# UNIT 3- Heaps

- Binomial Heap
- Operations of Binomial Heap

# Binomial Heap

- **Classical, Complicated to achieve all heap operations in O(log n) time**
- **Invented by Vuillemin(1978)**
- **Interested in one of the additional operation, the change of key values**

- **Identify the element that we want to change.**
- **Binomial Heap: set of Binomial trees**

# Binomial Tree

A *binomial tree* is an ordered tree defined recursively. Figure 12.28 shows the binomial trees.

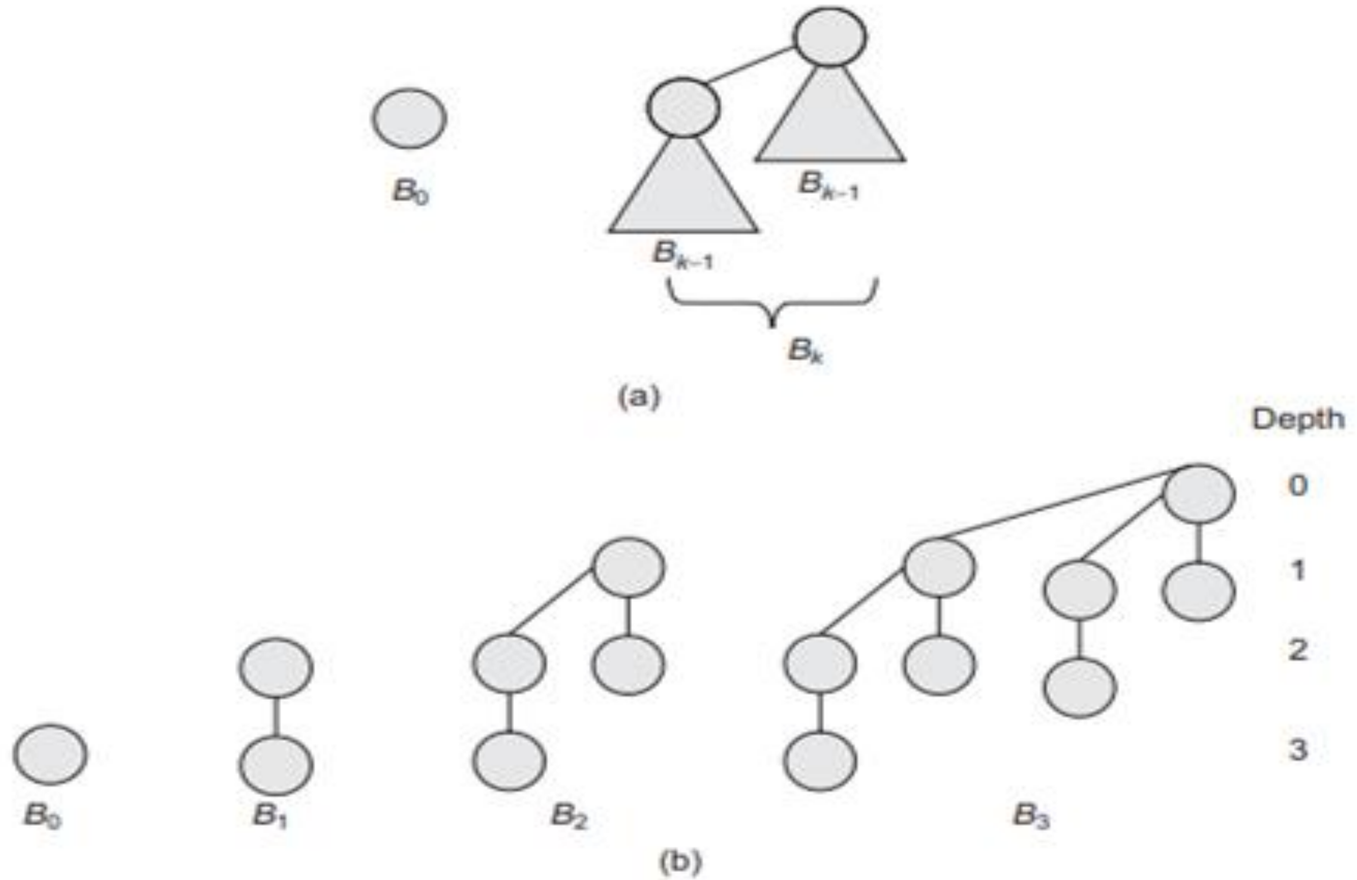



**Fig. 12.28** Binomial trees (a) Recursive definition of the binomial tree $B_k$
(b) Binomial tree $B_0$ through $B_3$

# Binomial Tree

**For the binomial tree Bk ,**

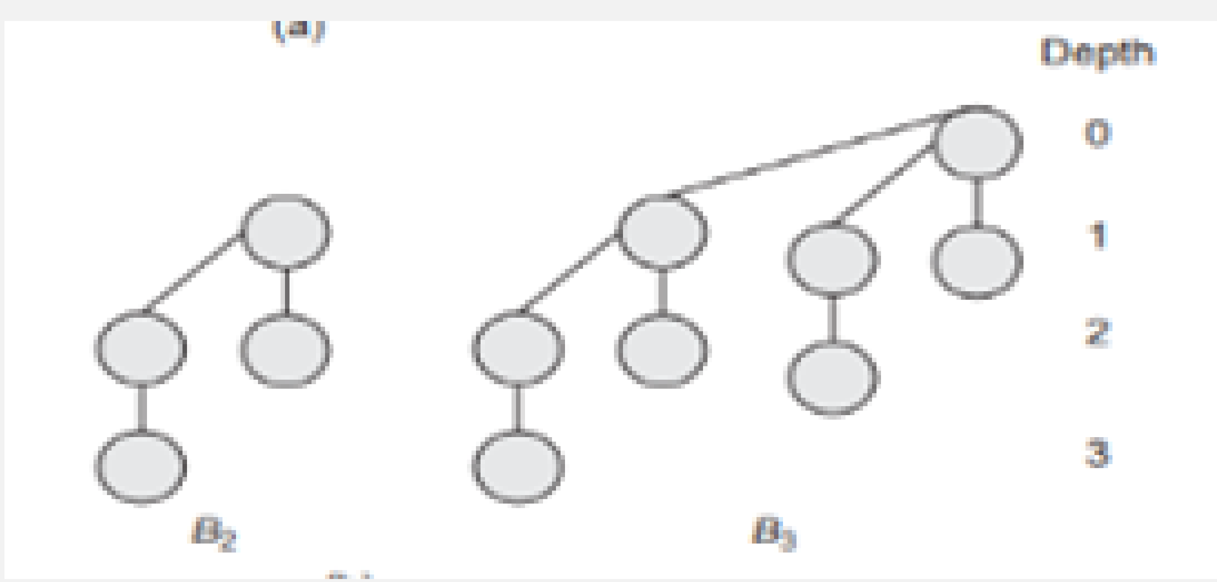
**1. There are 2^k nodes**

**2. The height of the tree is k**

**3. The root has degree k, which is greater than that of any other node; moreover,**

**if the children of the root are numbered from left to right by k - 1, k - 2, …, 0,**

**the child i is the root of a subtree**



# Binomial Tree

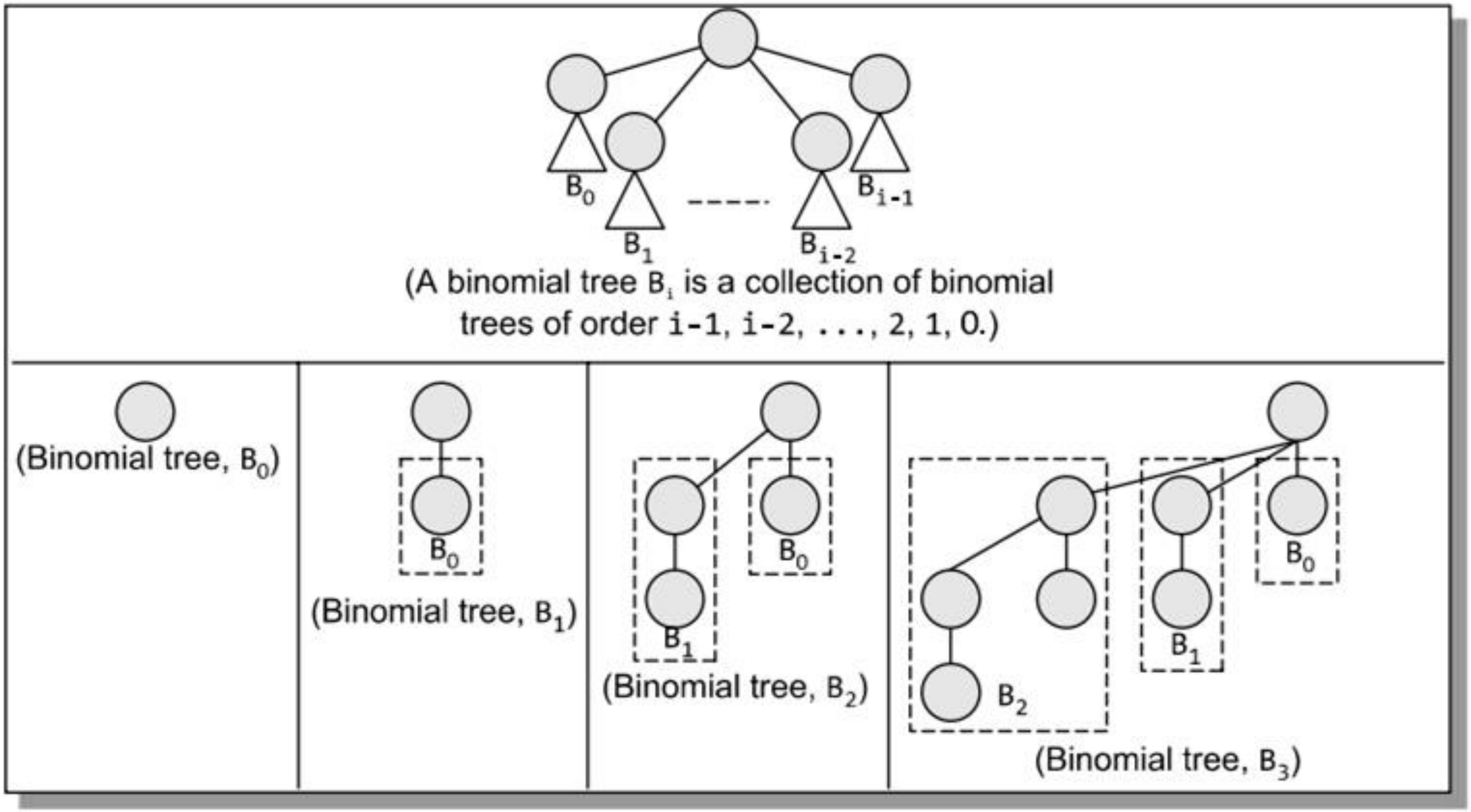



**Figure 12.12** Binomial trees

**Binomial Heap Property:**

**1. Each binomial tree in H follows the min-heap property. Each tree is min-heap ordered.**

**2. For any positive integer k, there is utmost one binomial tree in H whose root has degree k**

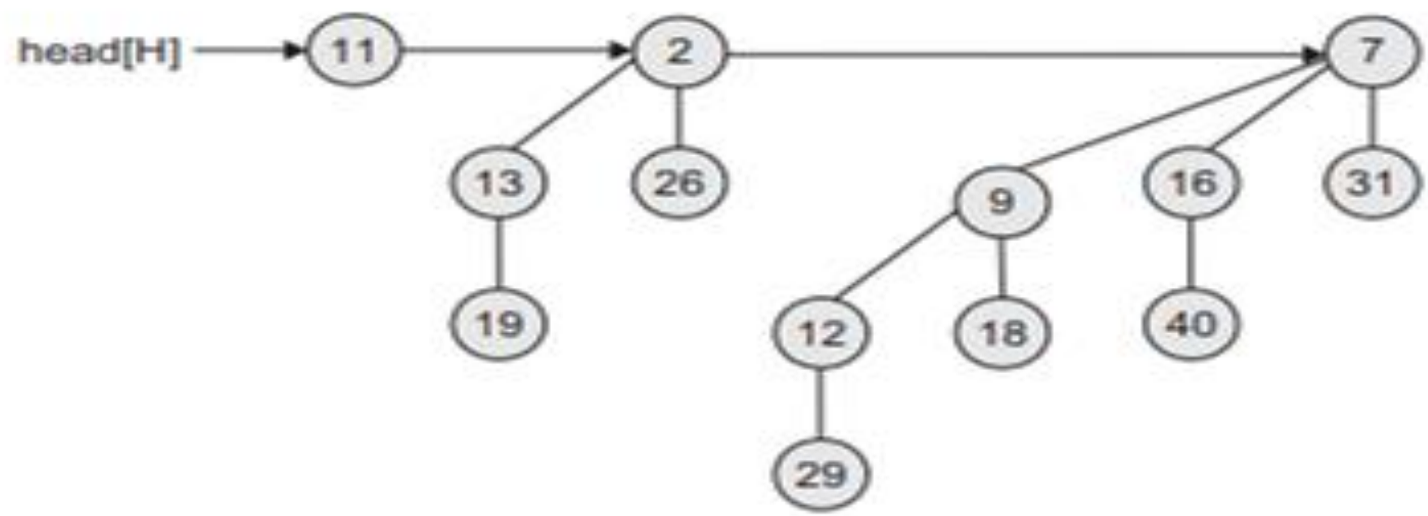



Fig. 12.29 A binomial heap with 13 nodes

# Representation of Binomial Heap

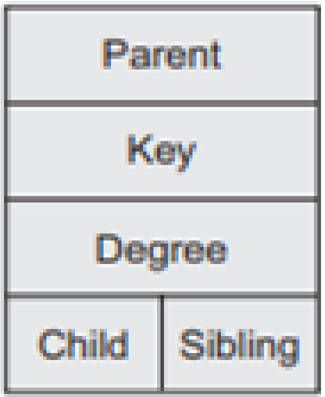



**Fig. 12.30** Representation of a node of binomial heap

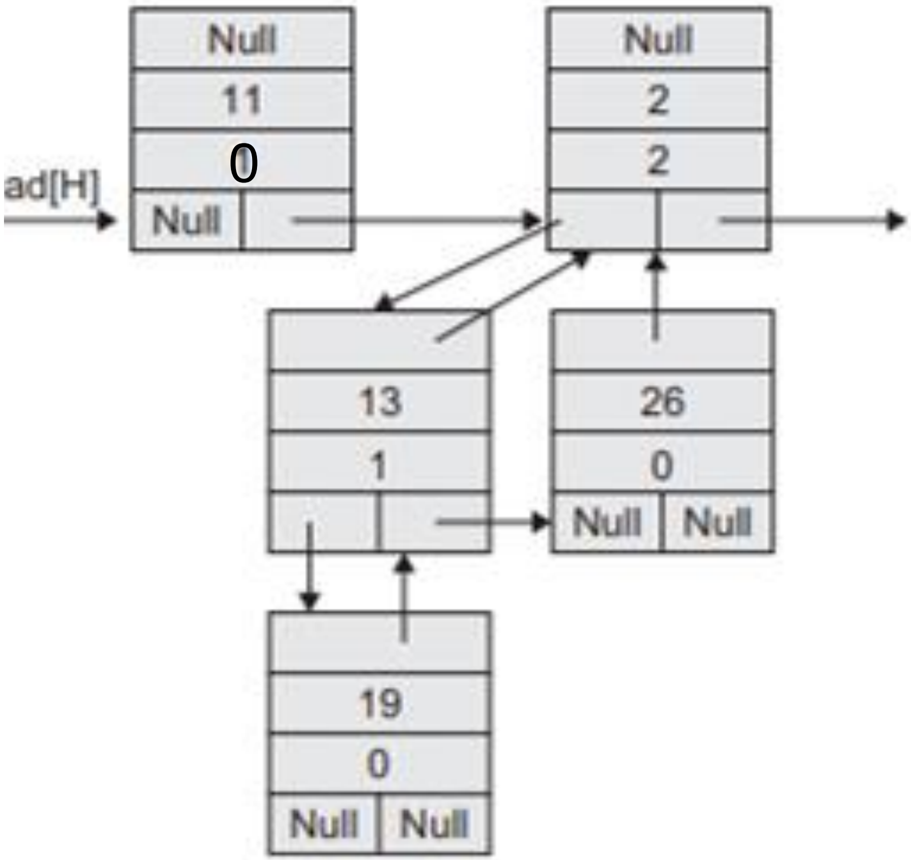



**Fig. 12.31** Representation of binomial heap of Fig. 12.29 using five-tuple node

## Representation of Binomial Heap

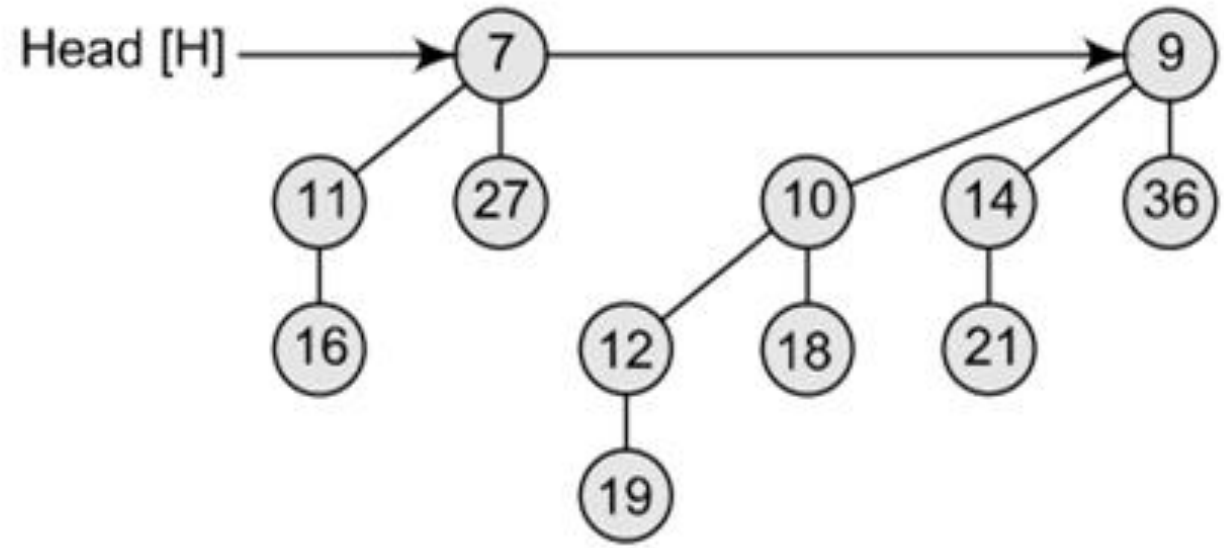



**Figure 12.13** Binomial heap

# Representation of Binomial Heap

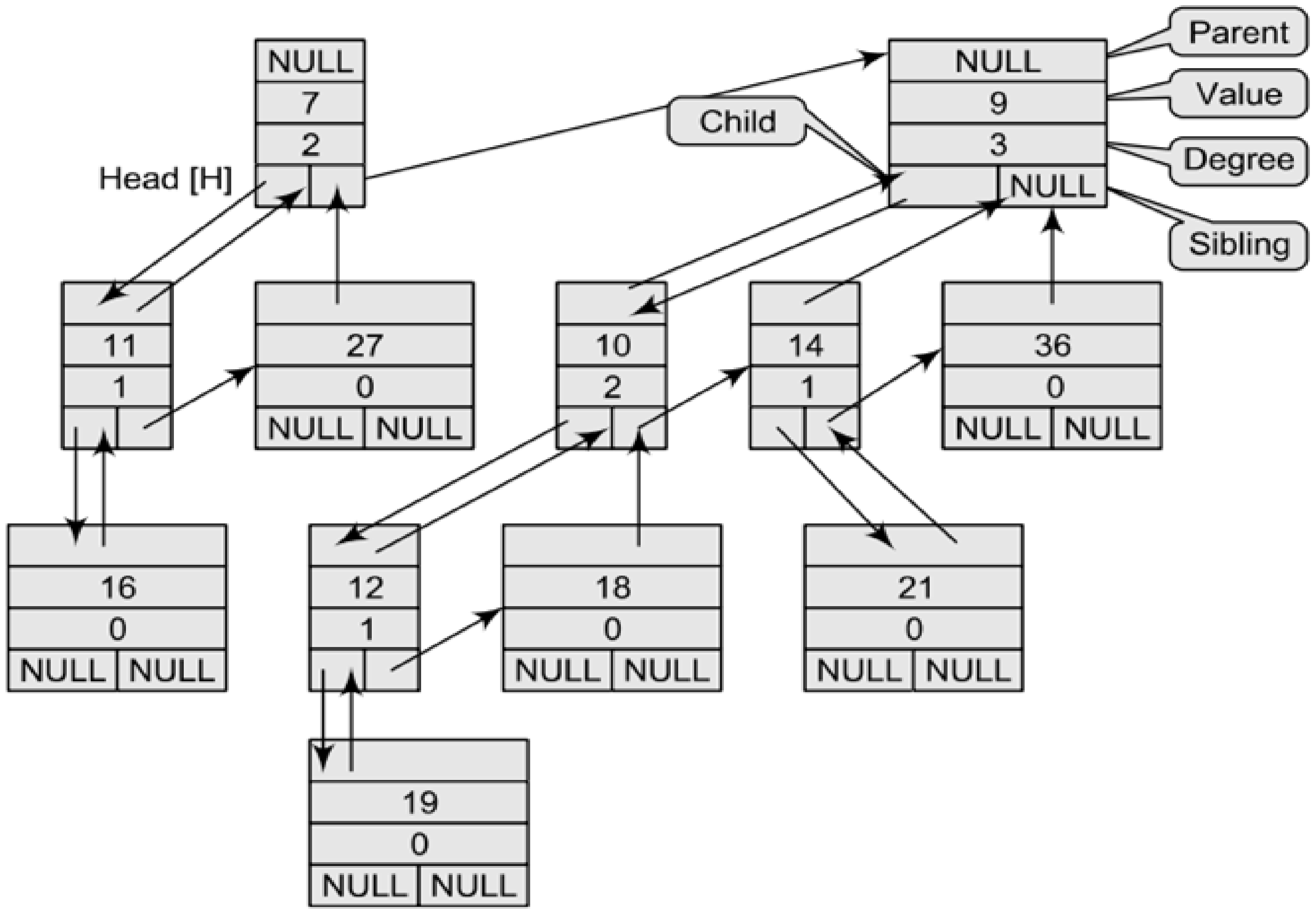



**Figure 12.14** Linked representation of the binomial tree shown in Fig. 12.13

# Operations on Binomial Heaps

1. **CreateBHeap— simply allocates and returns an object H, where head[H] = null. O(1)time**
2. **FindMinimumKey—Returns a pointer to the node with the minimum key in an n-node binomial heap H.**
3. **UnitingTwoBHeap—Takes the union of the two binomial heaps.**
4. **InsertNode—Inserts a node into binomial heap H.**
5. **ExtractMinimumKeyNode—returns the pointer to the extracted node.**
6. **DecreaseKey—Decreases the key of a node in a binomial heap H to a new value k.**
7. **DeleteKey—Deletes the specified key from binomial heap H.**

# Operations on Binomial Heaps

1. **FindMinimumKey: Returns a pointer to the node with the minimum key in an n-node binomial heap H.**
2. **There are atmost log(n+1) roots to check, O(log n) time**

```
Min_Binomial-Heap(H)

Step 1: [INITIALIZATION] SET Y = NULL, X = Head[H] and Min = ∞
Step 2: REPEAT Steps 3 and 4 While X ≠ NULL
Step 3:      IF Val[X] < Min
                   SET Min = Val[X]
                   SET Y = X
             [END OF IF]
Step 4:      SET X = Sibling[X]
        [END OF LOOP]
Step 5: RETURN Y
```

**Figure 12.15** Algorithm to find the node with minimum value

# Operations on Binomial Heaps

1. **FindMinimumKey: Returns a pointer to the node with the minimum key in an n-node binomial heap H.**
2. **There are atmost log(n+1) roots to check, O(log n) time**

**Example 12.4** Consider the binomial heap given below and see how the procedure works in this case.

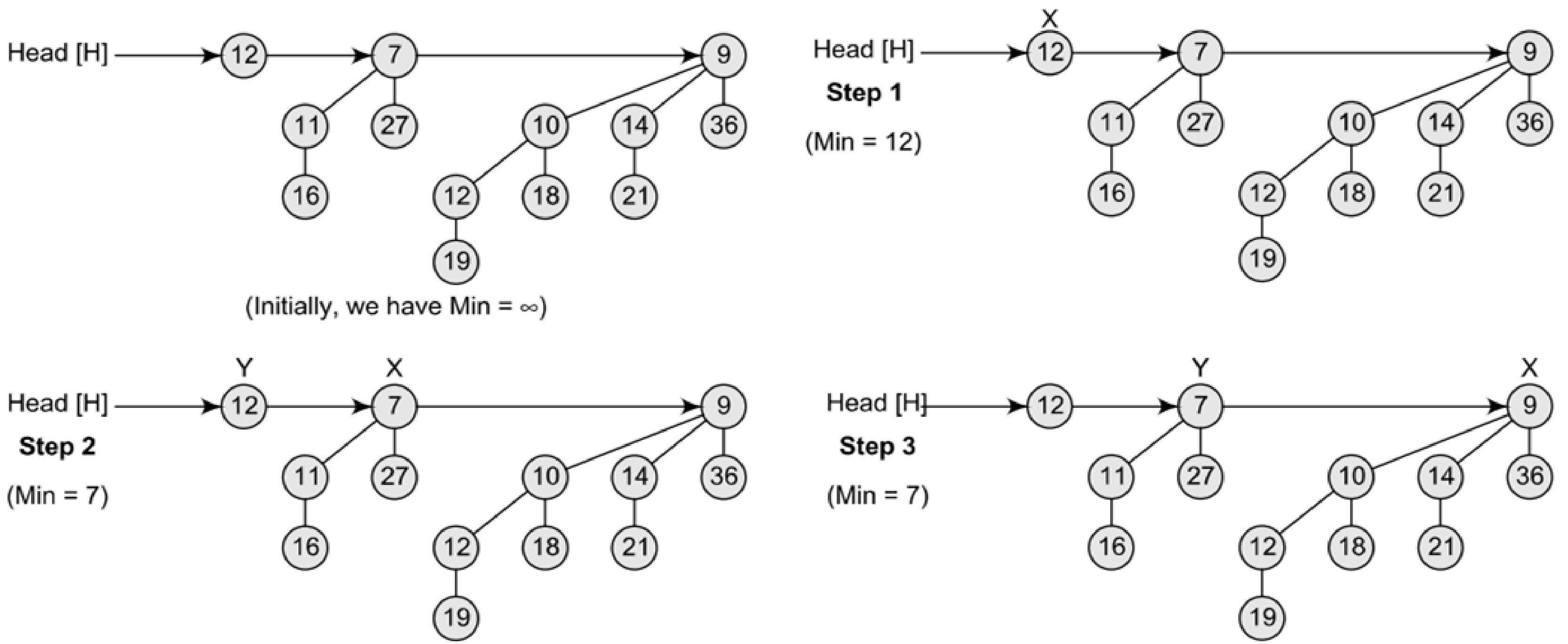



Figure 12.16 Binomial heap

**UnitingTwoBHeap—Takes the union of the two binomial heaps.**

**Example 12.5** Unite the binomial heaps given below.

***Solution***

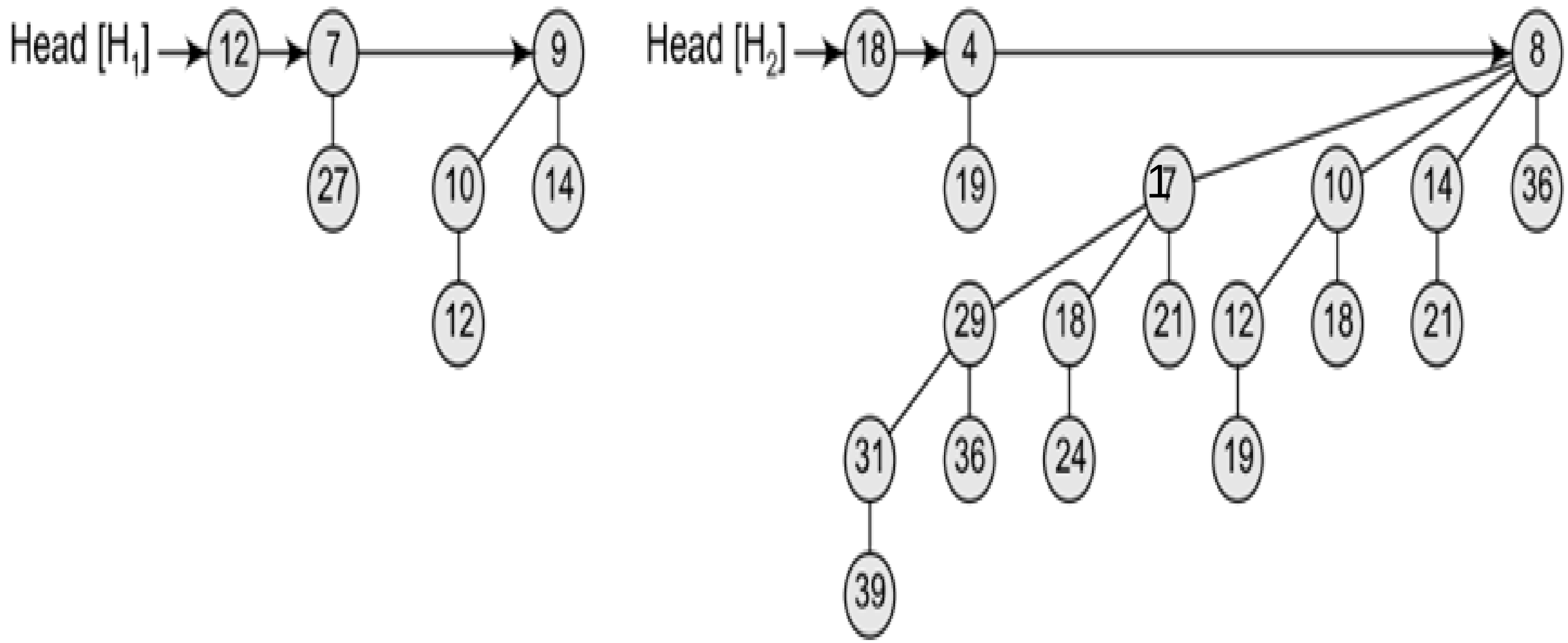



Figure 12.10(a)

# Operations on Binomial Heaps

```
Union_Binomial-Heap(H1, H2)

Step 1: SET H = Create_Binomial-Heap()
Step 2: SET Head[H] = Merge_Binomial-Heap(H1, H2)
Step 3: Free the memory occupied by H1 and H2
Step 4: IF Head[H] = NULL, then RETURN H
Step 5: SET PREV = NULL, PTR = Head[H] and NEXT =
        Sibling[PTR]
Step 6: Repeat Step 7 while NEXT ≠ NULL
Step 7:     IF Degree[PTR] ≠ Degree[NEXT] OR
            (Sibling[NEXT] ≠ NULL AND
            Degree[Sibling[NEXT]] = Degree[PTR]), then
                  SET PREV = PTR, PTR = NEXT
            ELSE IF Val[PTR] ≤ Val[NEXT], then
                  SET Sibling[PTR] = Sibling[NEXT]
                  Link_Binomial-Tree(NEXT, PTR)
                  ELSE
                        IF PREV = NULL, then
                           Head[H] = NEXT
                        ELSE
                           Sibling[PREV] = NEXT
                          Link_Binomial-Tree(PTR, NEXT)
                           SET PTR = NEXT
            SET NEXT = Sibling[PTR]
Step 8: RETURN H
```

**Figure 12.18** Algorithm to unite two binomial heaps

# Operations on Binomial Heaps

**UnitingTwoBHeap—Takes the union of the two binomial heaps.**

After `Merge_Binomial-Heap()`, the resultant heap can be given as follows:

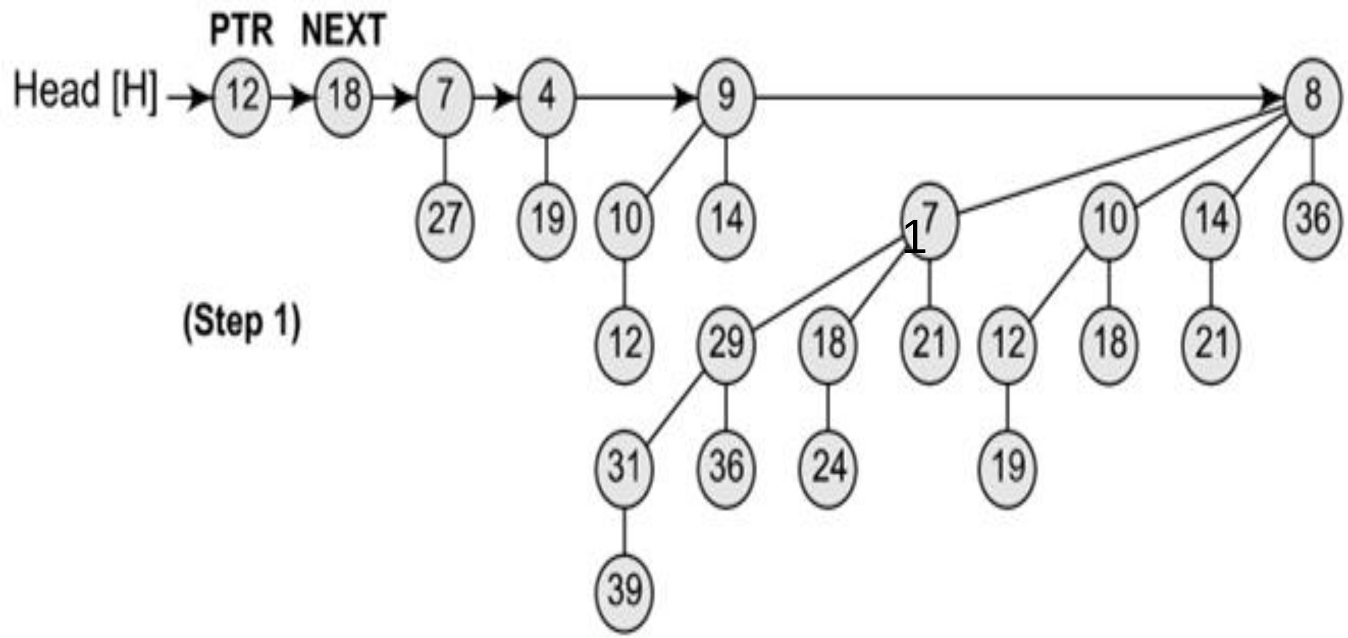



Figure 12.19(b)

# Operations on Binomial Heaps

**UnitingTwoBHeap—Takes the union of the two binomial heaps.**

Link NEXT to PTR, making PTR the parent of the node pointed by NEXT.

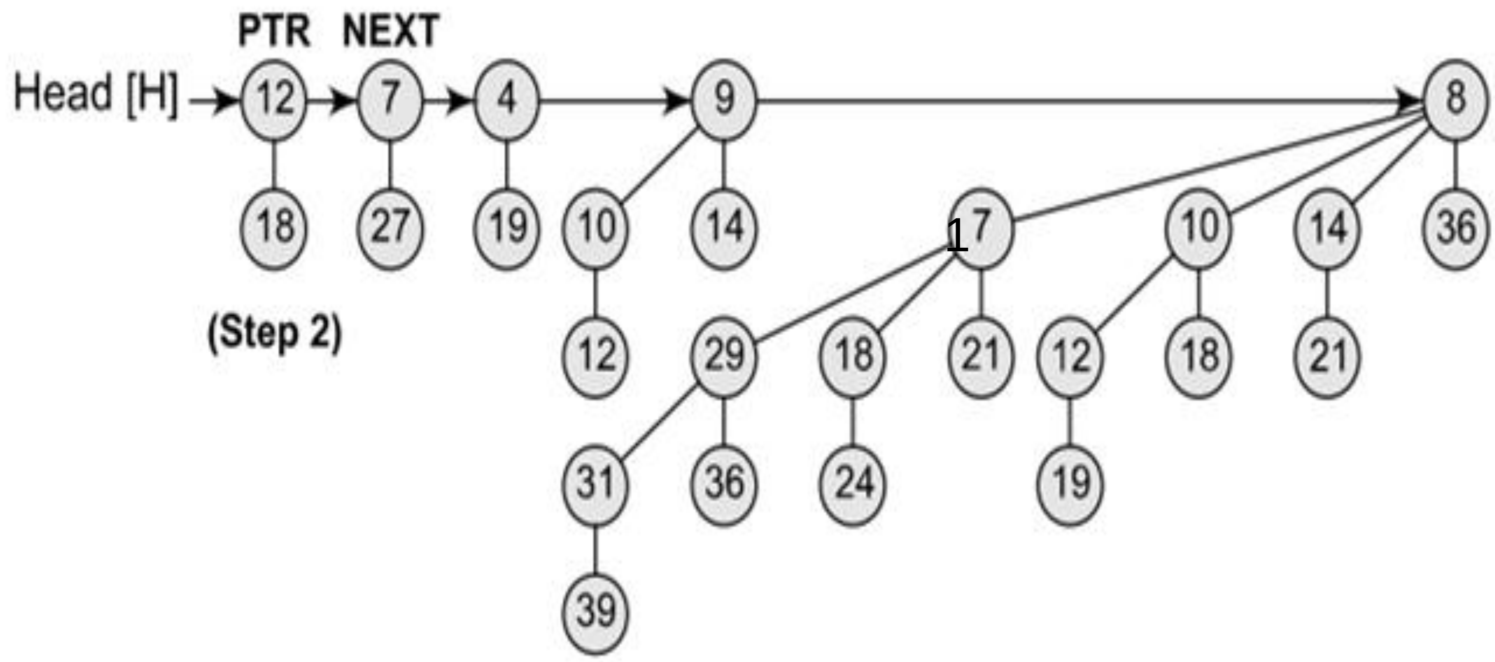



Figure 12.19(c)

**UnitingTwoBHeap—Takes the union of the two binomial heaps.**

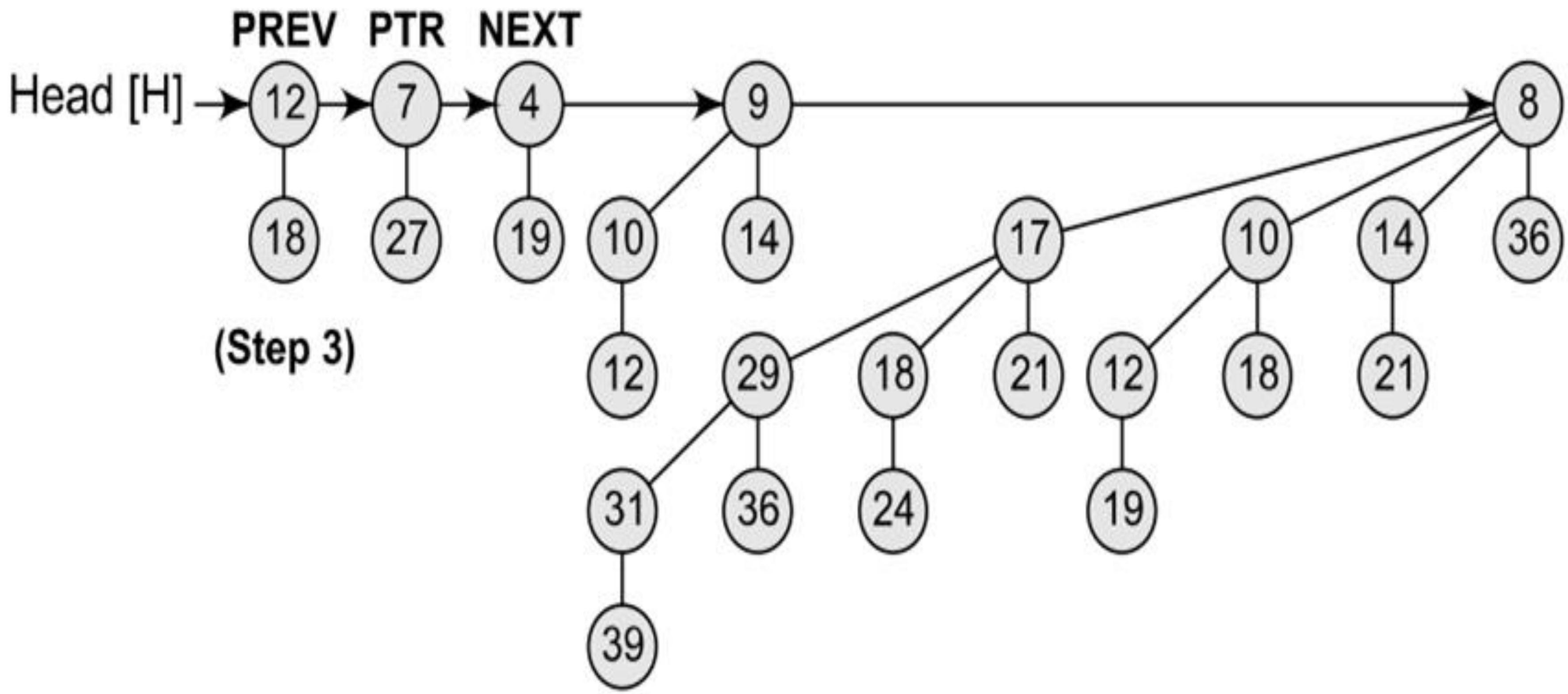



**Figure 12.19(d)**

**UnitingTwoBHeap—Takes the union of the two binomial heaps.**

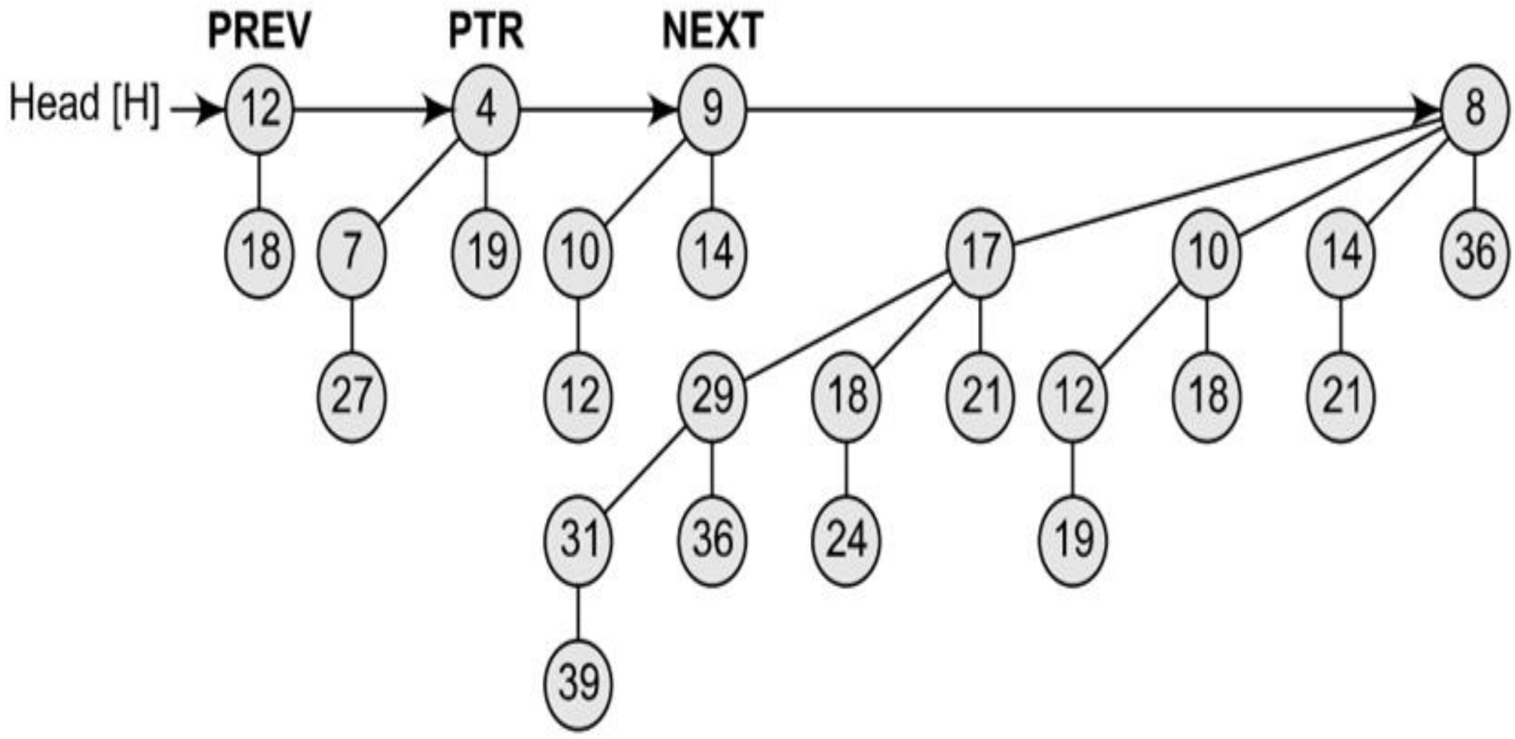



Figure 12.19(e)

# Operations on Binomial Heaps

**UnitingTwoBHeap—Takes the union of the two binomial heaps.**

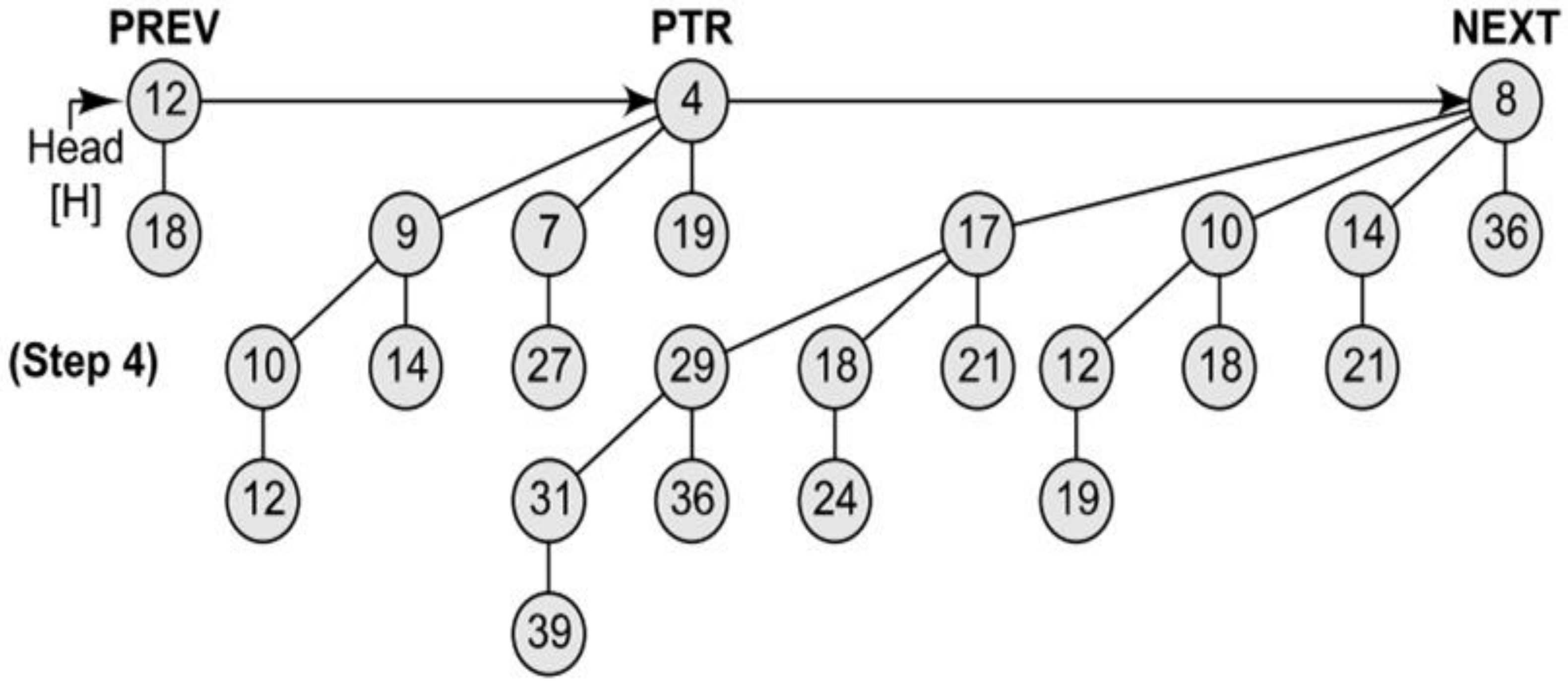



**Figure 12.19(f)** Binomial heap

# Operations on Binomial Heaps

**2. InsertNode—Inserts a node into binomial heap H.**

**To create H’ needs O(1) time. To unite H’ with H requires O(log n) time.**

```
Insert_Binomial-Heap(H, x)

Step 1: SET H' = Create_Binomial-Heap()
Step 2: SET Parent[x] = NULL, Child[x] = NULL and
        Sibling[x] = NULL, Degree[x] = NULL
Step 3: SET Head[H'] = x
Step 4: SET Head[H] = Union_Binomial-Heap(H, H')
Step 5: END
```

**Figure 12.20** Algorithm to insert a new element in a binomial heap

# Operations on Binomial Heaps

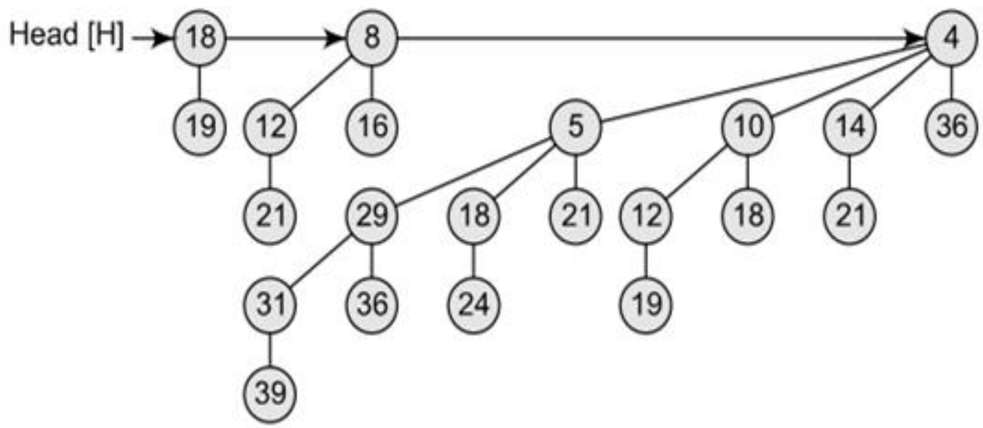
**ExtractMinimumKeyNode—returns the pointer to the extracted node.**

```
Min-Extract_Binomial Heap (H)

Step 1: Find the root R having minimum value in
        the root list of H
Step 2: Remove R from the root list of H
Step 3: SET H′ = Create_Binomial-Heap()
Step 4: Reverse the order of R's children thereby
        forming a linked list
Step 5: Set head[H′] to point to the head of the
        resulting list
Step 6: SET H = Union_Binomial-Heap(H, H′)
```

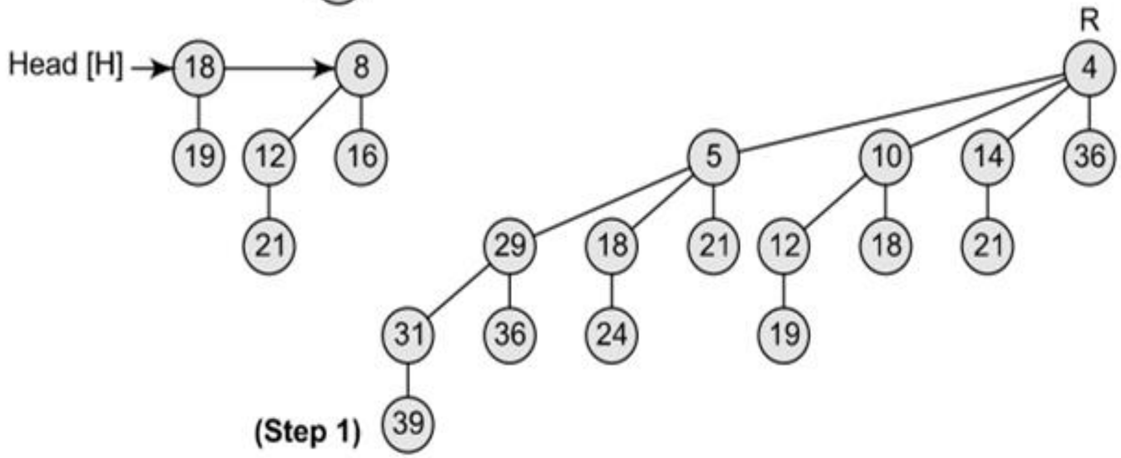
**Figure 12.21** Algorithm to extract the node with minimum key from a binomial heap

**ExtractMinimumKeyNode—returns the pointer to the extracted node.**

**Example 12.6** Extract the node with the minimum value from the given binomial heap.




***Solution***




**(Step 1)**

## Operations on Binomial Heaps

**ExtractMinimumKeyNode—returns the pointer to the extracted node.**

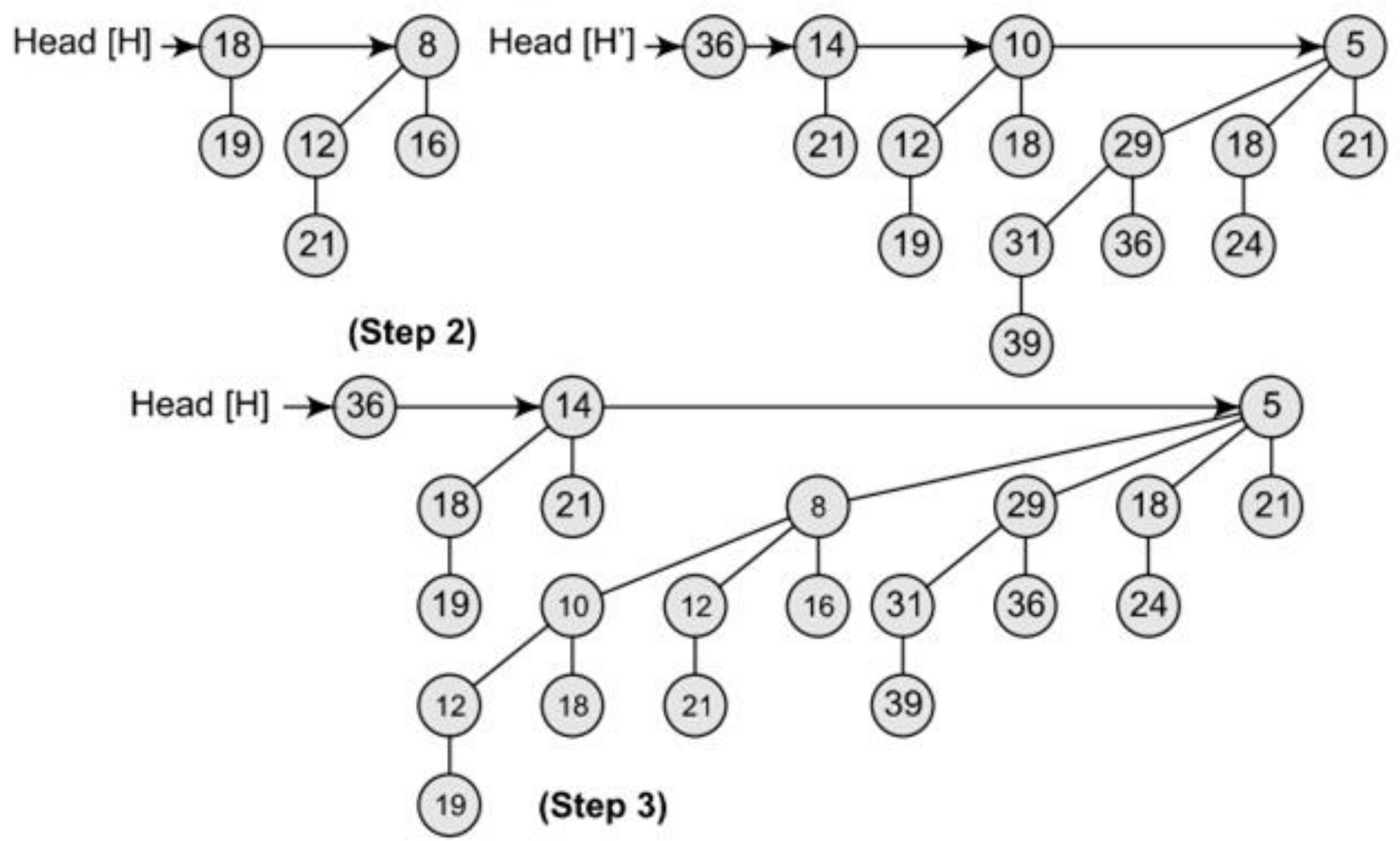



**Figure 12.22** Binomial heap

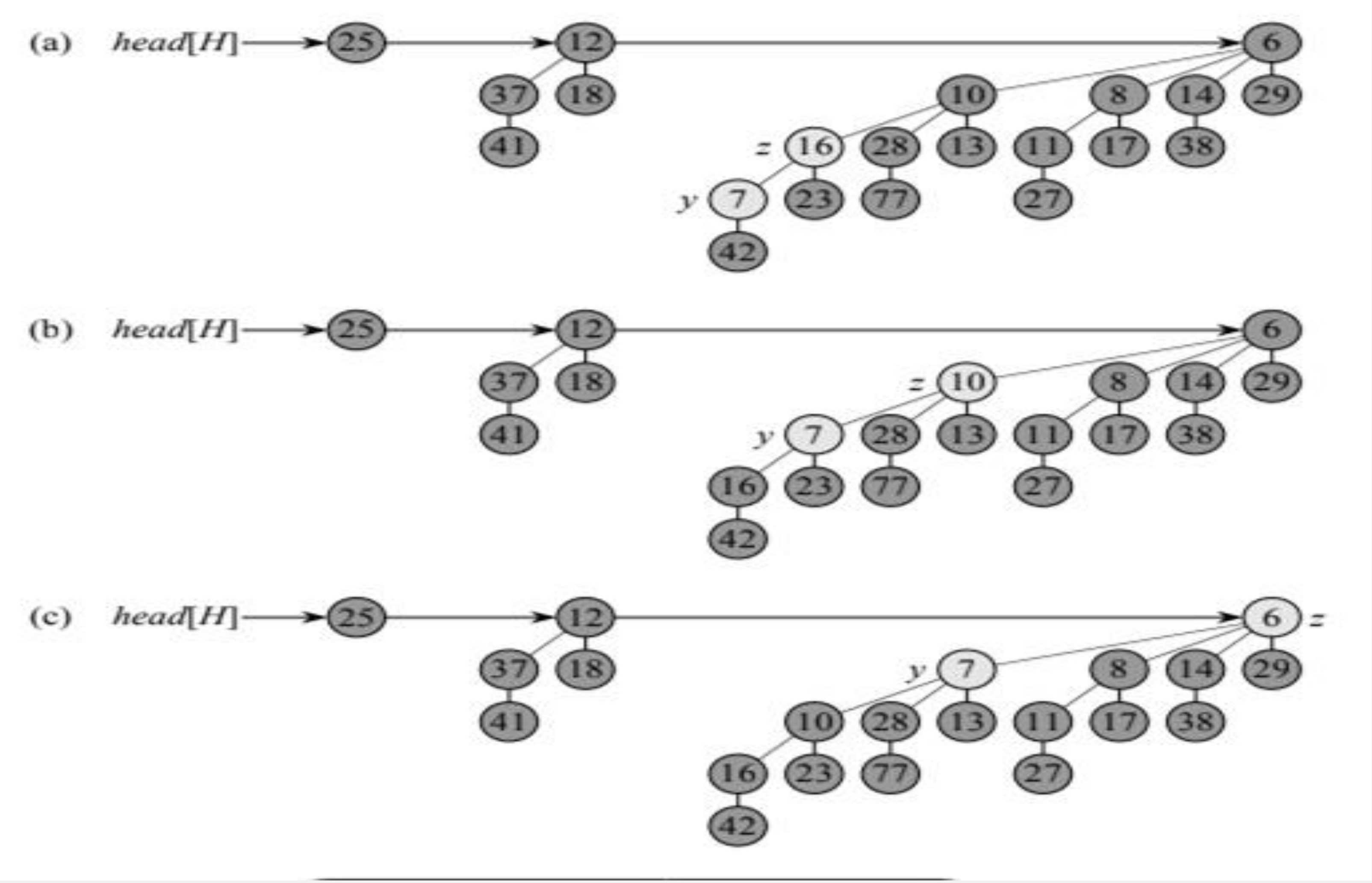
# Operations on Binomial Heaps

**DecreaseKey—Decreases the key of a node in a binomial heap H to a new value k.**

```
Binomial-Heap_Decrease_Val(H, x, k)

Step 1: IF Val[x] < k, then Print " ERROR"
Step 2: SET Val[x] = k
Step 3: SET PTR = x and PAR = Parent[PTR]
Step 4: Repeat while PAR ≠ NULL and Val[PTR] < Val[PAR]
Step 5:           SWAP ( Val[PTR], Val[PAR] )
Step 6:           SET PTR = PAR and PAR = Parent [PTR]
        [END OF LOOP]
Step 7: END
```

**Figure 12.23** Algorithm to decrease the value of a node x in a binomial heap H

## DecreaseKey



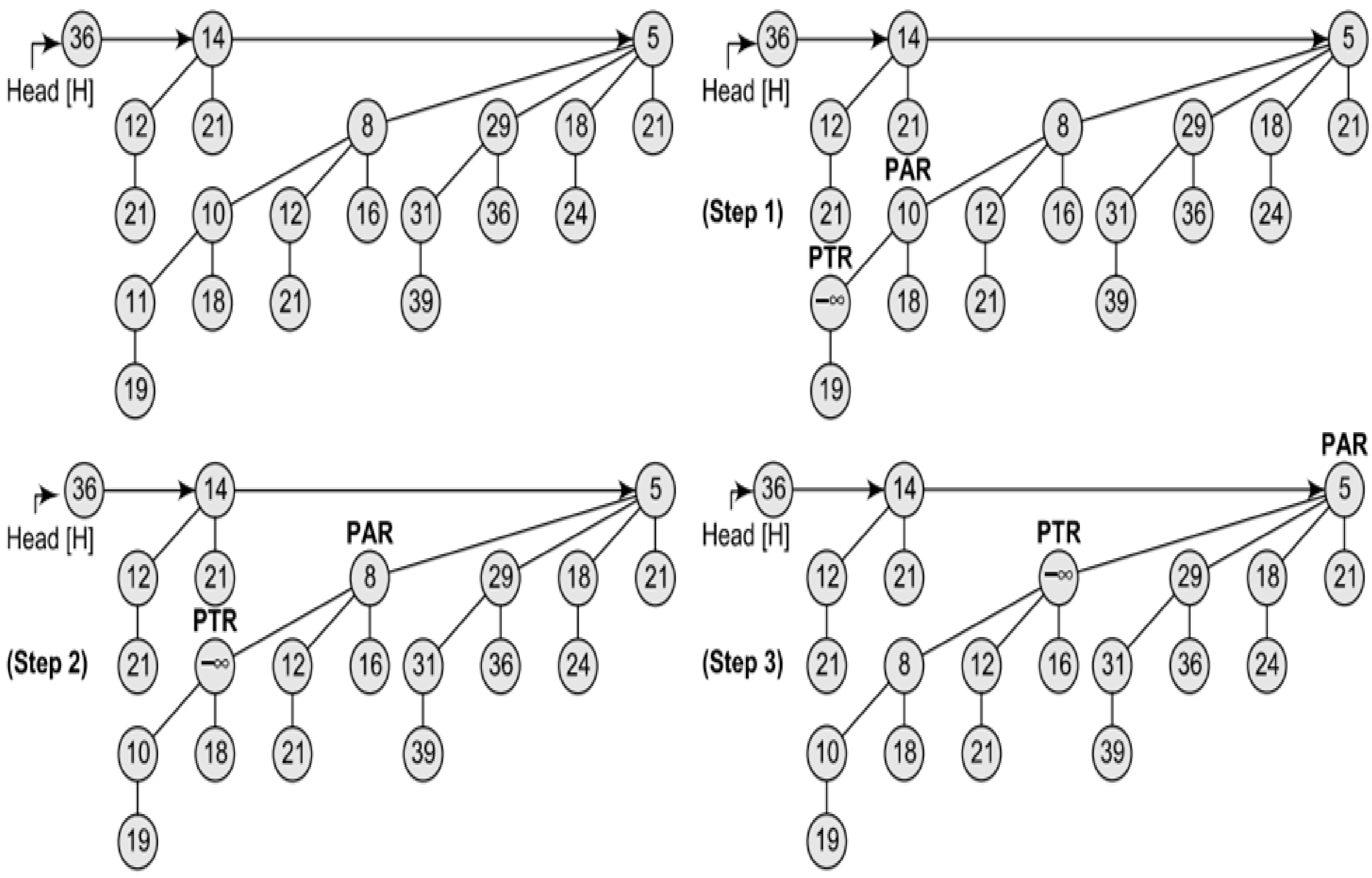
**DeleteKey—Deletes the specified key from binomial heap H.**

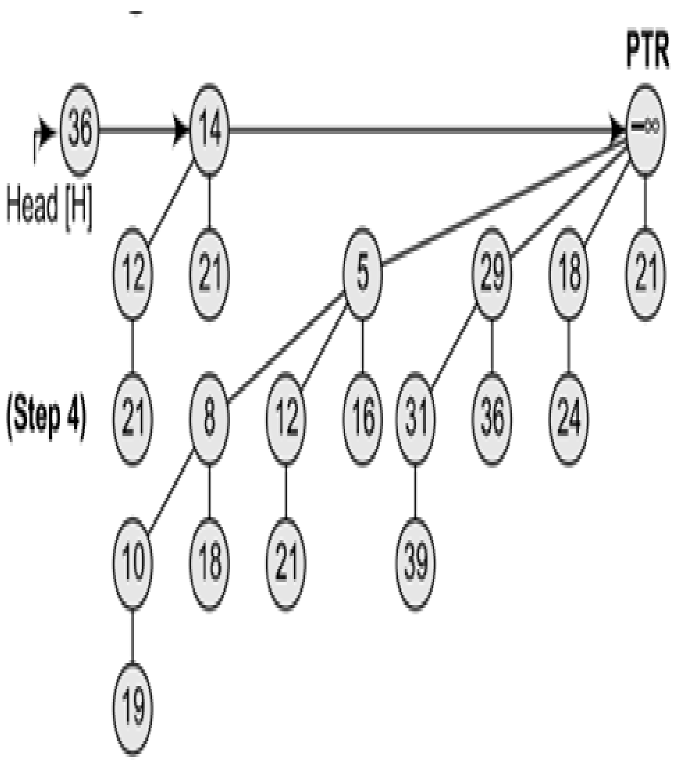
```
Binomial-Heap_Delete-Node(H, x)

Step 1: Binomial-Heap_Decrease_Val(H, x, -∞)
Step 2: Min-Extract_Binomial-Heap(H)
Step 3: END
```

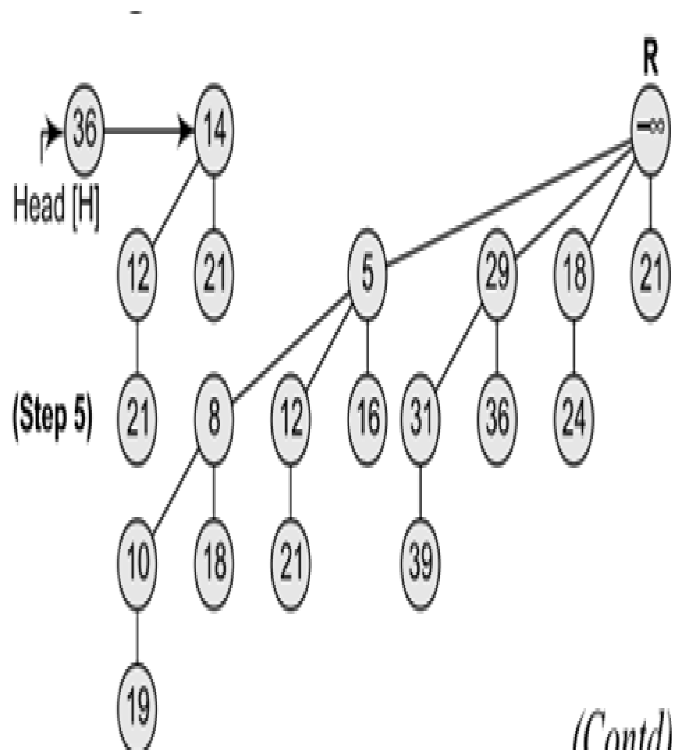
**Figure 12.24** Algorithm to delete a node from a bionomial heap

**Example 12.7** Delete the node with the value 11 from the binomial heap H.

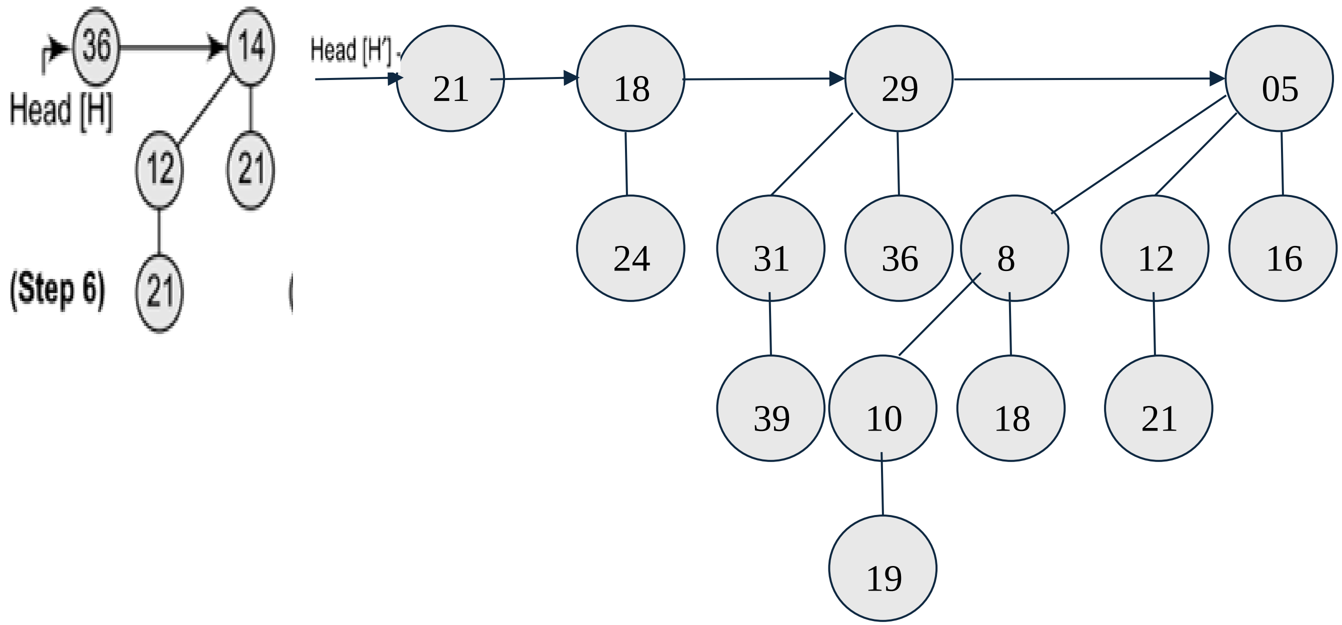
***Solution***

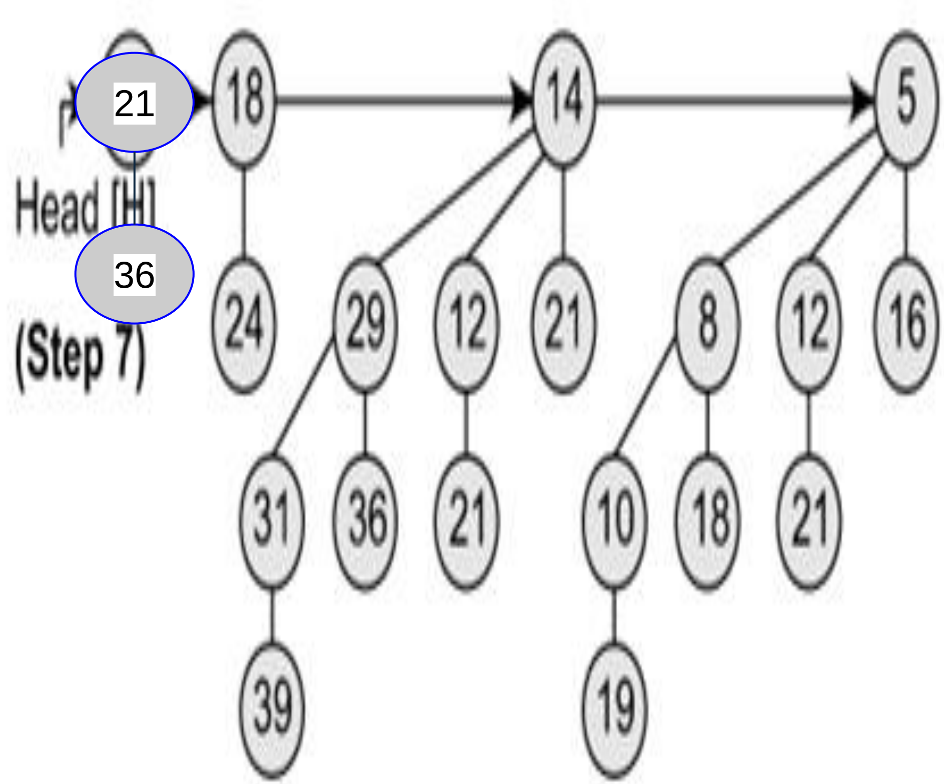


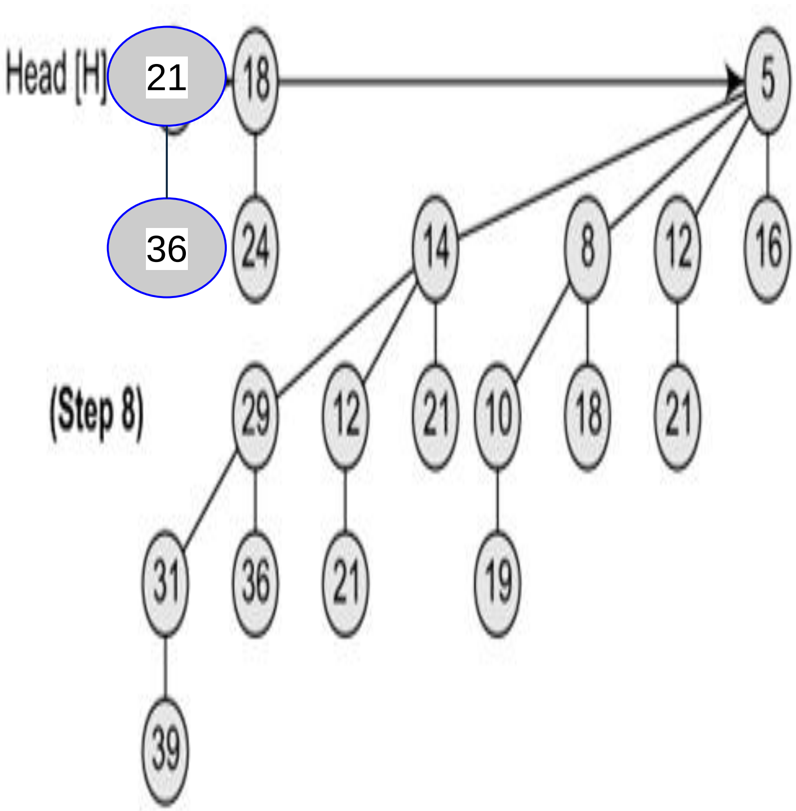

PTR
Head [H]
36
14
−∞
12
21
5
29
18
21
(Step 4)
21
8
12
16
31
36
24
10
18
21
39
19


R
Head [H]
36
14
−∞
12
21
5
29
18
21
(Step 5)
21
8
12
16
31
36
24
10
18
21
39
19

*(Contd)*


Head [H]
36
14
12
21
21
(Step 6)
Head [H']
21
18
29
05
24
31
36
8
12
16
39
10
18
21
19


Head [H]
21
36
18
14
5
24
29
12
21
8
12
16
31
36
21
10
18
21
39
19
(Step 7)


Head [H]
21
36
18
5
24
14
8
12
16
(Step 8)
29
12
21
10
18
21
31
36
21
19
39