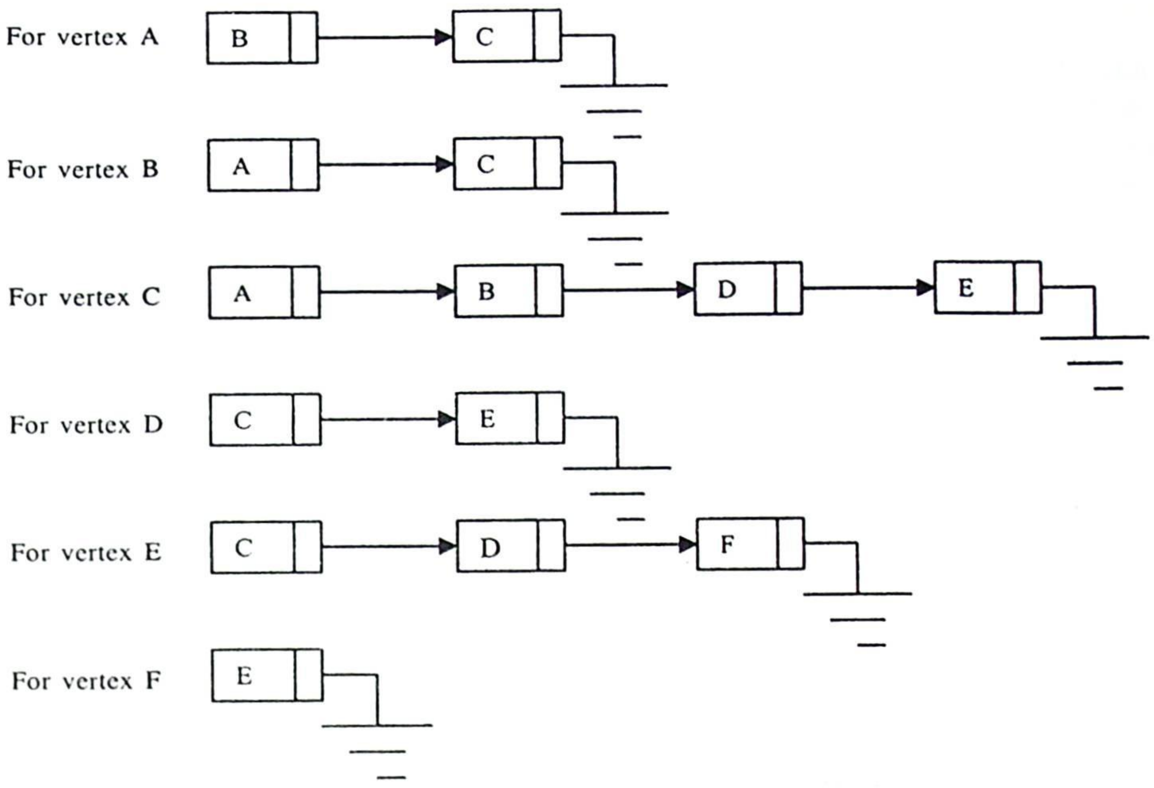

FIGURE 1.7(b) Adjacency lists.

	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	1	0	0	0
C	1	1	0	1	1	0
D	0	0	1	0	1	0
E	0	0	1	1	0	1
F	0	0	0	0	1	0

FIGURE 1.7(c) Adjacency matrix.

binary, then each vertex has at most two children: the left child and the right child. Leaf vertices have no children. The root vertex has no parent; all other vertices have unique parents. A binary tree may be represented using an array or a linked list.

The binary tree may be represented in the form of an array as shown in Figure 1.8(b).

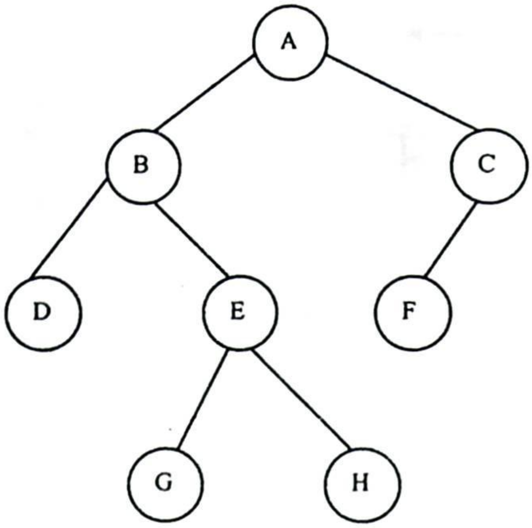

FIGURE 1.8(a) Binary tree.

A	B	C	D	E	F				G	H
1	2	3	4	5	6	7	8	9	10	11

FIGURE 1.8(b) Array representation of binary tree.

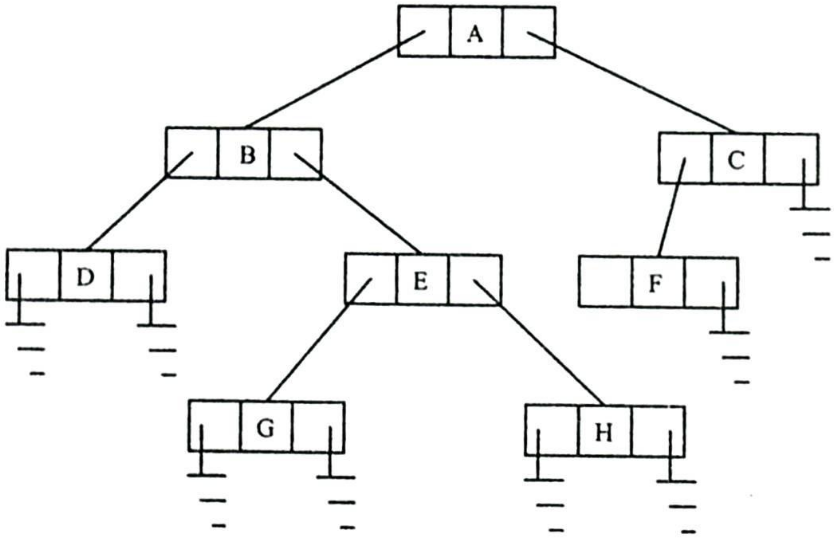

FIGURE 1.8(c) Linked list representation of binary tree.

The left-child of vertex (i) is stored in the location $2i$, if it exists and the right child is stored in the location $2i + 1$, if it exists. The parent of vertex i is stored in location $\lfloor i/2 \rfloor$. The above tree may be represented in the form of a linked list as shown in Figure 1.8(c).

1.8.8 Priority Queue (Heap)

A priority queue is a data structure for maintaining a set S of elements, each with an associated value called a *key*. A priority queue supports the operations: insertion of an element into the set S, searching maximum (minimum) element in the set S, and deletion of maximum (minimum) element from the set S. Priority queues are used in many applications such as job scheduling on a shared computer, event-driven simulation, etc. The name 'priority queue' comes from the fact that the keys determine the 'priority' used to pick elements to be removed from the set S. We can use a heap to implement a priority queue.

A max (min) heap is a complete binary tree with the property that the key value at each node is at least as large (small) as the values at its children. A binary tree with height h is called *complete* if the levels 0, 1, 2, ..., $h - 1$ have the maximum number of nodes possible and in level h all the nodes are towards left. Figure 1.9(a) shows a min-heap.

The important operations involving heaps are creation of heap, deletion from a heap and addition to a heap. Creation of a heap with n elements requires $O(n)$ time. Addition and deletion operations each requires $O(\log n)$ time. We illustrate in Figure 1.9(b) how a max-heap is created for the elements {5, 1, 13, 14, 7, 19}.

1.9 PSEUDOCODE CONVENTION

Several conventions are used for the description of algorithms. These are mixtures of natural and programming language-like constructs.

In this book, we follow the following conventions for the description of algorithms:

(i) An algorithm is described within a pair of lines:

```
Procedure Name
...
...
End Name
```

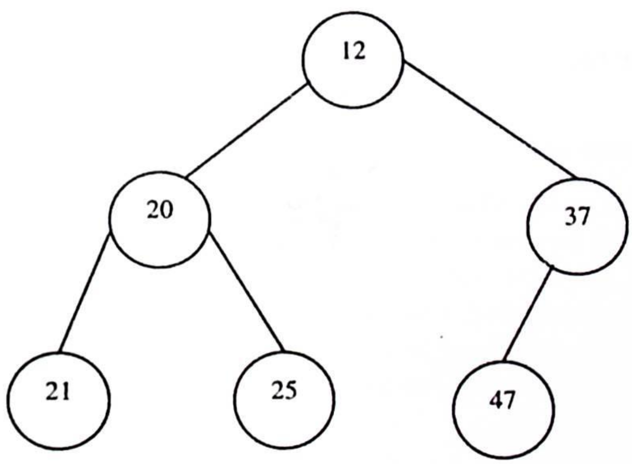

FIGURE 1.9(a) Heap.

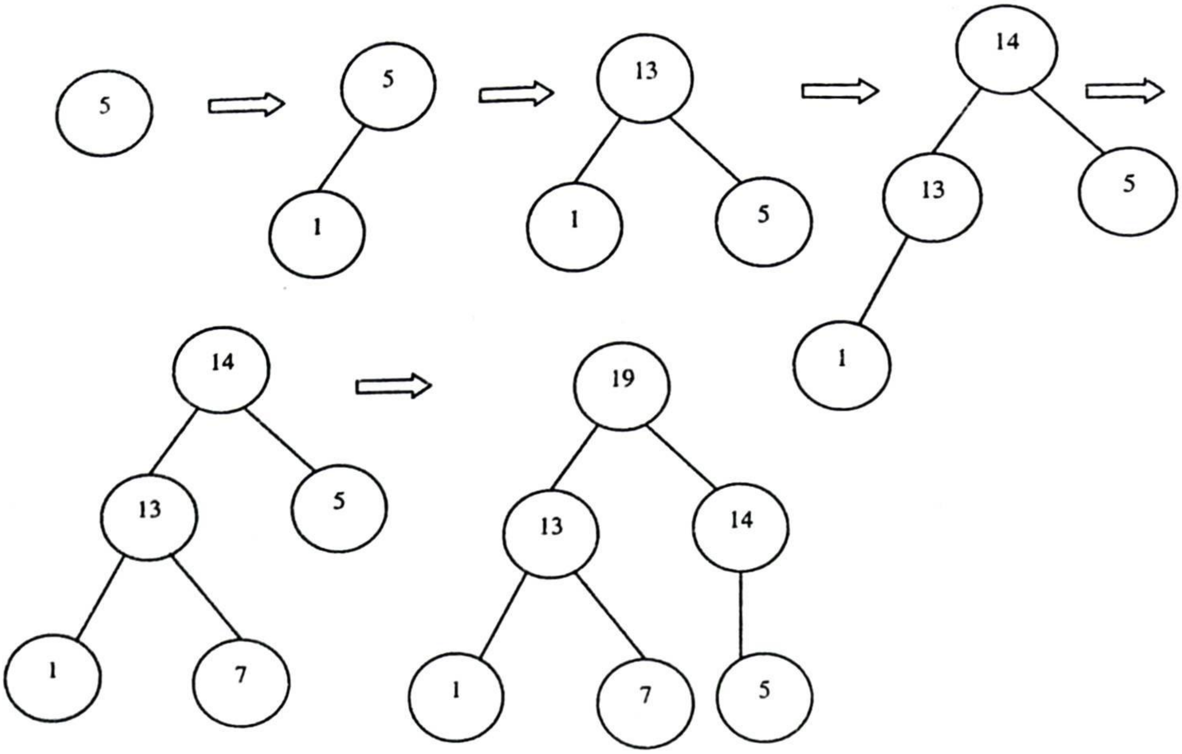

FIGURE 1.9(b) Heap creation.

(ii) Assignment is denoted as:

```
Variable_name = Expression;
```

(iii) *If statement* is shown as:

```
If Condition Then
  Statement(s);
  Else
  Statement(s);
Endif
```

(iv) *Do loop* is shown as:

```
For Variable = Initial Value to Final Value {Step Step-size} Do
...
...
Endfor
```

(v) *Repeat construct* is shown as:

```
Repeat
...
...
Until condition
```