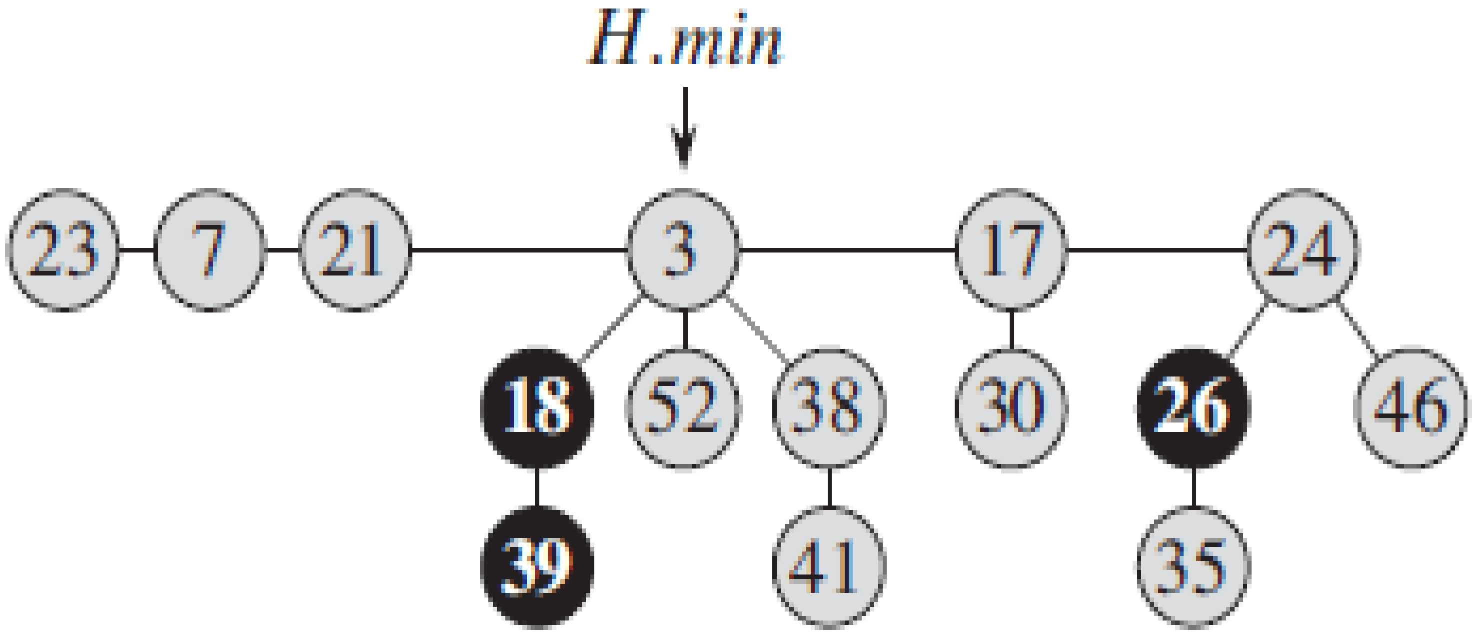
Design and Analysis of Algorithms

Lecture-24

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

Extracting the minimum node

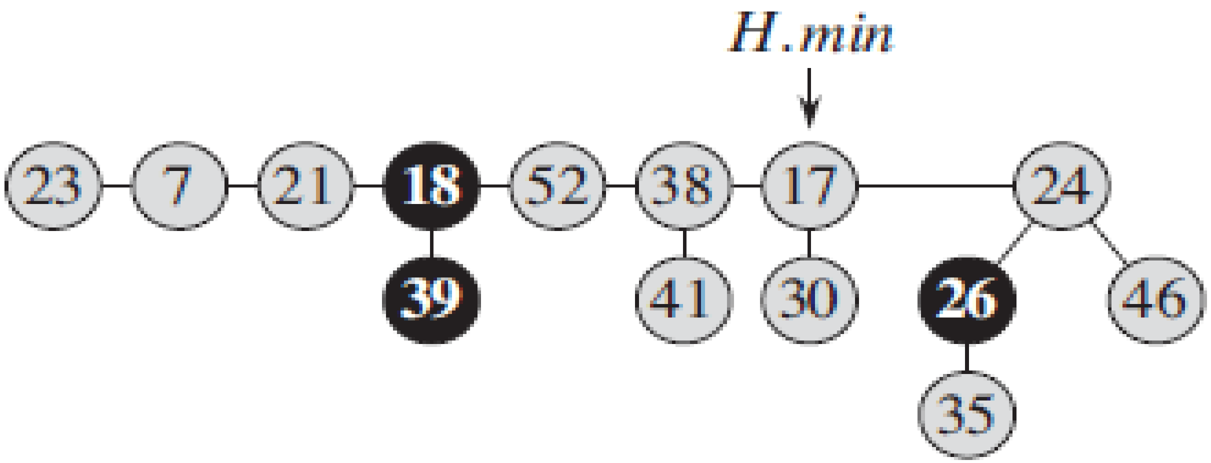
Example: Extract the minimum node from the following Fibonacci heap.



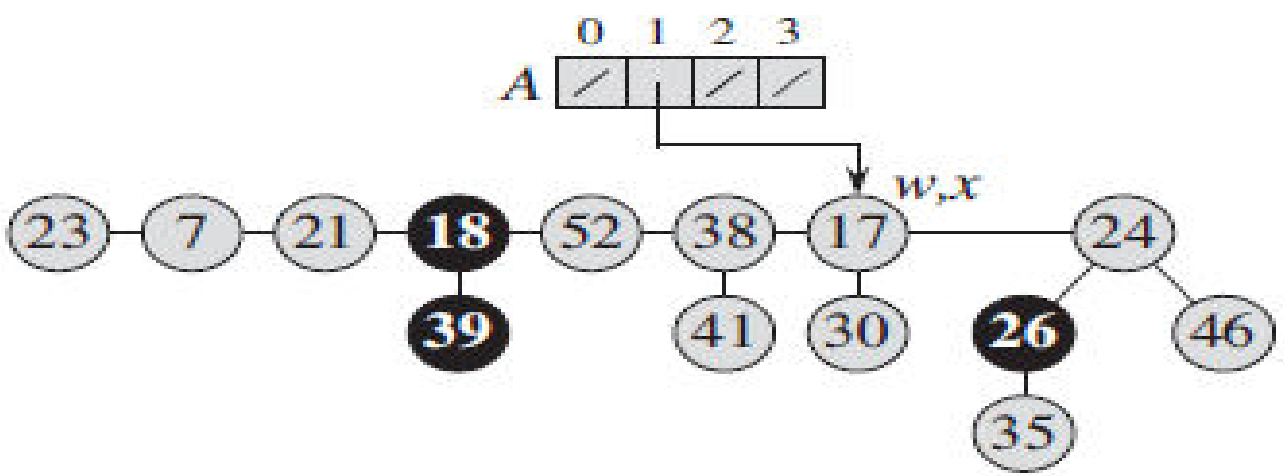
Extracting the minimum node

Solution:

Step-1:

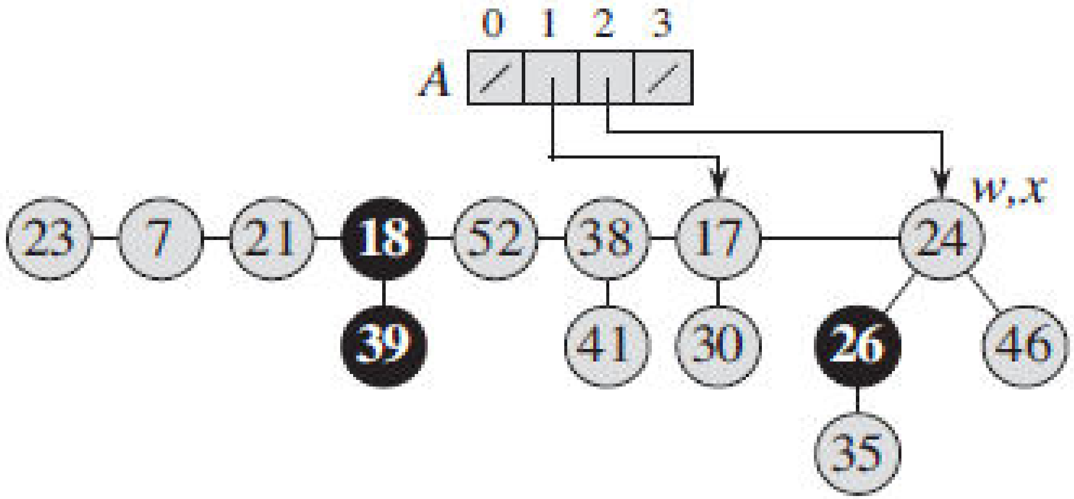


Step-2:

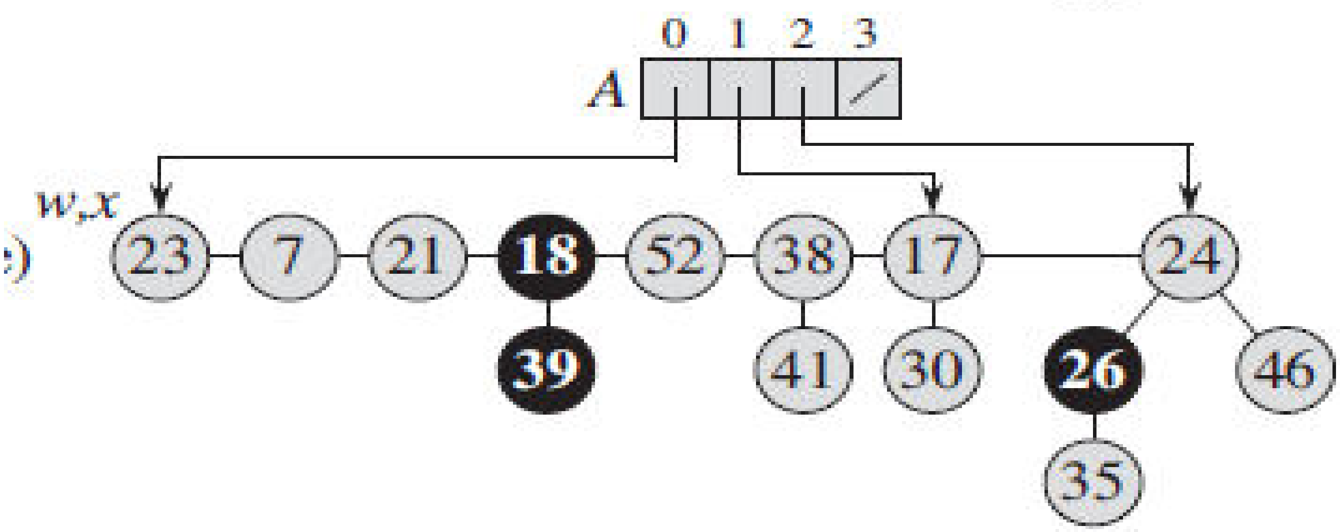


Extracting the minimum node

Step-3:

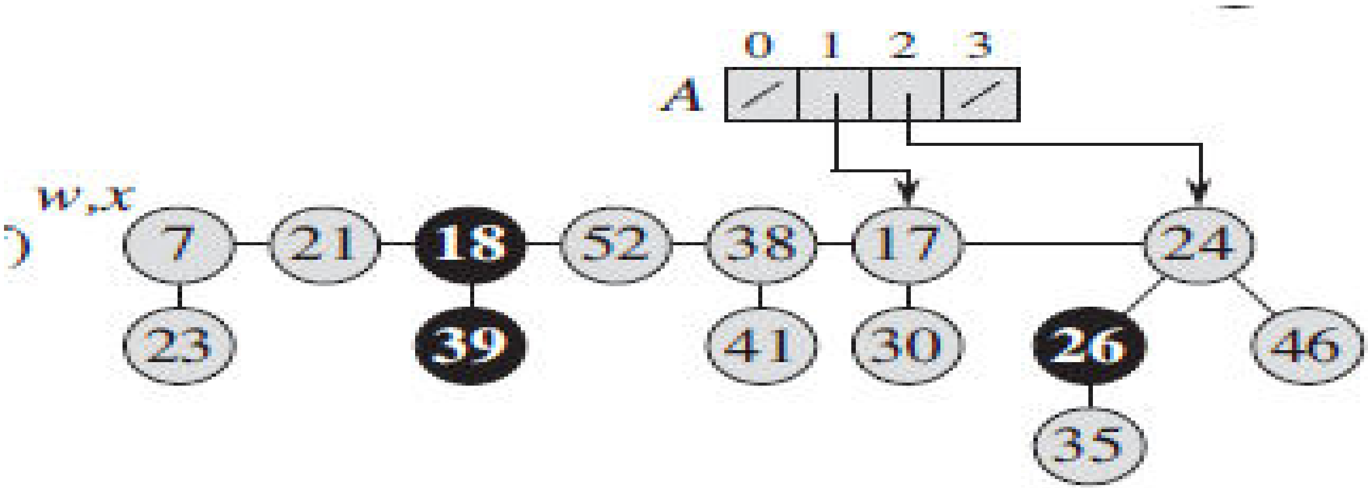


Step-4·

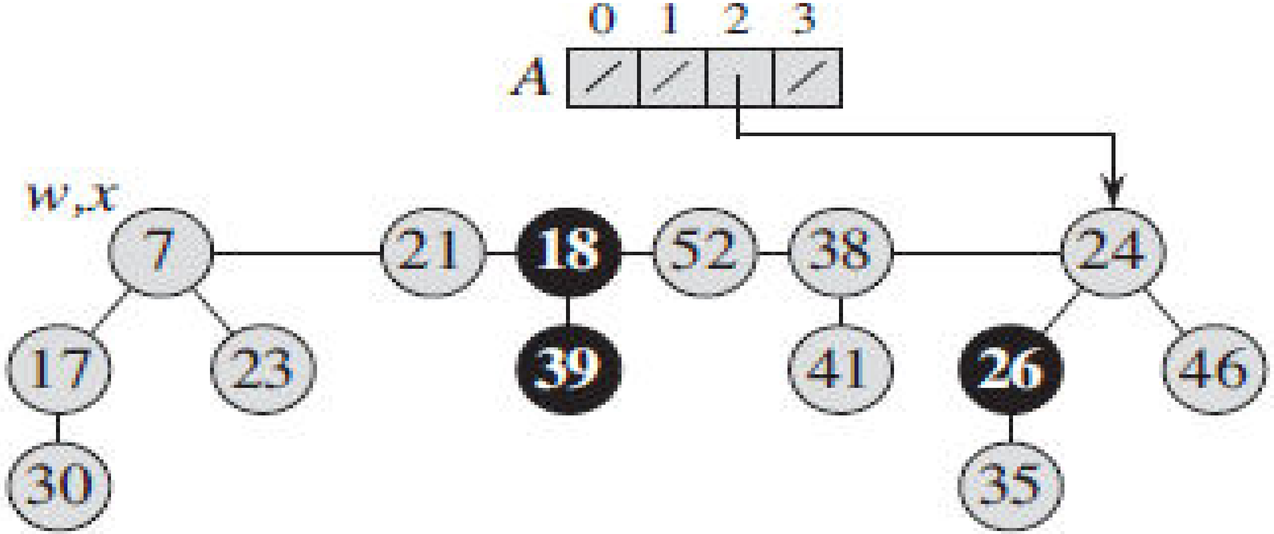


Extracting the minimum node

Step-5:

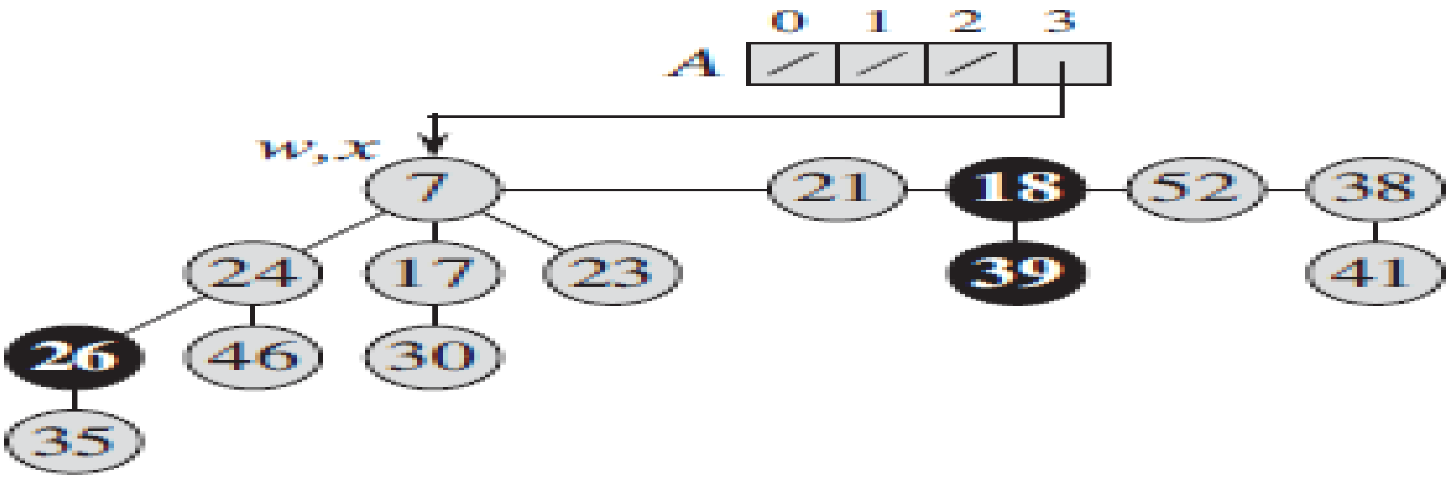


Step-6:

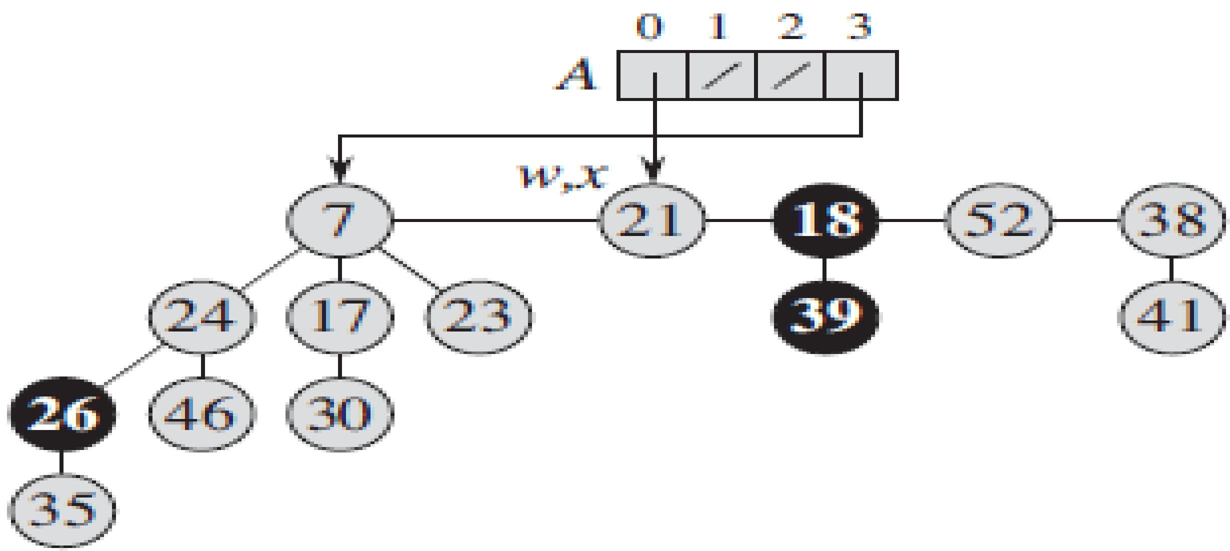


Extracting the minimum node

Step-7:

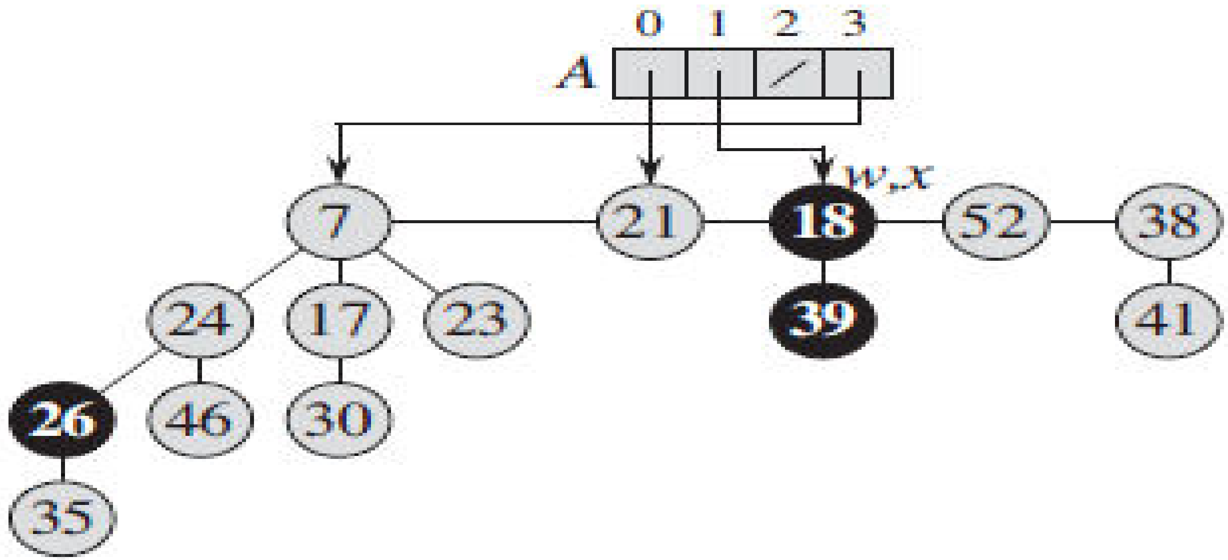


Step-8:

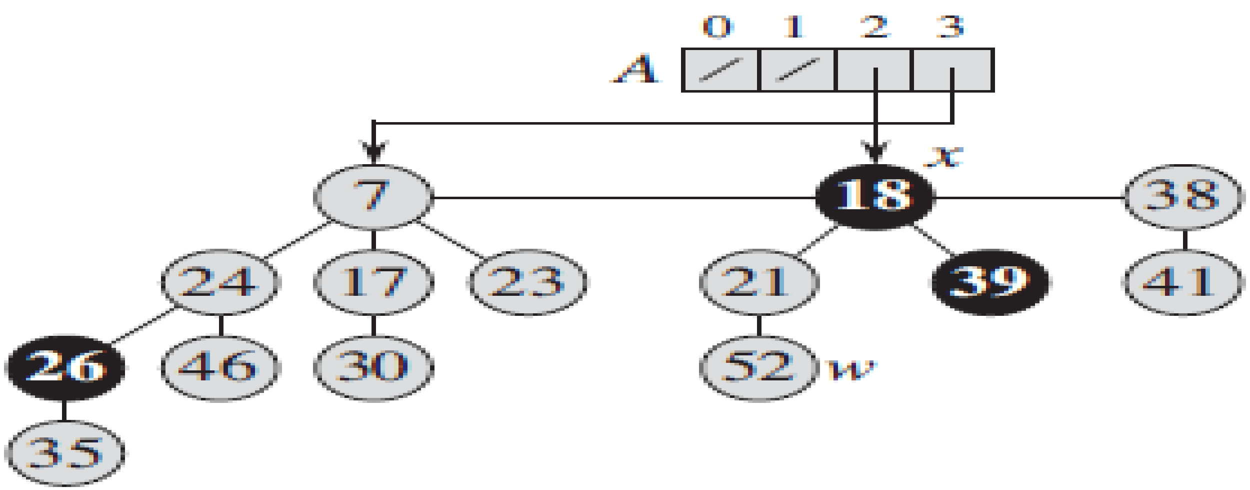


Extracting the minimum node

Step-9:

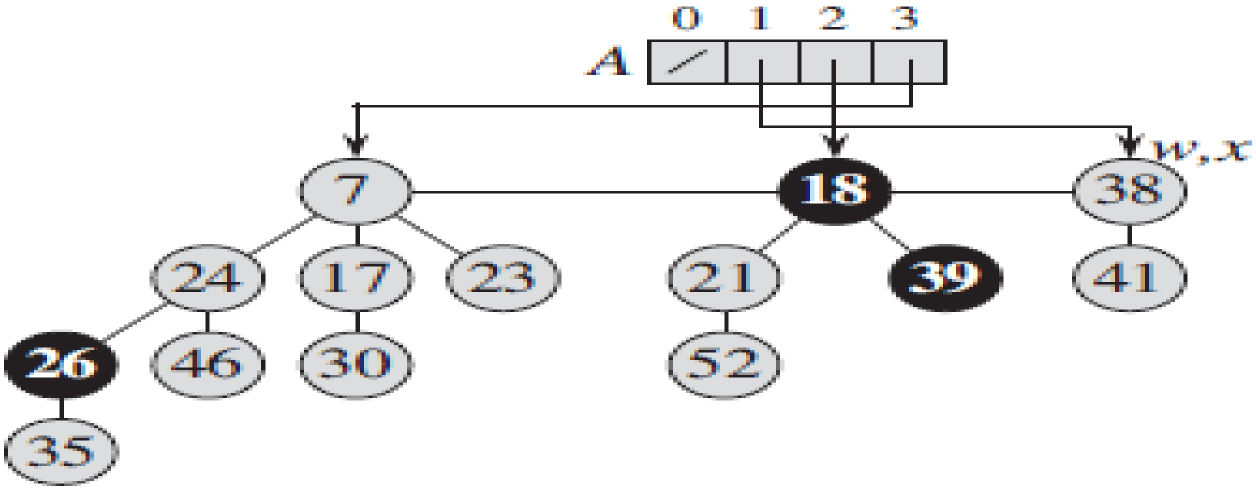


Step-10:

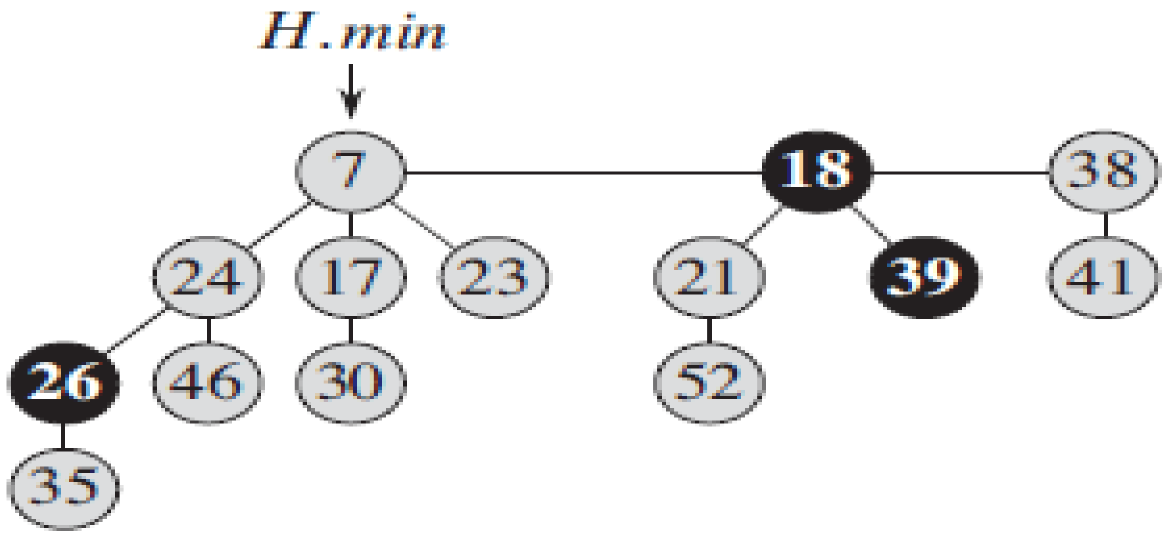


Extracting the minimum node

Step-11:



Step-12:



Final Fibonacci Heap

Extracting the minimum node

```
FIB-HEAP-EXTRACT-MIN(H)
z = H.min
if z ≠ NIL
    for each child x of z
        add x to the root list of H
        x.p = NIL
    remove z from the root list of H
    if z == z.right
        H.min = NIL
    else H.min = z.right
        CONSOLIDATE(H)
    H.n = H.n − 1
return z
```

Extracting the minimum node

```
CONSOLIDATE(H)
let A[0 .. D(H.n)] be a new array
for i = 0 to D(H.n)
    A[i] = NIL
for each node w in the root list of H
    x = w
    d = x.degree
    while A[d] ≠ NIL
        y = A[d]          // another node with the same degree as x
        if x.key > y.key
            exchange x with y
        FIB-HEAP-LINK(H, y, x)
        A[d] = NIL
        d = d + 1
    A[d] = x
H.min = NIL
for i = 0 to D(H.n)
    if A[i] ≠ NIL
        if H.min == NIL
            create a root list for H containing just A[i]
            H.min = A[i]
        else insert A[i] into H's root list
            if A[i].key < H.min.key
                H.min = A[i]
```

Extracting the minimum node

```
FIB-HEAP-LINK(H, y, x)
remove y from the root list of H
make y a child of x, incrementing x.degree
y.mark = FALSE
```

Computation of Amortized cost:

Let H denote the Fibonacci heap just prior to the FIB-HEAP-EXTRACT-MIN operation. Let n is the number of nodes in Fibonacci heap H. Let H' is the Fibonacci heap after this operation. Therefore,

Actual cost = $O(t(H)-1 + D(n)) = O(D(n) + t(H))$

Now, $t(H') = D(n)$ and $m(H') = m(H)$

Therefore, amortized cost = actual cost + change in potential

$$= O(D(n) + t(H)) + (\Phi(H') - \Phi(H))$$

$$= O(D(n) + t(H)) + (D(n) + 2m(H) - t(H) - 2m(H))$$

$$= O(D(n) + t(H) + D(n) - t(H))$$

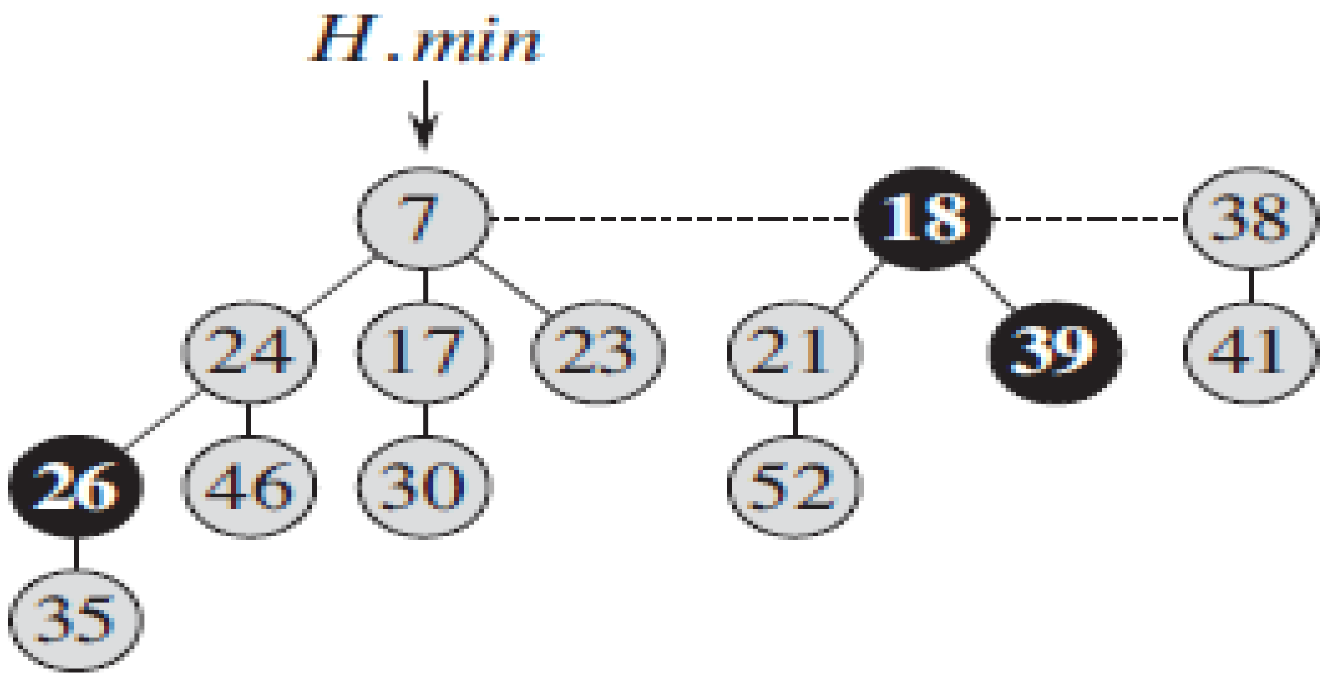
$$= O(D(n)) = O(\log n)$$

Decreasing a key

Example: Consider following Fibonacci heap.

(1) Decrease the node with key 46 to key value 15.

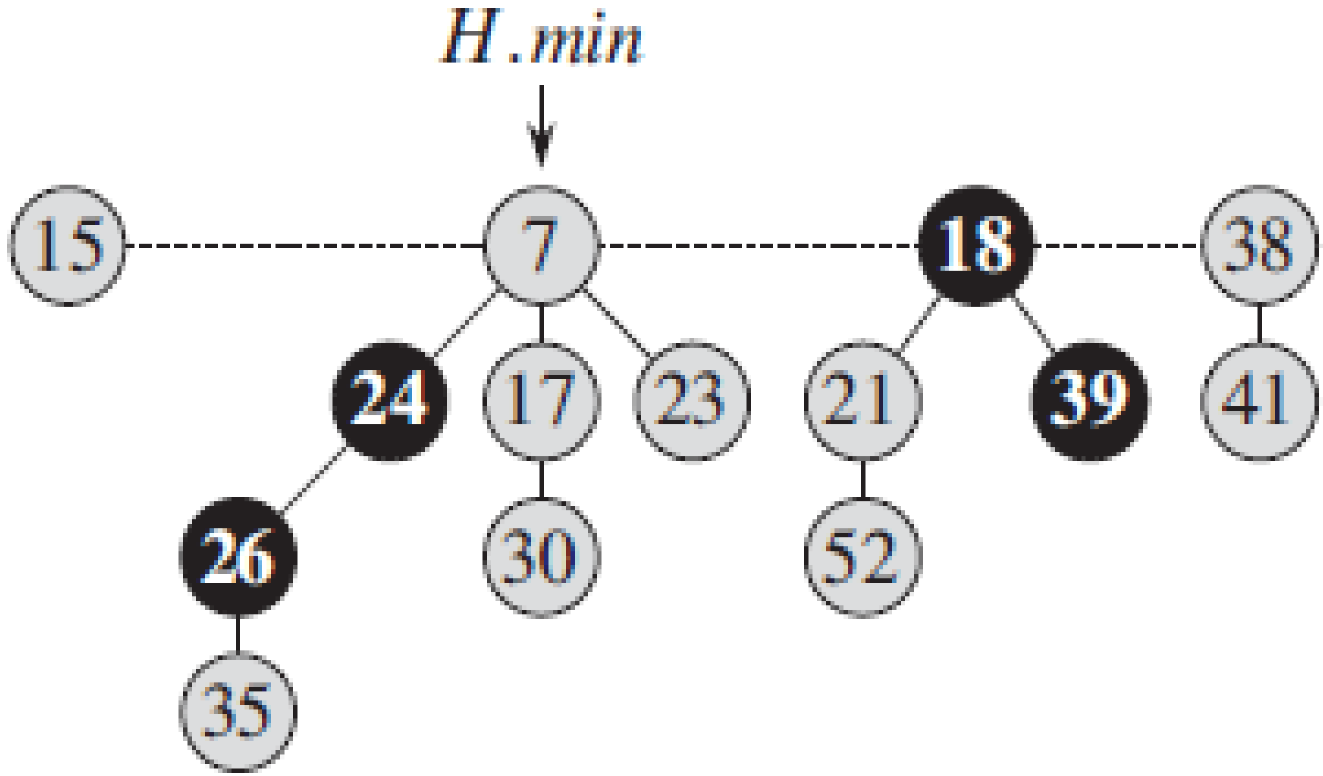
(2) After this, decrease node with key 35 to key value 5.



Decreasing a key

Solution:

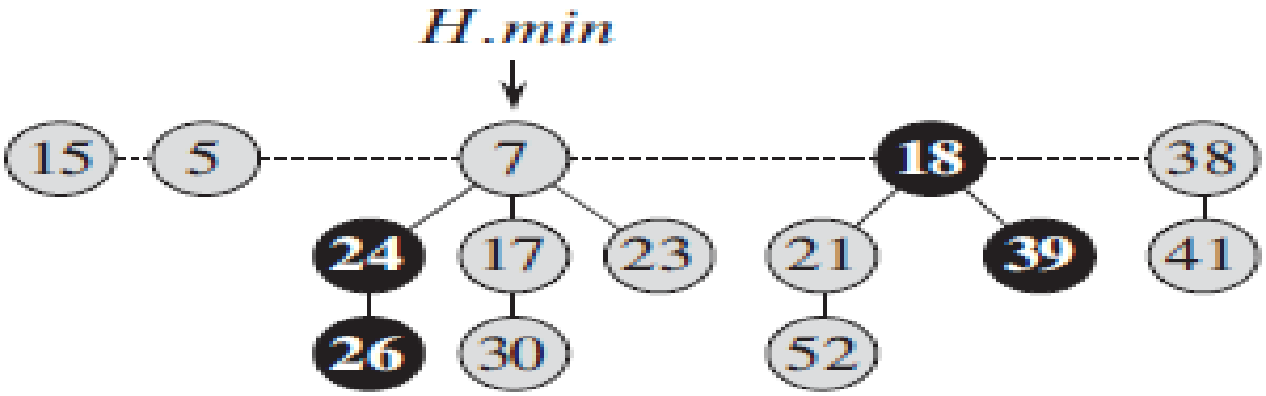
(1) Decrease the node with key 46 to key value 15.



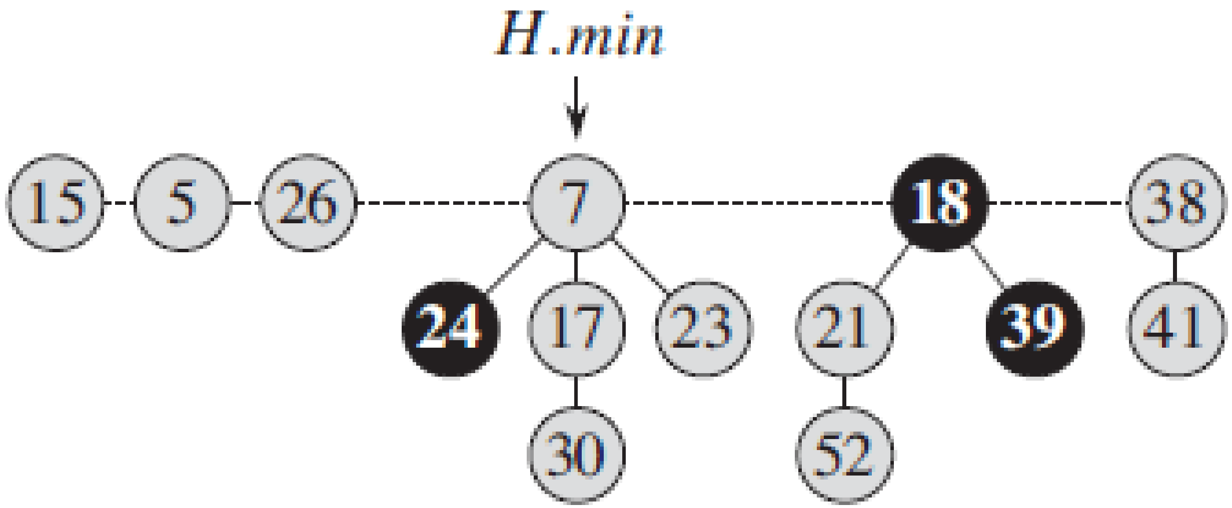
Decreasing a key

2. Decrease the node with key 35 to key value 5.

Step-1:

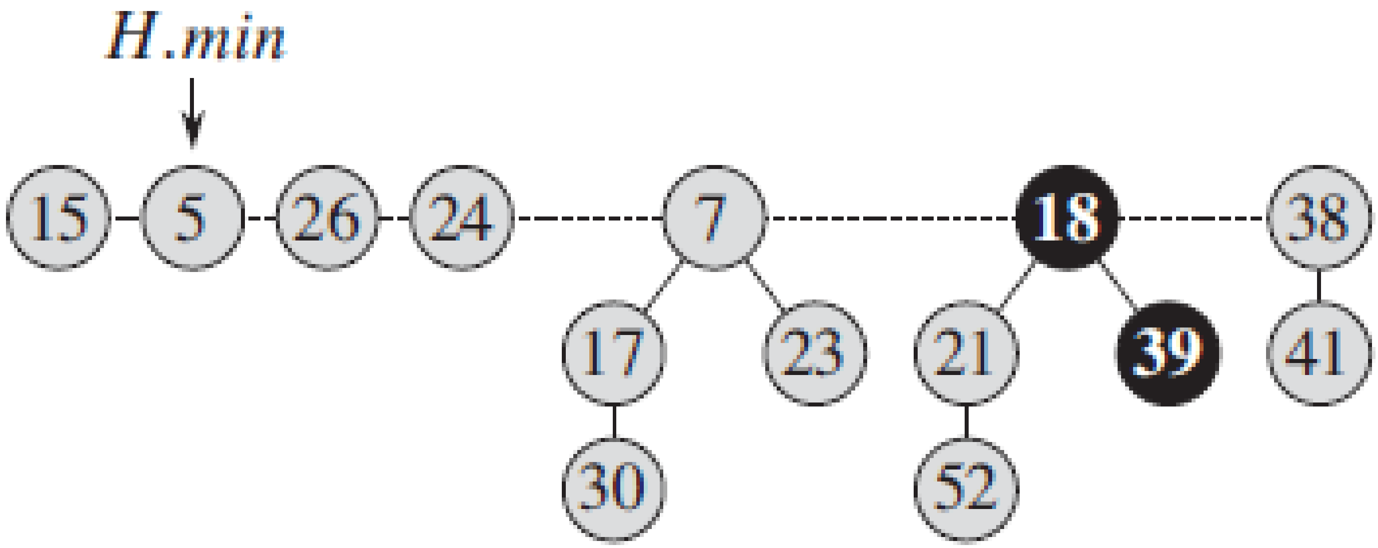


Step-2:



Decreasing a key

Step-3:



Final Fibonacci heap

Decreasing a key

```
FIB-HEAP-DECREASE-KEY(H, x, k)
if k > x.key
    error "new key is greater than current key"
x.key = k
y = x.p
if y ≠ NIL and x.key < y.key
    CUT(H, x, y)
    CASCADING-CUT(H, y)
if x.key < H.min.key
    H.min = x
```

Decreasing a key

```
CUT(H, x, y)
remove x from the child list of y, decrementing y.degree
add x to the root list of H
x.p = NIL
x.mark = FALSE
```

```
CASCADING-CUT(H, y)
z = y.p
if z ≠ NIL
    if y.mark == FALSE
        y.mark = TRUE
    else CUT(H, y, z)
        CASCADING-CUT(H, z)
```

Decreasing a key

Amortized cost:

Suppose the cascading cut function is called c times.

Therefore, the actual cost of FIB-HEAP-DECREASE-KEY is $O(c)$.

Now, Let H is the initial Fibonacci heap and H' is the Fibonacci heap after this operation. Therefore,

$$t(H') = t(H) + c$$

(the original t(H) trees, c-1 trees produced by cascading cuts, and the tree rooted at x)

Maximum number of marked nodes,

$$m(H') = m(H) - c + 2$$

(c -1 were unmarked by cascading cuts and the last call of CASCADING-CUT may have marked a node)

Therefore, amortized cost $= O(c) + ((t(H') + 2m(H')) -(t(H) + 2m(H)))$

$$= O(c) + (t(H) + c + 2(m(H) - c + 2) - (t(H) + 2m(H)))$$

$$= O(c) - c + 4 \ = O(4) = O(1)$$

Thank You.