ECAP770

Advance Data Structures

Ashwani Xumar

Assistant Professor

Learning Outcomes

After this lecture, you will be able to

- understand binomial heaps

Binomial Heap

- A binomial Heap is a collection of Binomial Trees that satisfies the heap properties, i.e., min heap. it supports quicker merging of two heaps in O (log n).

Binomial Heap

- A binomial Heap is a collection of Binomial Trees that satisfies the heap properties, i.e., min heap. it supports quicker merging of two heaps in O (log n).
- A min heap is a heap in which each node has a value lesser than the value of its child nodes.

Binomial Tree

- A Binomial tree is a tree in which B_k is an ordered tree defined recursively, where k is defined as the order of the binomial tree.

Binomial Tree

- A Binomial tree is a tree in which B_k is an ordered tree defined recursively, where k is defined as the order of the binomial tree.
- The binomial tree B_0 consists of a single node. The binomial tree B_k consists of two binomial trees B_{k-1} that are linked together, the root of one is the leftmost child of the root of the other.

Binomial Tree

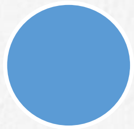
- If the binomial tree is represented as B0 then the tree consists of a single node.

Binomial Tree

- If the binomial tree is represented as B0 then the tree consists of a single node.
- In general terms, B_k consists of two binomial trees, i.e., Bk-1 and Bk-1 are linked together in which one tree becomes the left sub tree of another binomial tree.

Binomial Tree B_0

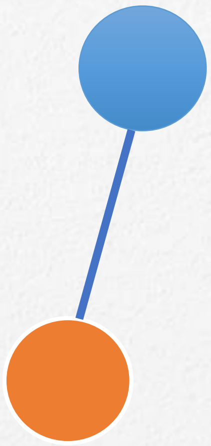
- If B0, k= 0, there would be only one node in the tree



B_0

Binomial Tree B_1

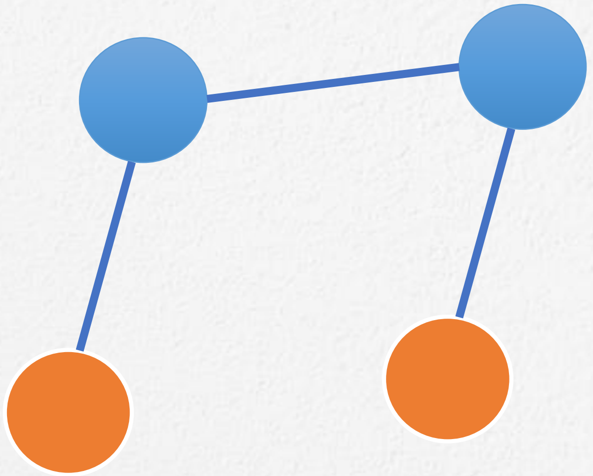
- If B1, k= 1, means k-1 equal to 0. Therefore, there would be two binomial trees of B0 in which one B0 becomes the left sub tree of another B0.



B_1

Binomial Tree B_2

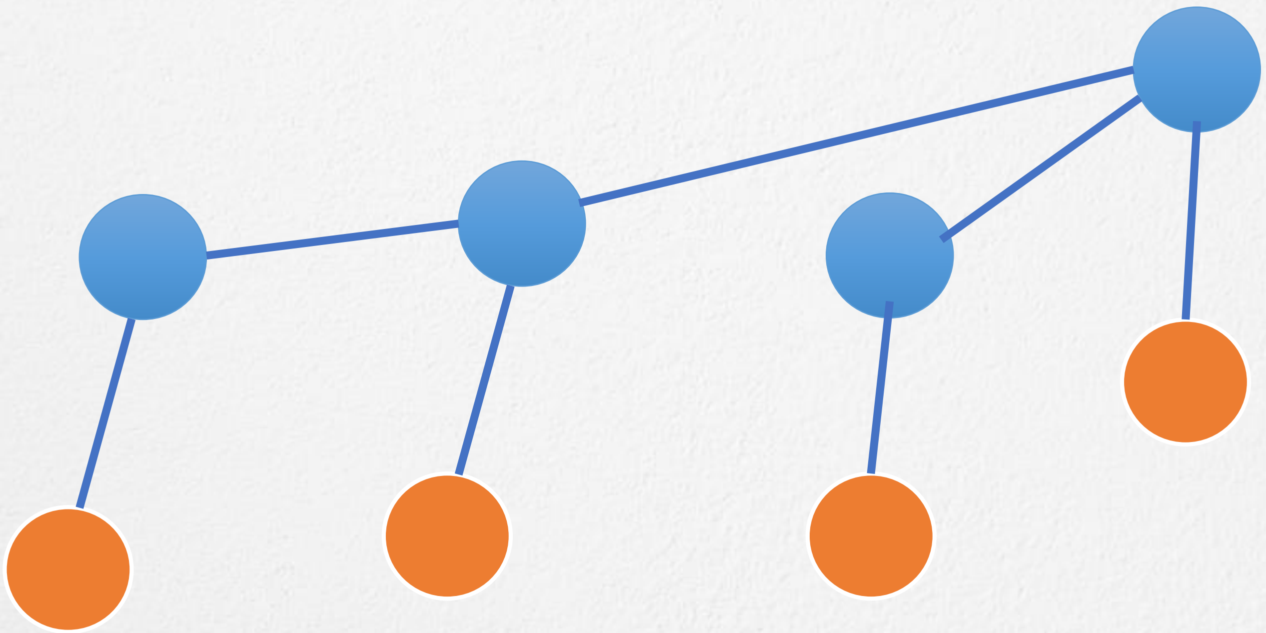
- If B2, k= 2, means k-1 equal to 1. Therefore, there would be two binomial trees of B1 in which one B1 becomes the left sub tree of another B1.



B_2

Binomial Tree B3

- If B3 , k= 3, means k-1 equal to 2. Therefore, there would be two binomial trees of B2 in which one B2 becomes the left sub tree of another B2.



Operations of Binomial Heap

- Union of two binomial heap
- Finding the minimum key
- Creating a new binomial heap
- Inserting a node
- Extracting minimum key
- Decreasing a key
- Deleting a node

Union of two binomial heap

- Case 1: If degree[x] is not equal to degree[next x] then move pointer ahead.
- Case 2: if degree[x] = degree[next x] = degree[sibling(next x)] then

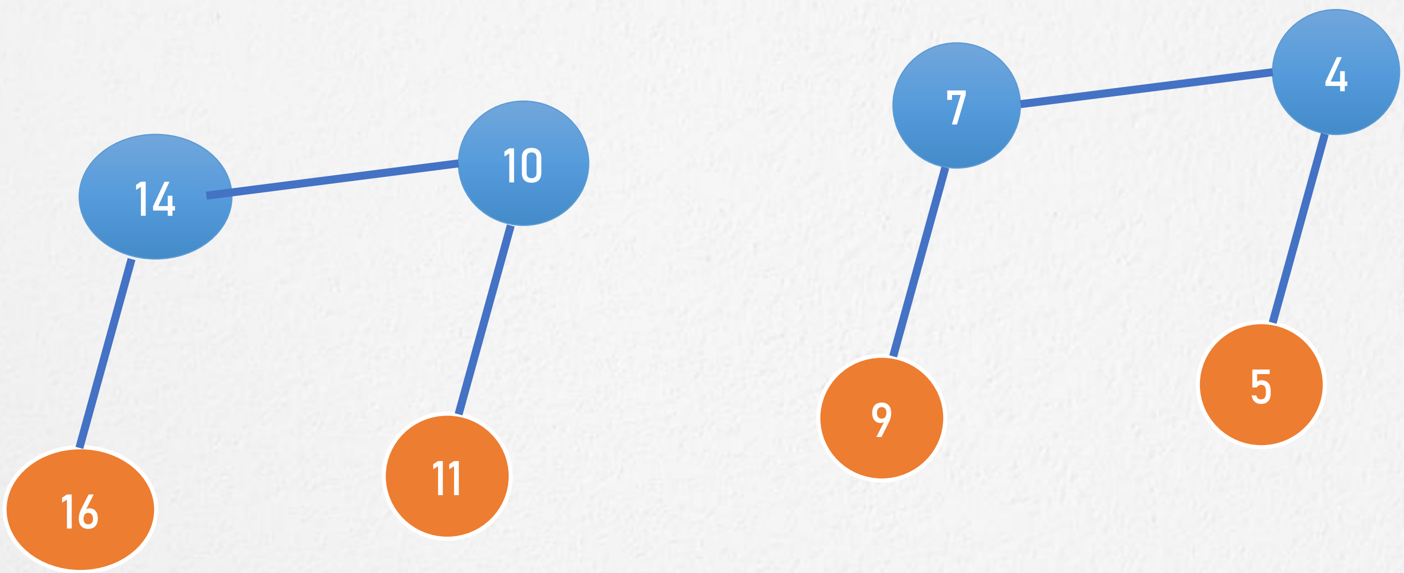
  Move pointer ahead.

Union of two binomial heap

- Case 3: If degree[x] = degree[next x] but not equal to degree[sibling[next x]] and key[x] < key[next x] then remove [next x] from root and attached to x.
- Case 4: If degree[x] = degree[next x] but not equal to degree[sibling[next x]] and key[x] > key[next x] then remove x from root and attached to [next x].

Union of two binomial heap

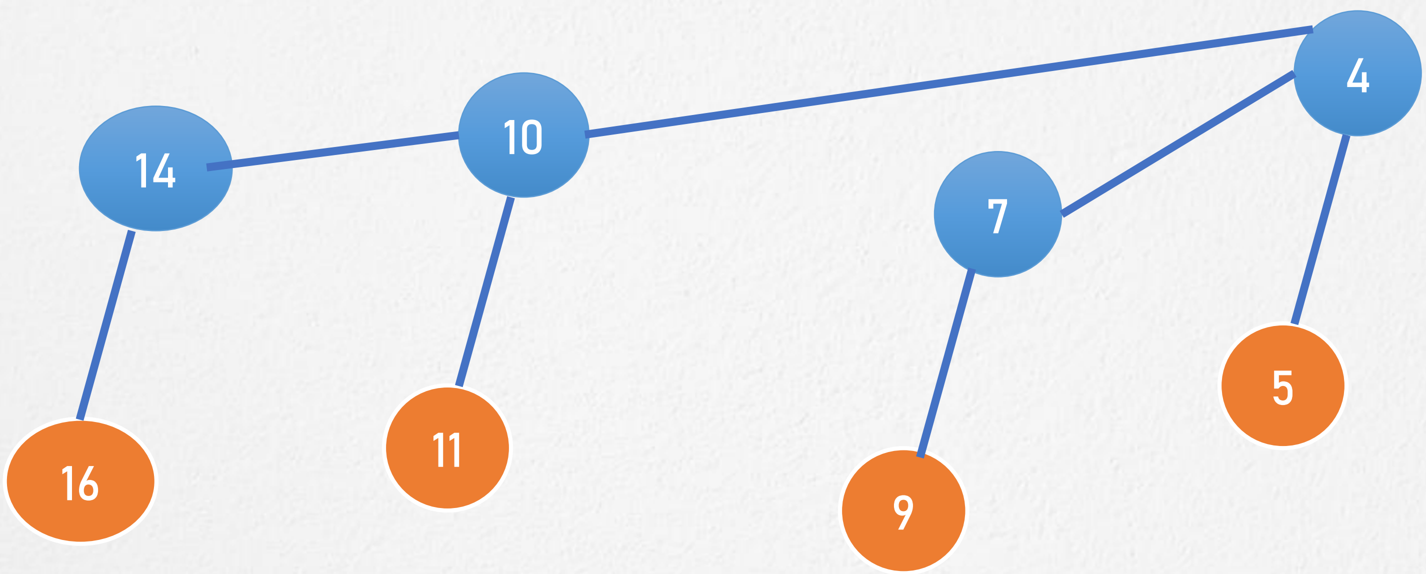
Comparison of root keys of H1 and H2



H1

H2

Union of two binomial heap



(H1 and H2)

Find minimum

- To find the minimum element of the heap, find the minimum among the roots of the binomial trees.
- It requires O(Logn) time. It can be optimized to O(1) by maintaining a pointer to minimum key root.

Decrease Key

- We compare the decreases key with it parent and if parent’s key is more, we swap keys and recur for the parent.
- Swap process stop when we either reach a node whose parent has a smaller key or we hit the root node.
- Time complexity of decrease Key() is O(Logn).

Extract minimum key

- First find this element, remove it from its binomial tree, and obtain a list of its sub trees.
- Transform this list of sub trees into a separate binomial heap by reordering them from smallest to largest order. Then merge this heap with the original heap.
- This operation requires O(Logn) time.

That's all for now…