ A[i] for every node i other than the root.
    - In a min-heap, the smallest element is at the root. Min-heaps are commonly used to implement priority queues.

Maintaining the Property: MAX-HEAPIFY

- To maintain the max-heap property, we use the MAX-HEAPIFY procedure.
  - Assumption: When MAX-HEAPIFY(A, i) is called, it assumes the binary trees rooted at LEFT(i) and RIGHT(i) are already max-heaps.
  - Problem: A[i] might be smaller than its children, violating the property.
  - Action: MAX-HEAPIFY lets the value at A[i] "float down" in the max-heap so that the subtree rooted at index i obeys the max-heap property.

Maintaining the Property: 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)
```

The action of MAX-HEAPIFY(A, 2), where A.heap-size = 10.

Maintaining the Property: MAX-HEAPIFY

- This procedure works by finding the largest of the elements A[i], A[LEFT(i)], and A[RIGHT(i)].

- If A[i] is not the largest, it is swapped with the largest child. This swap may cause the subtree rooted at the child (now at index largest) to violate the max-heap property, so MAX-HEAPIFY is called recursively on that subtree.

- The running time of MAX-HEAPIFY on a node of height h is O(h), or O(log n) on a heap of size n.

Building a Heap

- We can build a max-heap from an unordered input array A[1..n] by using the MAX-HEAPIFY procedure in a bottom-up manner.

- The elements in the subarray $A[(\lfloor n/2 \rfloor + 1) .. n]$ are all leaves of the tree, so they are already trivial 1-element heaps.

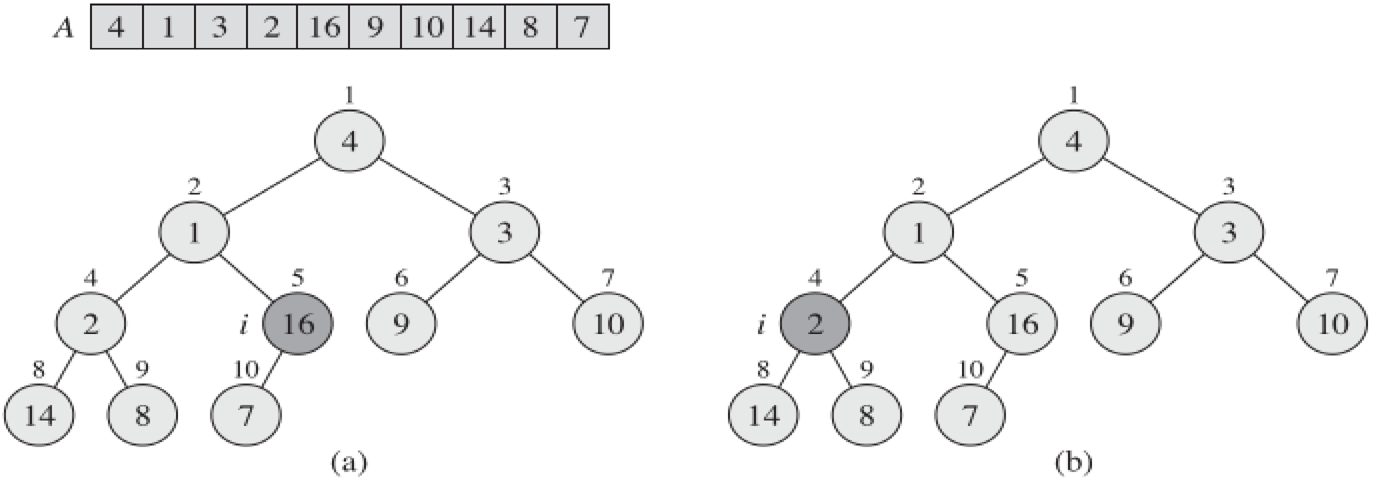
- The BUILD-MAX-HEAP procedure goes through the remaining (non-leaf) nodes and runs MAX-HEAPIFY on each one.

Building a Heap

- This works because of a key loop invariant:
  - At the start of each iteration of the for loop (for index i), each node i+1, i+2, ..., n is the root of a max-heap.
- When MAX-HEAPIFY(A, i) is called, the loop invariant ensures that the children of node i (which are numbered higher than i) are already roots of max-heaps, which is the condition required for MAX-HEAPIFY to work correctly.
- At termination, i=0, and node 1 is the root of a max-heap for the entire array.

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

Building a Heap

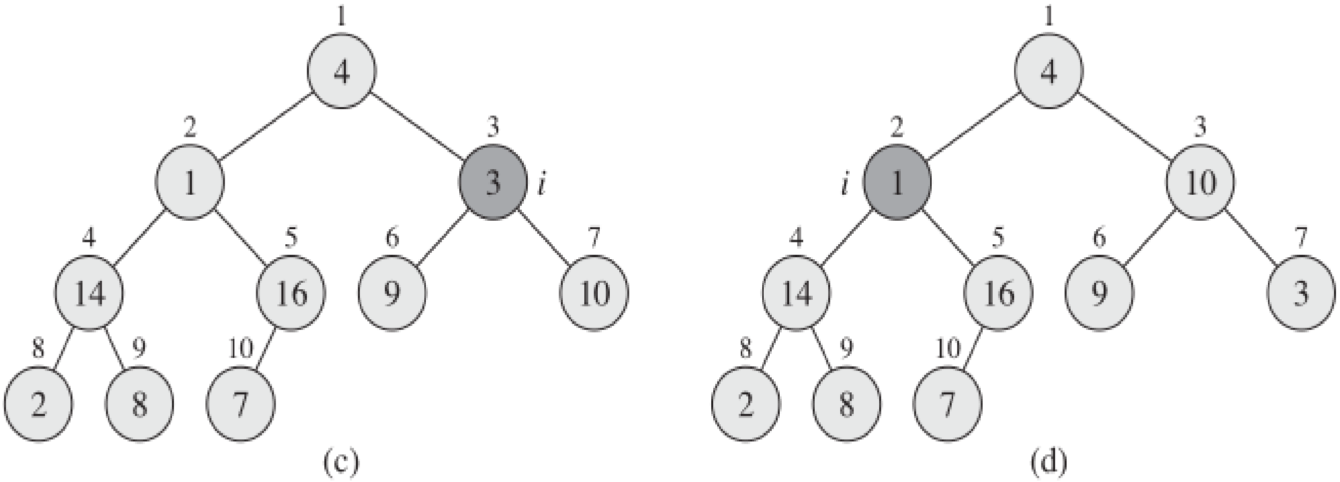


The operation of BUILD-MAX-HEAP, showing the data structure before the call to MAX-HEAPIFY in line 3 of BUILD-MAX-HEAP.

(a) A 10-element input array A and the binary tree it represents. The figure shows that the loop index i refers to node 5 before the call MAX-HEAPIFY(A, i).

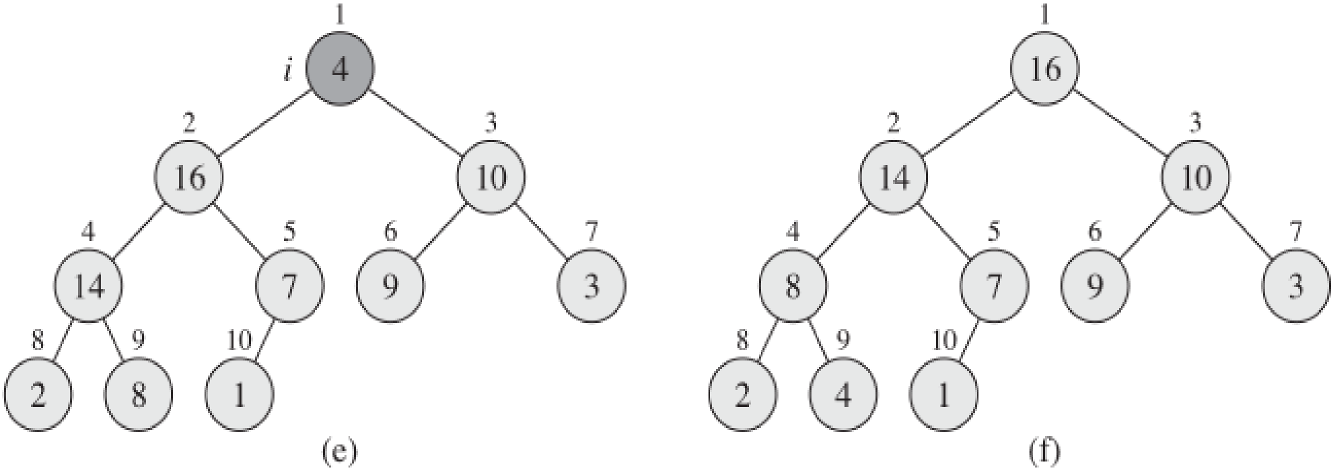
(b) The data structure that results.

Building a Heap



(c)–(d) Subsequent iterations of the for loop in BUILD-MAX-HEAP.

Building a Heap



(e) Subsequent iteration of the for loop in BUILD-MAX-HEAP.

(f) The max-heap after BUILD-MAX-HEAP finishes.

Building a Heap

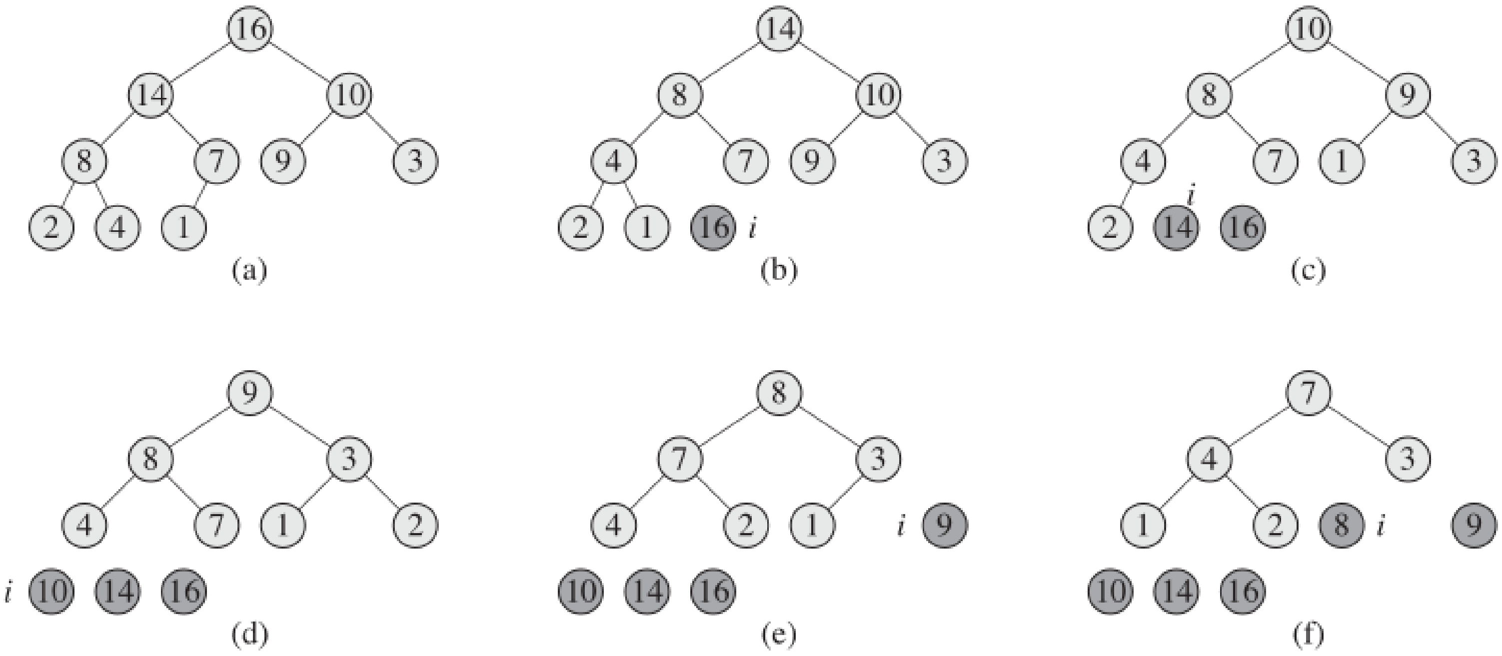
- We can compute a simple upper bound on the running time of BUILD-MAX-HEAP as follows:
  - Each call to MAX-HEAPIFY costs O(log n) time, and BUILD-MAX-HEAP makes O(n) such calls.
  - Thus, the running time is O(n log n). However, this bound is not asymptotically tight.
- A tighter analysis shows that the time required by MAX-HEAPIFY varies with the height of the node, and most nodes have small heights (A max-heap of 1073741823 elements has a very small height of 29 only!).
  - So, the number of levels log n can be omitted as a lower order term.
  - Thus, we can bound the running time of BUILD-MAX-HEAP as O(n).
  - Hence, we can build a max-heap from an unordered array in linear time.

The Heapsort Algorithm

The heapsort algorithm combines BUILD-MAX-HEAP and MAX-HEAPIFY to sort an array.

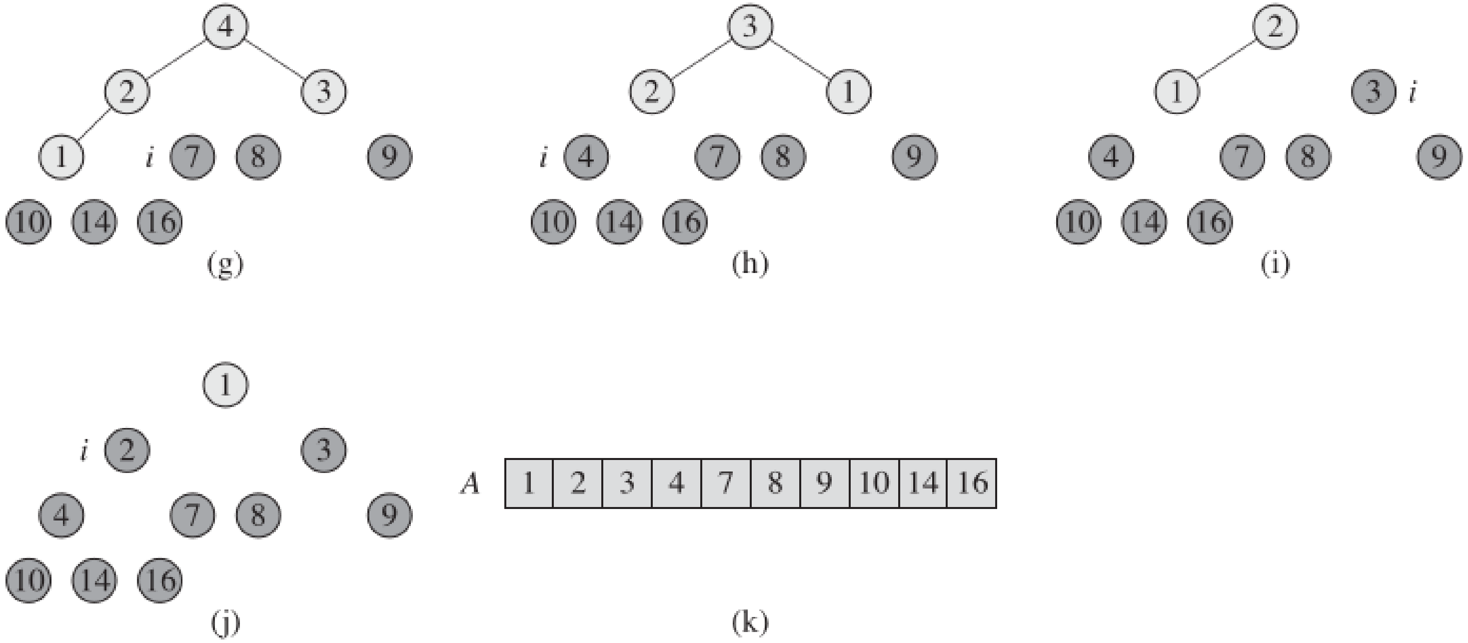
1. It starts by using BUILD-MAX-HEAP to build a max-heap on the input array A[1..n].
2. Since the maximum element of the array is now at the root A[1], it is exchanged with A[n], placing the largest element in its correct final position.
3. The heap size is reduced by one, "discarding" node n from the heap.
4. The new root element might violate the max-heap property. To restore it, MAX-HEAPIFY(A, 1) is called, which leaves a max-heap in A[1..n-1].
5. This process (steps 2-4) is repeated for the max-heap of size n-1 down to a heap of size 2.

The Heapsort Algorithm



The operation of HEAPSORT

The Heapsort Algorithm



The operation of HEAPSORT

The Heapsort Algorithm

- Running Time: The HEAPSORT procedure takes O(n log n) time.
  - The initial call to BUILD-MAX-HEAP takes O(n) time.
  - There are n-1 calls to MAX-HEAPIFY (one for each iteration of the loop).
  - Each call to MAX-HEAPIFY takes O(log n) time.
  - Thus, the total time is O(n) + O((n-1) log n), which is O(n log n).

```
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)
```

Reference

- *Introduction to Algorithms, Third Edition*
    - *Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein*