Design and Analysis of Algorithms

Lecture-24

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

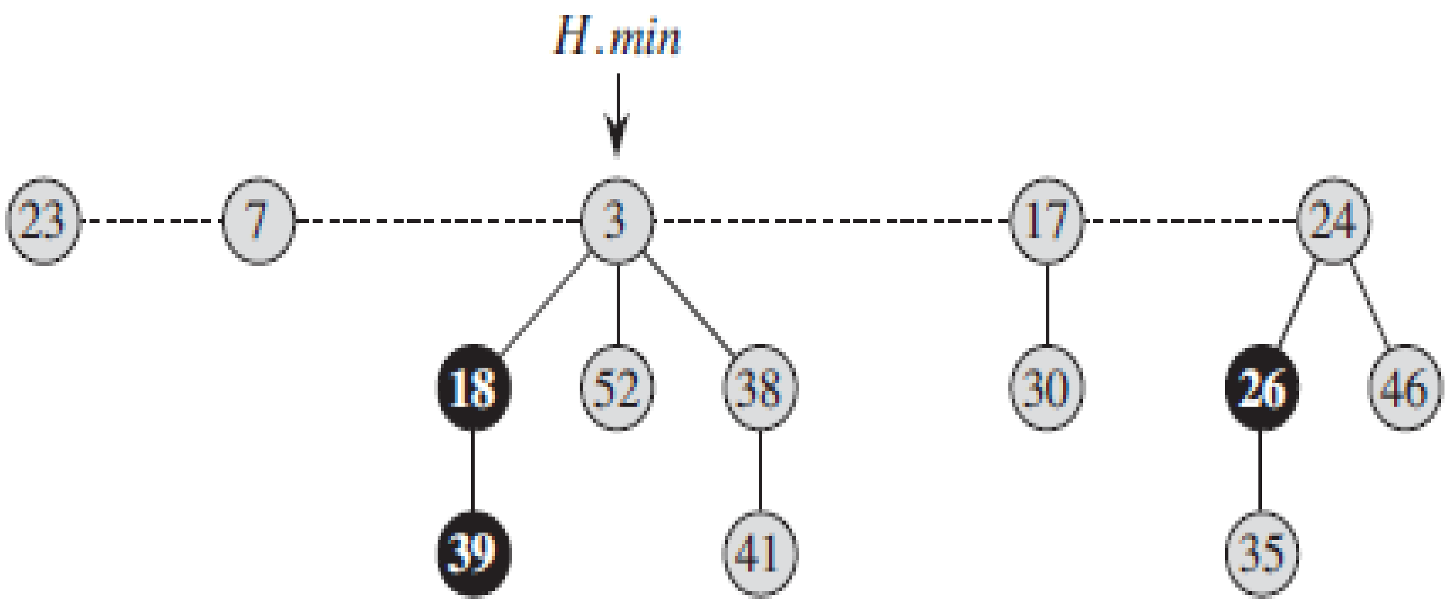
Fibonacci Heap

Fibonacci Heap

Definition

A Fibonacci heap is a collection of rooted trees that are min-heap ordered. That is, each tree obeys the min-heap property: the key of a node is greater than or equal to the key of its parent.

Example:



Representation of Fibonacci Heap

- Circular doubly linked list is used to represent Fibonacci heap.
- Each node x in binomial heap consists of following fields:-

  x.p → pointer points to parent of x

  x.child → pointer points to any child of x

  x.left → pointer points to left sibling of x

  x.right → pointer points to right sibling of x

  x.key → value stored at node x

  x.degree→ number of children of node x

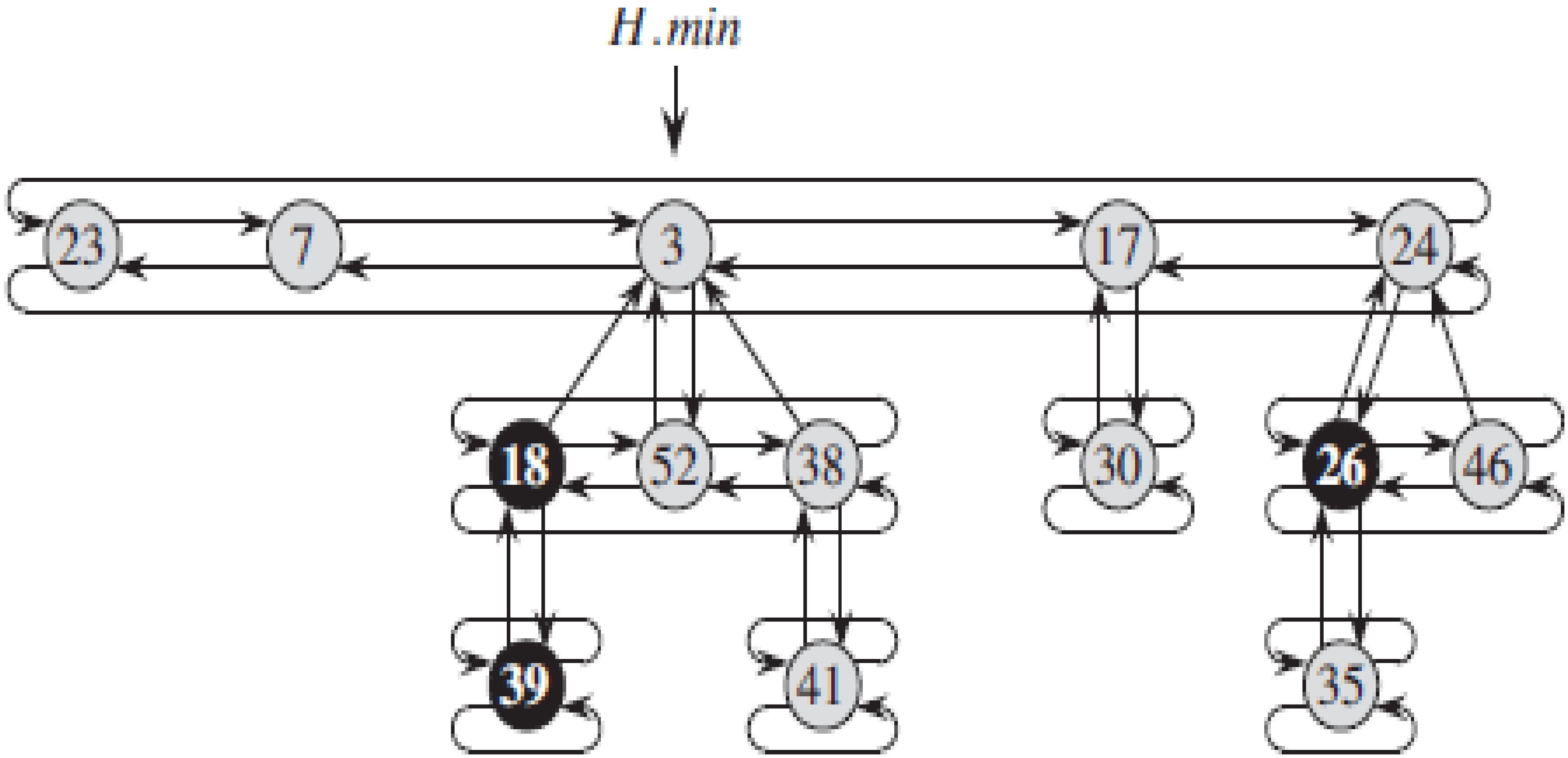
  x.mark → The boolean-valued attribute indicates whether node x has lost a child since the last time x was made the child of another node.

- Fibonacci heap H has two fields:- H.min and H.n .

H.min → pointer points to the root of a tree containing the minimum key;

H.n → the number of nodes currently in H

Representation of Fibonacci Heap



Potential Function

To analyze the performance of Fibonacci heap, we use the potential function.

We then define the potential $\Phi(\mathrm{H})$ of Fibonacci heap H by

$$\Phi(\mathrm{H}) = \mathrm{t}(\mathrm{H}) + 2\mathrm{m}(\mathrm{H})$$

Where, $\mathrm{t}(\mathrm{H})$ is the number of tree in H and $\mathrm{m}(\mathrm{H})$ is the number of marked nodes.

Example: Consider the Fibonacci heap of previous slide.

Here, t(H) = 5 and m(H) = 3. Therefore

$$\Phi(\mathrm{H}) = 5 + 2*3 = 11$$

Maximum degree

Maximum degree of any n-node Fibonacci is denoted by D(n).

D(n) ≤ $\lfloor \log n \rfloor$

Amortized Cost

Amortized cost is computed for an operation. It is defined as following:-

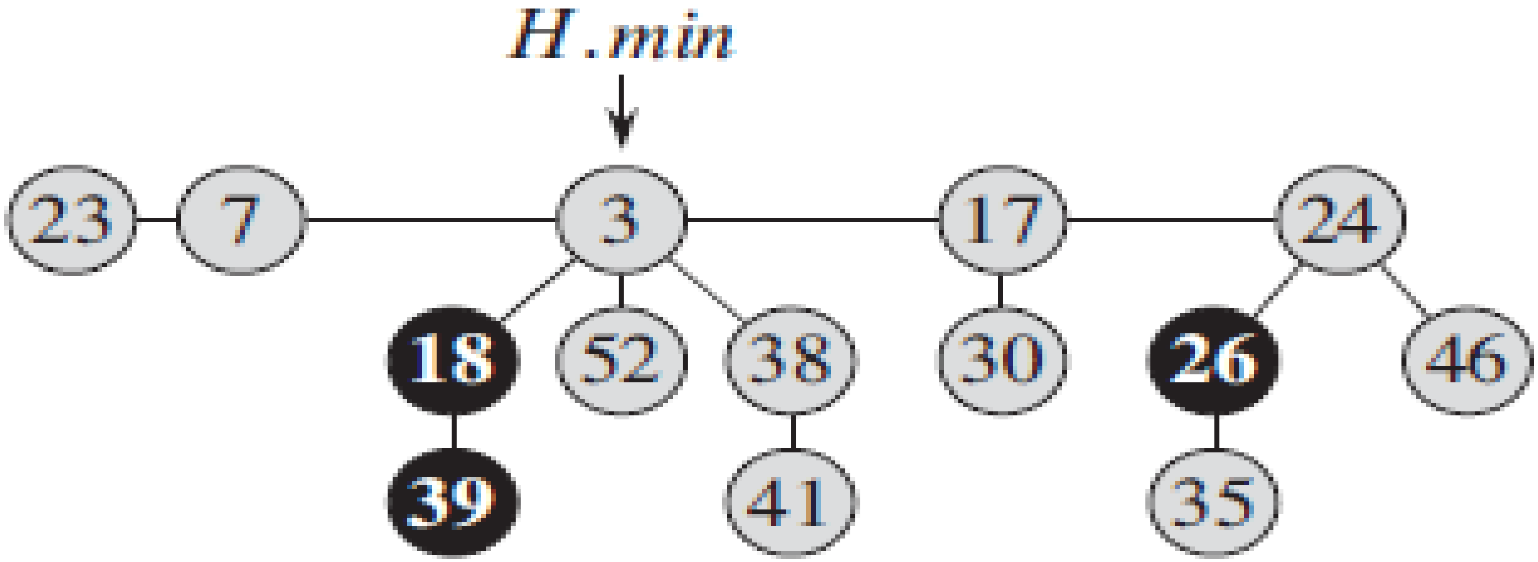
Amortized cost = Actual cost + change in potential function due to operation

Mergeable-heap operations

1. Inserting a node
2. Finding the minimum node
3. Uniting two Fibonacci heaps
4. Extracting the minimum node
5. Decreasing a key
6. Deleting a node

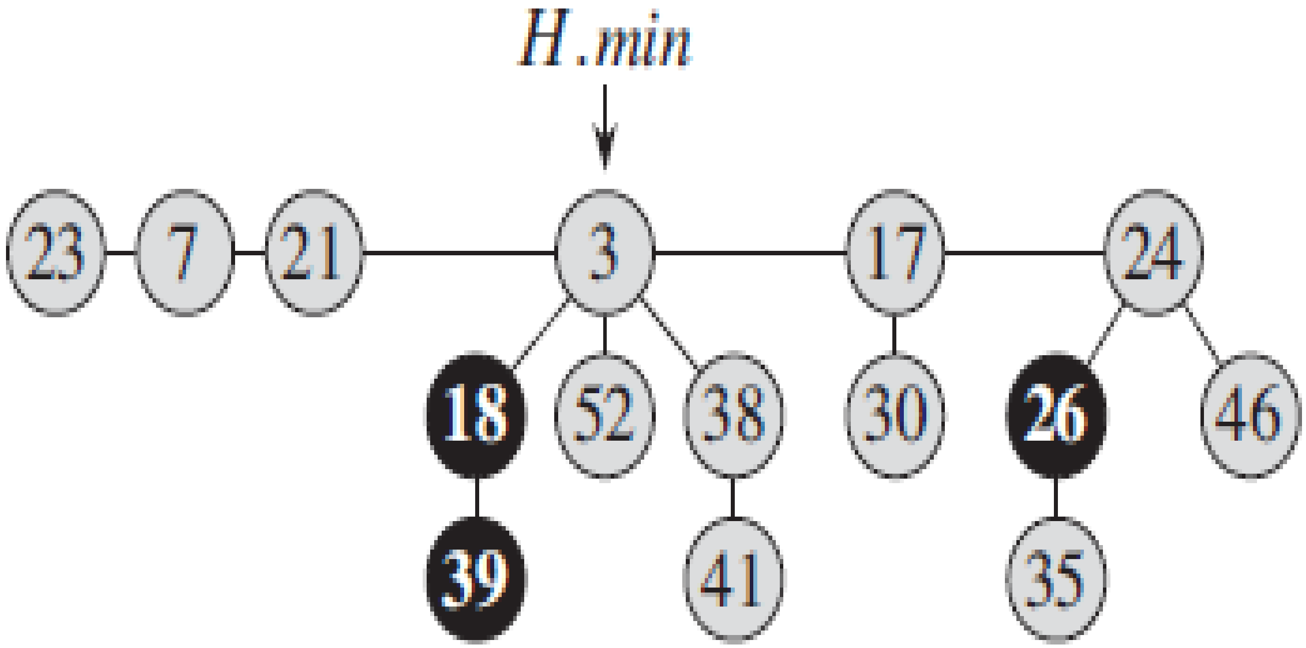
Inserting a node

Example: Insert a node with key 21 in the following Fibonacci heap.



Inserting a node

Solution: Fibonacci heap after inserting element 21 in the Fibonacci heap.



Inserting a node

The following procedure inserts node x into Fibonacci heap H, assuming that the node has already been allocated and that x:key has already been filled in.

```
FIB-HEAP-INSERT(H, x)
x.degree = 0
x.p = NIL
x.child = NIL
x.mark = FALSE
if H.min == NIL
    create a root list for H containing just x
    H.min = x
else insert x into H's root list
    if x.key < H.min.key
        H.min = x
H.n = H.n + 1
```

Inserting a node

To determine the amortized cost of FIB-HEAP-INSERT, let H be the input Fibonacci heap and H' be the resulting Fibonacci heap. Then,

$$t(H') = t(H) + 1$$

and

$$m(H') = m(H),$$

and the increase in potential = $\Phi(H') - \Phi(H)$

$$= t(H') + 2m(H') - (t(H) + 2m(H))$$

$$= t(H) + 1 + 2m(H) - (t(H) + 2m(H))$$

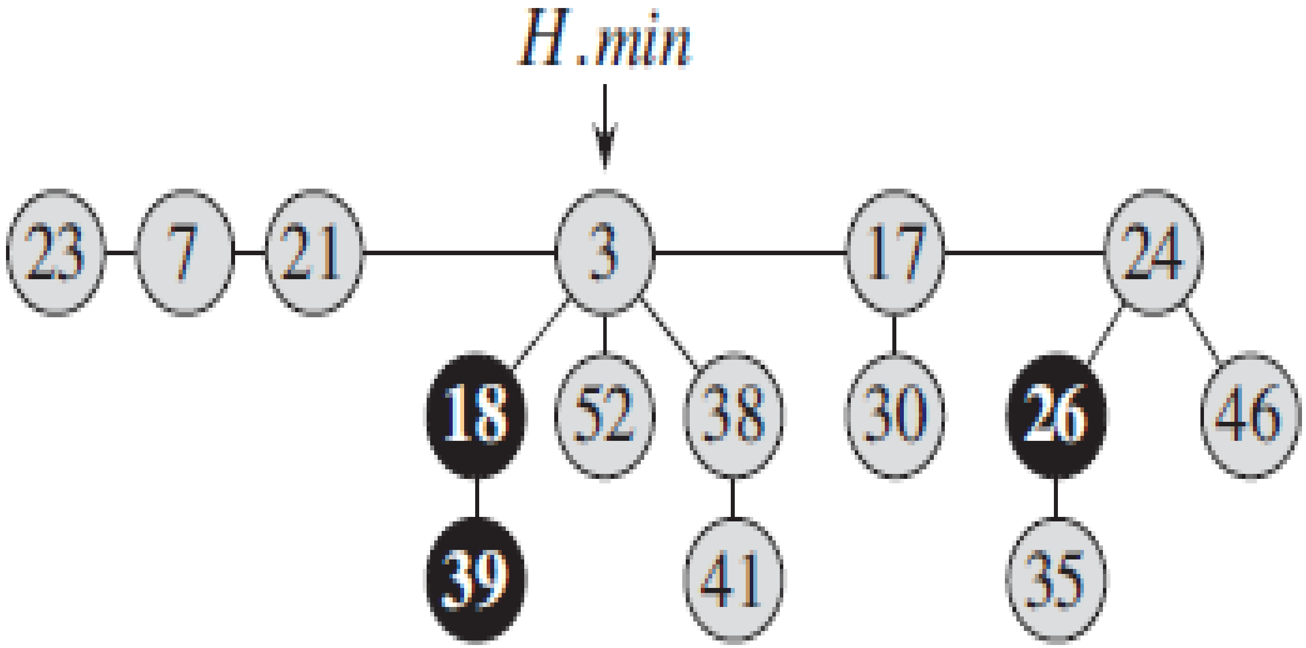
$$= 1$$

The actual cost = O(1), therefore

Amortized cost = Actual cost + $\Phi(H') - \Phi(H)$

$= O(1) + 1 =$ **O(1)**

Finding the minimum node



The minimum node of a Fibonacci heap H is given by the pointer H.min, so we can find the minimum node in O(1) actual time. Because the potential of H does not change, therefore the amortized cost of this operation is equal to its O(1) actual cost.

Uniting two Fibonacci heaps

The following procedure unites Fibonacci heaps H1 and H2, destroying H1 andH2 in the process. It simply concatenates the root lists of H1 and H2 and then determines the new minimum node. Afterward, the objects representing H1 and H2 will never be used again.

```
FIB-HEAP-UNION(H1, H2)
H = MAKE-FIB-HEAP()
H.min = H1.min
concatenate the root list of H2 with the root list of H
if (H1.min == NIL) or (H2.min ≠ NIL and H2.min.key < H1.min.key)
    H.min = H2.min
H.n = H1.n + H2.n
return H
```

Uniting two Fibonacci heaps

Amortied cost:

The change in potential function

$\Phi(\mathrm{H}) - (\Phi(\mathrm{H}_1) + \Phi(\mathrm{H}_2))$

$$= \mathrm{t(H)} + 2\mathrm{m(H)} - (\mathrm{t}(\mathrm{H}_1) + 2\mathrm{m}(\mathrm{H}_1) + \mathrm{t}(\mathrm{H}_2) + 2\mathrm{m}(\mathrm{H}_2))$$

$$= 0$$

Because t(H) = $\mathrm{t}(\mathrm{H}_1) + \mathrm{t}(\mathrm{H}_2)$ and m(H) = $\mathrm{m}(\mathrm{H}_1) + \mathrm{m}(\mathrm{H}_2)$

Therefore the amortized cost

= actual cost + change in potential

= O(1) + 0

= O(1)