Design and Analysis of Algorithms

Lecture-22

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

Operations defined on binomial heaps

Finding the minimum key

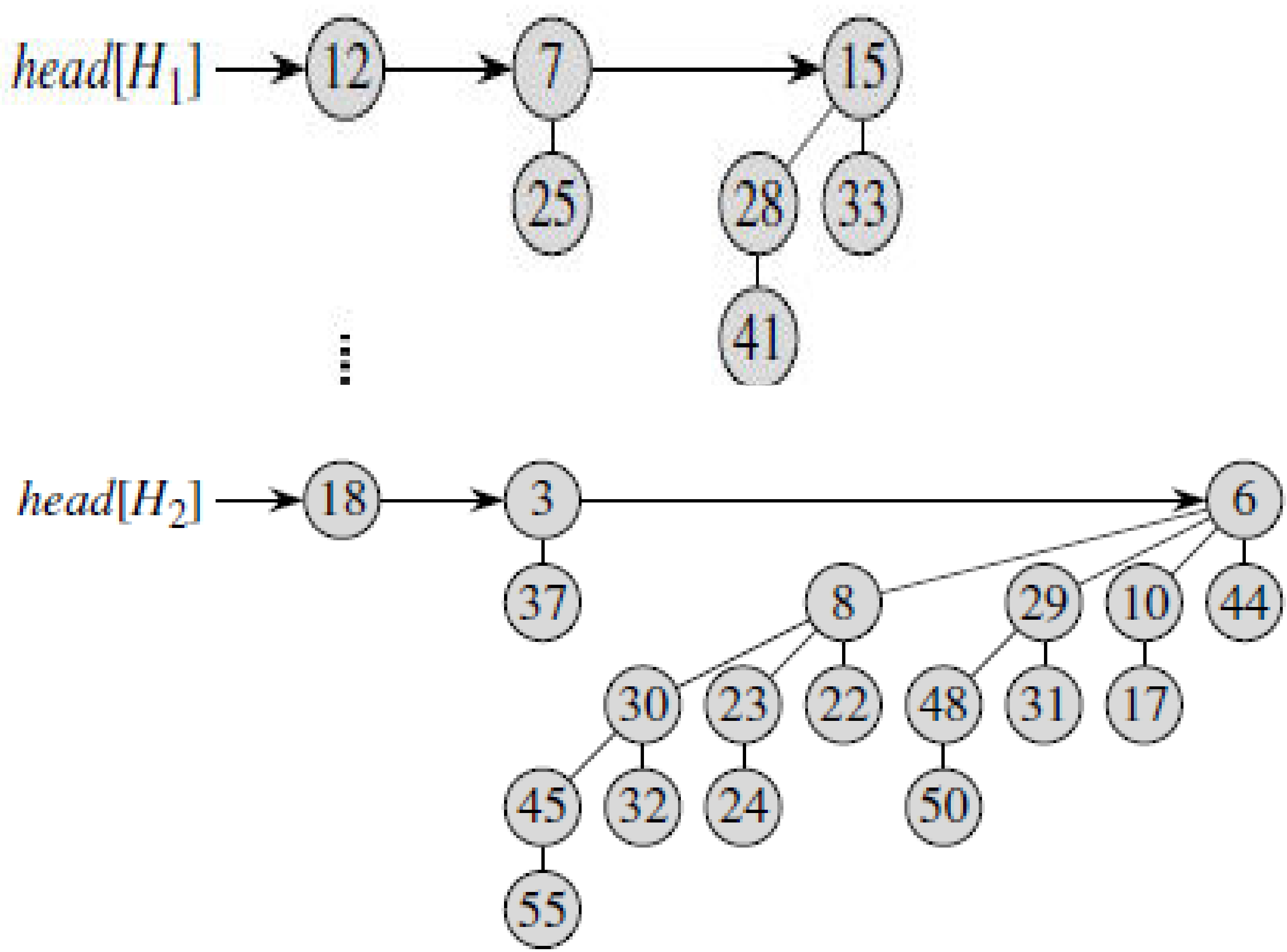
The procedure BINOMIAL-HEAP-MINIMUM returns a pointer to the node with the minimum key in an n-node binomial heap H.

```
BINOMIAL-HEAP-MINIMUM(H)
y ← NIL
x ← head[H]
min ← ∞
while x ≠ NIL
      do if key[x] < min
            then min ← key[x]
                 y ← x
         x ← sibling[x]
return y
```

Note: The running time of BINOMIAL-HEAP-MINIMUM is O(lg n).

Union of two binomial heaps

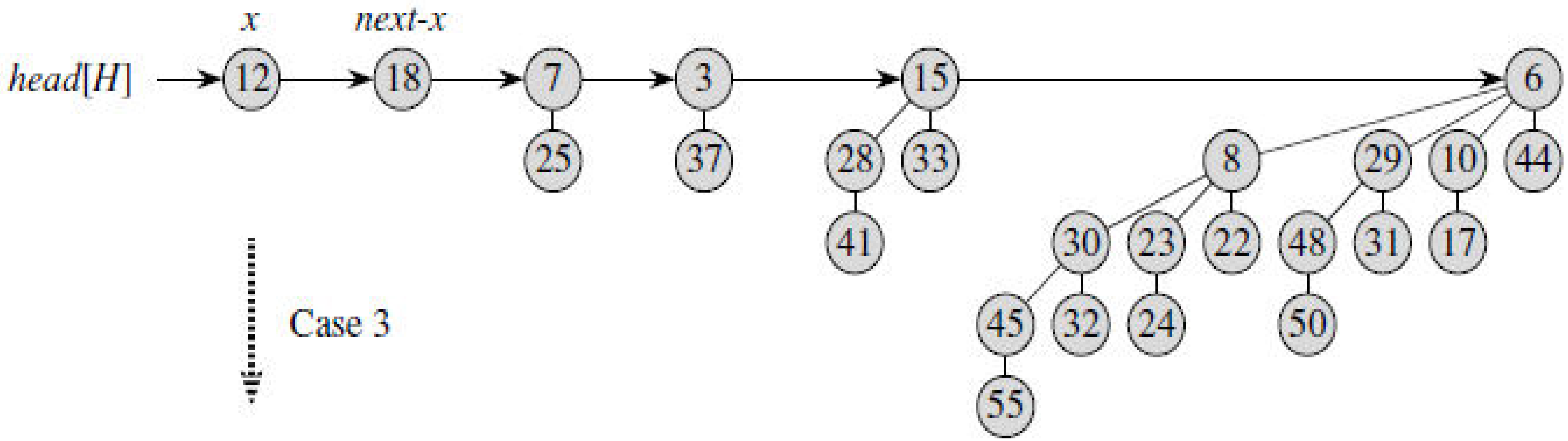
Example: Consider following two binomial heaps H_1 and H_2. Find the union of these binomial heaps.



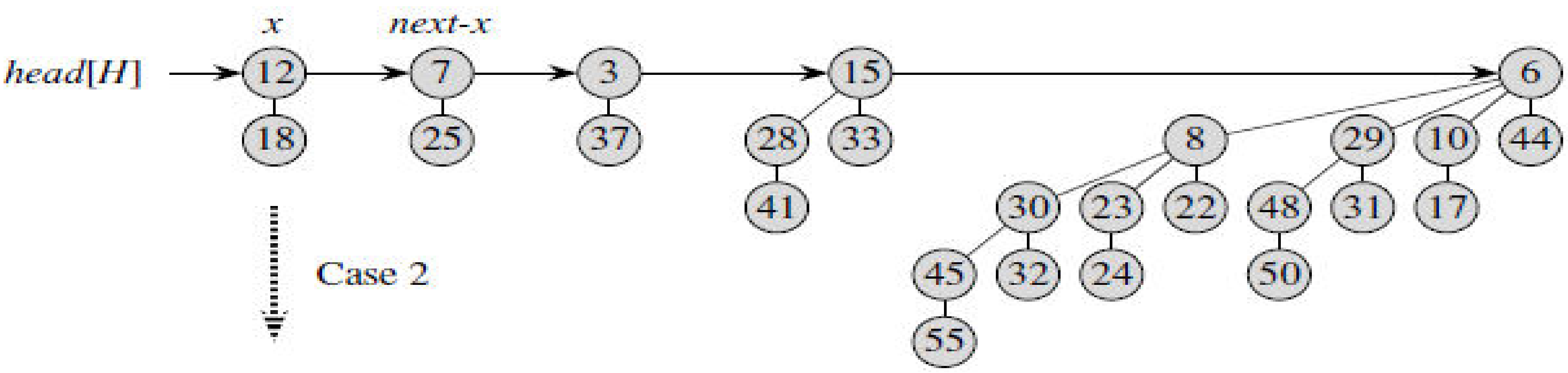
Union of two binomial heaps

Solution:

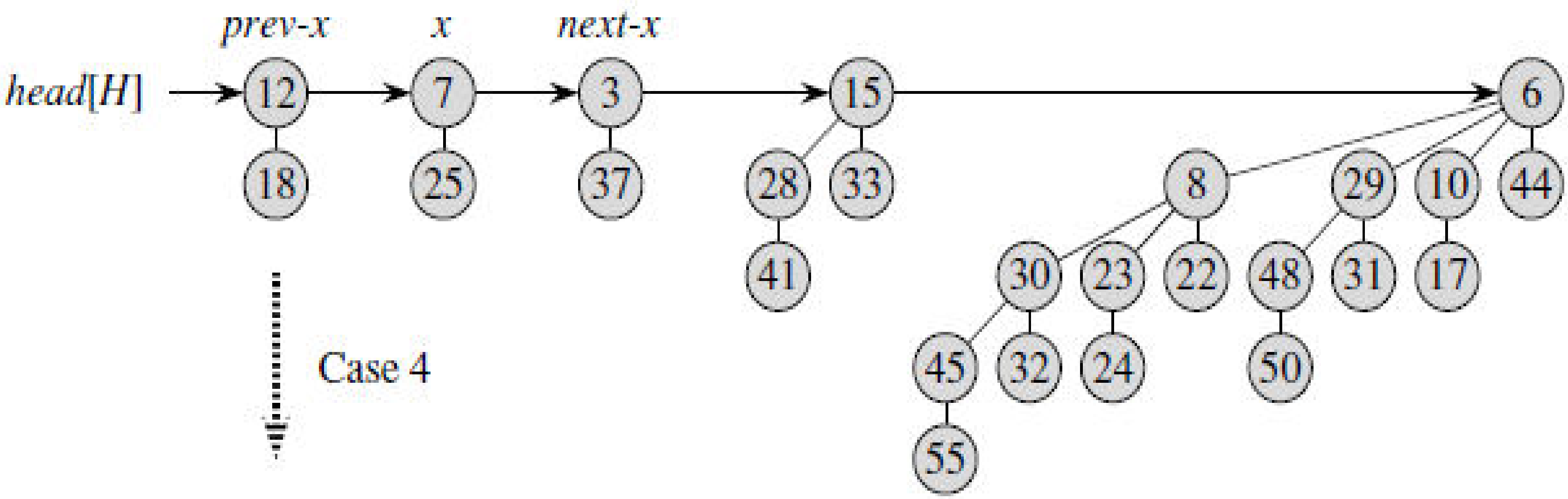
Step-1: Merge both binomial heaps.

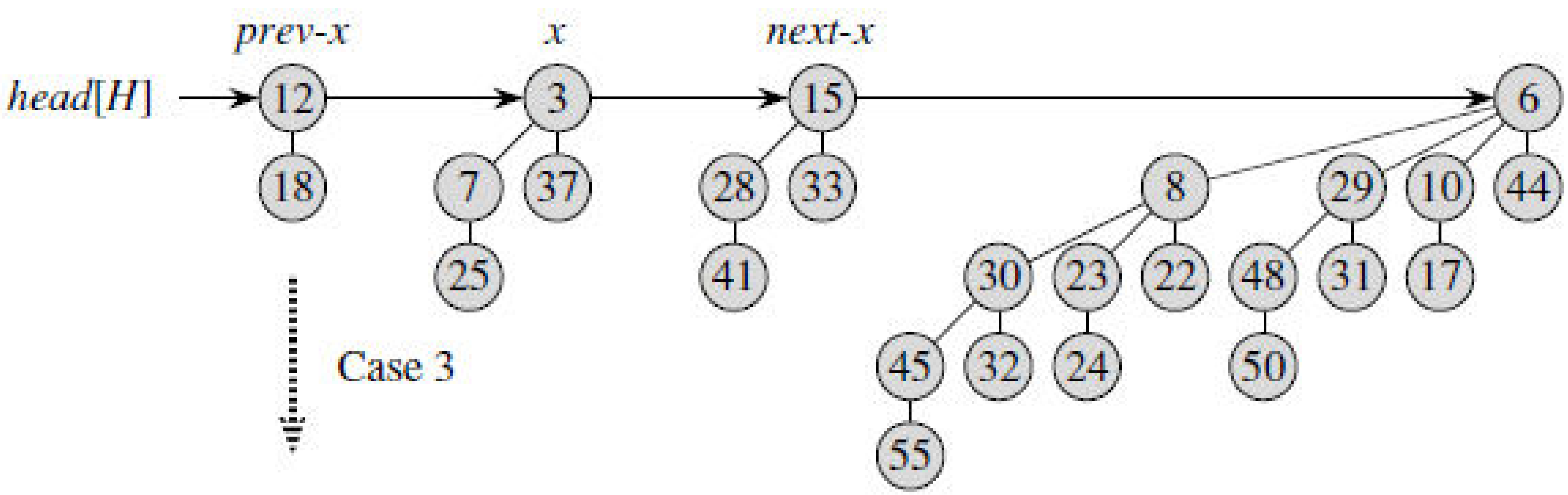


Step-2: Apply the linking process of equal degree root nodes.

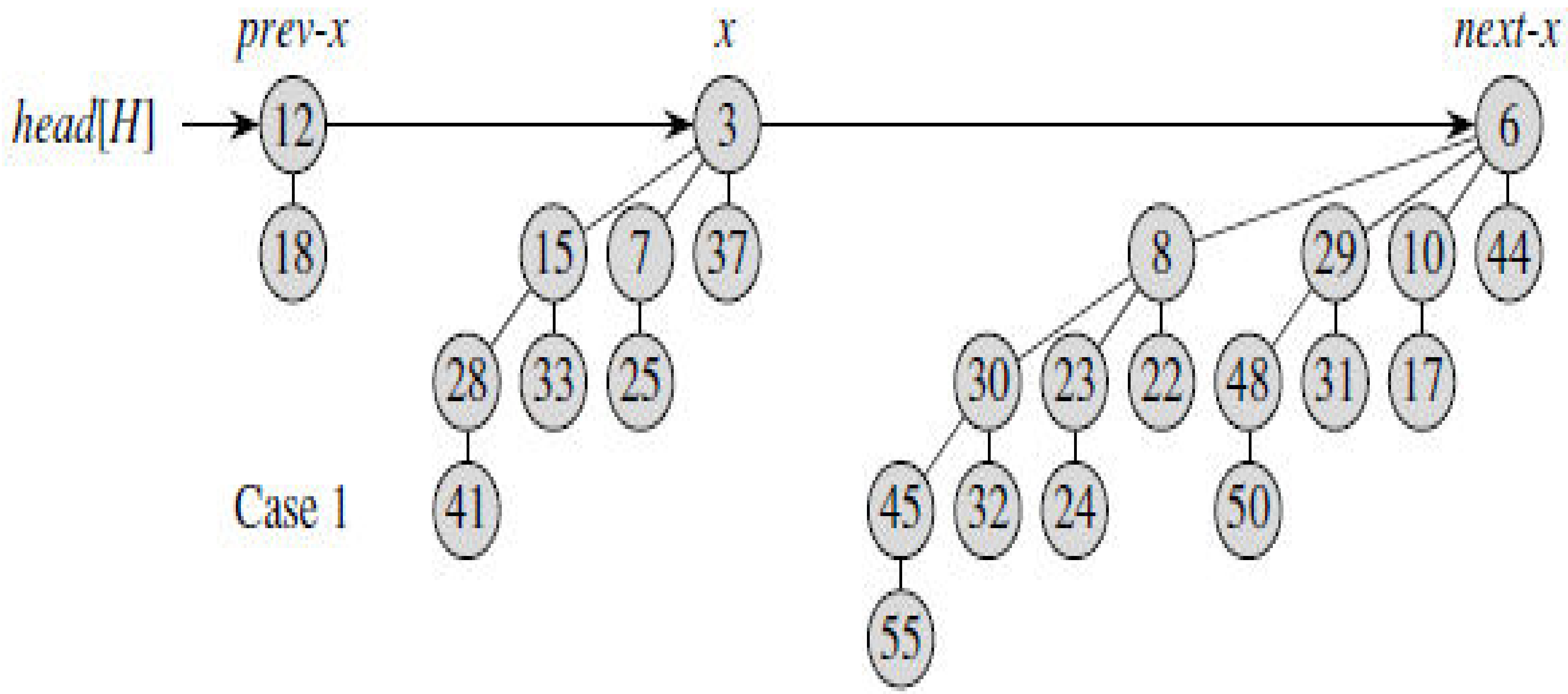


Union of two binomial heaps





Union of two binomial heaps



Final binomial heap

Union of two binomial heaps

```
BINOMIAL-HEAP-UNION(H1, H2)
H ← MAKE-BINOMIAL-HEAP()
head[H] ← BINOMIAL-HEAP-MERGE(H1, H2)
free the objects H1 and H2 but not the lists they point to
if head[H] = NIL
   then return H
prev-x ← NIL
x ← head[H]
next-x ← sibling[x]
while next-x ≠ NIL
     do if (degree[x] ≠ degree[next-x]) or
              (sibling[next-x] ≠ NIL and degree[sibling[next-x]] = degree[x])
           then prev-x ← x                                   ▷ Cases 1 and 2
                x ← next-x                                   ▷ Cases 1 and 2
           else  if key[x] ≤ key[next-x]
                    then sibling[x] ← sibling[next-x]              ▷ Case 3
                         BINOMIAL-LINK(next-x, x)                  ▷ Case 3
                    else  if prev-x = NIL                          ▷ Case 4
                             then head[H] ← next-x                 ▷ Case 4
                             else  sibling[prev-x] ← next-x        ▷ Case 4
                          BINOMIAL-LINK(x, next-x)                 ▷ Case 4
                          x ← next-x                               ▷ Case 4
        next-x ← sibling[x]
return H
```

Union of two binomial heaps

The BINOMIAL-HEAP-UNION procedure has two phases.

- The first phase, performed by the call of BINOMIAL-HEAP-MERGE, merges the root lists of binomial heaps H1 and H2 into a single linked list H that is sorted by degree into monotonically increasing order.
- In the second phase, we link roots of equal degree until at most one root remains of each degree.

BINOMIAL-LINK(y, z)

$p[y] \leftarrow z$
$sibling[y] \leftarrow child[z]$
$child[z] \leftarrow y$
$degree[z] \leftarrow degree[z] + 1$

Time Complexity:

Time complexity of BINOMIAL-HEAP-UNION is O(lg n).

Inserting a node

The following procedure inserts node x into binomial heap H, assuming that x has already been allocated and key[x] has already been filled in.

BINOMIAL-HEAP-INSERT(H, x)

$H' \leftarrow$ MAKE-BINOMIAL-HEAP()
$p[x] \leftarrow$ NIL
$child[x] \leftarrow$ NIL
$sibling[x] \leftarrow$ NIL
$degree[x] \leftarrow 0$
$head[H'] \leftarrow x$
$H \leftarrow$ BINOMIAL-HEAP-UNION(H, H')

Time Complexity:

Time complexity of this algorithm is O(lg n).