ECAP770

Advance Data Structures

Ashwani Xumar
Assistant Professor

Learning Outcomes

After this lecture, you will be able to

- understand Fibonacci Heap

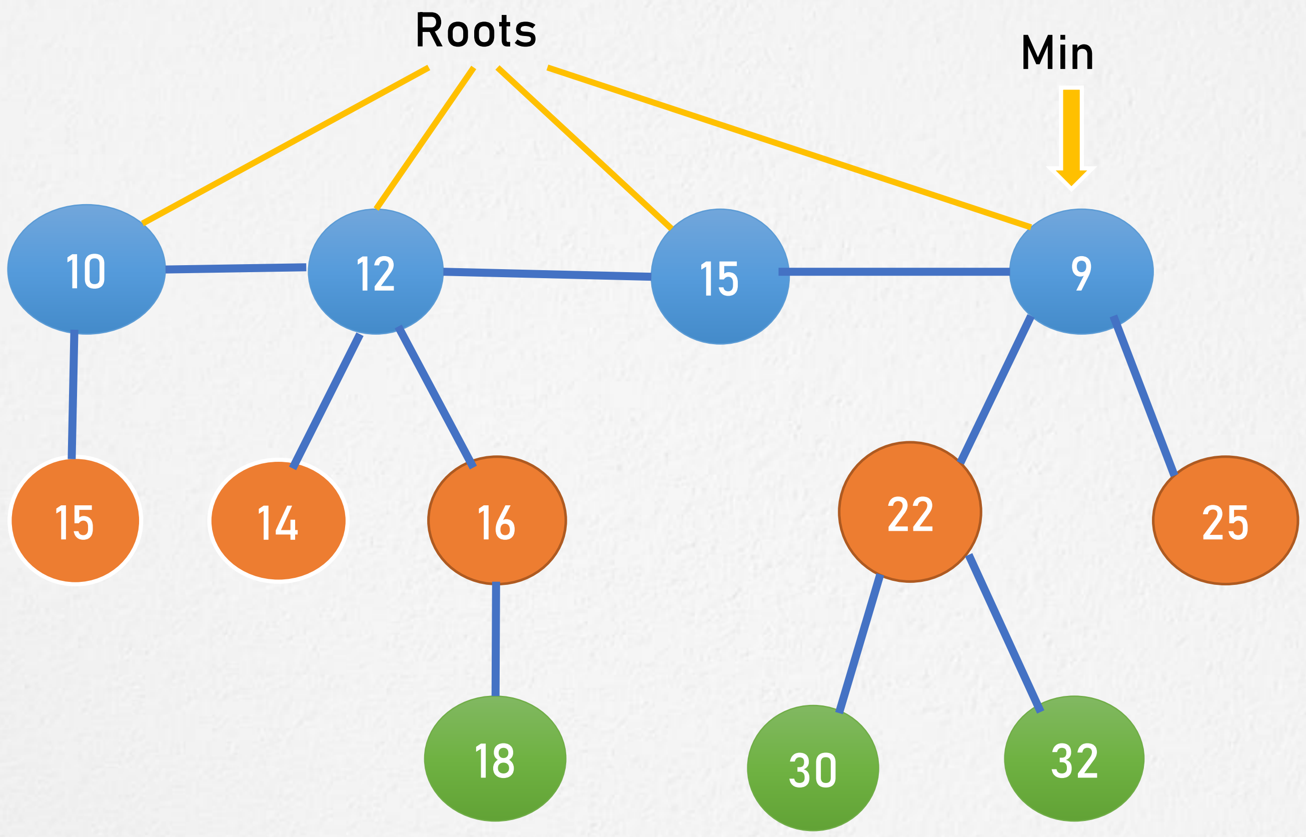
Fibonacci Heap

- Fibonacci heap data structure is collection of trees which follow min heap or max heap property.
- In a Fibonacci heap, a node can have more than two children or no children at all.

Properties of a Fibonacci Heap

- A pointer is maintained at the minimum element node.
- The trees within a Fibonacci heap are unordered but rooted.
- It is a set of min heap-ordered trees.
- It consists of a set of marked nodes.

Fibonacci Heap



Fibonacci Heap

- The child nodes of a parent node are connected to each other through a circular doubly linked list.
- Deleting a node from the tree takes O(1) time.
- The concatenation of two such lists takes O(1) time.

Fibonacci Heap

- Fibonacci heaps have a faster amortized running time than other heap types.
- Fibonacci heaps have a less rigid structure as compared to binomial heaps.
- Fibonacci heaps are used to implement the priority queue element in Dijkstra's algorithm.

Operations on a Fibonacci Heap

- Insertion
- Find Min
- Union
- Extract Min
- Decrease a key
- Delete a node

Insertion

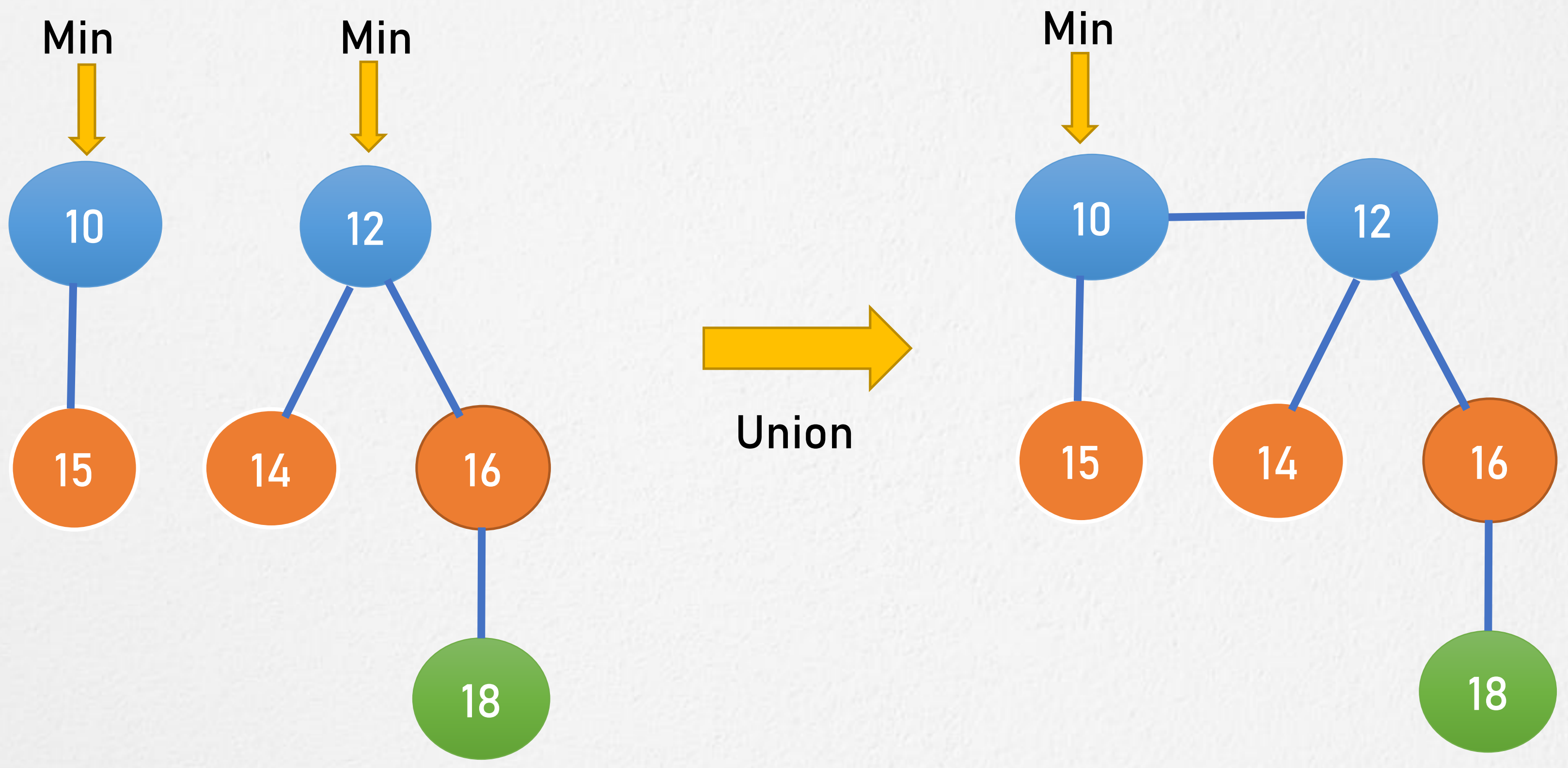
- Create a new node for the element.
- Check if the heap is empty.
- If the heap is empty, set the new node as a root node and mark it min.
- Else, insert the node into the root list and update min.

Union

Steps for Union of two Fibonacci heaps.

- Concatenate the roots of both the heaps.
- Update min by selecting a minimum key from the new root lists.

Union: Example



Extract Min

- In extract min minimum value is removed from the heap and the tree is re-adjusted.
- Steps for Extract Min
  1) Delete the min node.
  2) Set the min-pointer to the next root in the root list.
  3) Create an array of size equal to the maximum degree of the trees in the heap before deletion.

Extract Min

4) Do the following (steps 5-7) until there are no multiple roots with the same degree.
5) Map the degree of current root (min-pointer) to the degree in the array.
6) Map the degree of next root to the degree in array.

Extract Min

4) If there are more than two mappings for the same degree, then apply union operation to those roots such that the min-heap property is maintained (i.e. the minimum is at the root).

Decrease Key: Algorithm

- Decrease the value of the node ‘x’ to the new chosen value.
- CASE 1 - If min heap order not violated,
  - Update min pointer if necessary.

Decrease Key: Algorithm

- CASE 2 - If min heap order violated and parent of 'x' is unmarked,

  Cut off the link between 'x' and its parent.

  Mark the parent of 'x'.

  Add tree rooted at 'x' to the root list and update min pointer if necessary.

Decrease Key

- CASE 3 - If min heap order is violated and parent of ‘x’ is marked,
    - Cut off the link between ‘x’ and its parent p[x].
    - Add ‘x’ to the root list, updating min pointer if necessary.
    - Cut off link between p[x] and p[p[x]].
    - Add p[x] to the root list, updating min pointer if necessary.
    - If p[p[x]] is unmarked, mark it.
    - Else, cut off p[p[x]] and repeat steps 4.2 to 4.5, taking p[p[x]] as ‘x’.

Deleting a Node

- This process makes use of decrease-key and extract-min operations. The following steps are followed for deleting a node.
  1. Let k be the node to be deleted.
  2. Apply decrease-key operation to decrease the value of k to the lowest possible value (i.e. $-\infty$).
  3. Apply extract-min operation to remove this node.

That’s all for now…