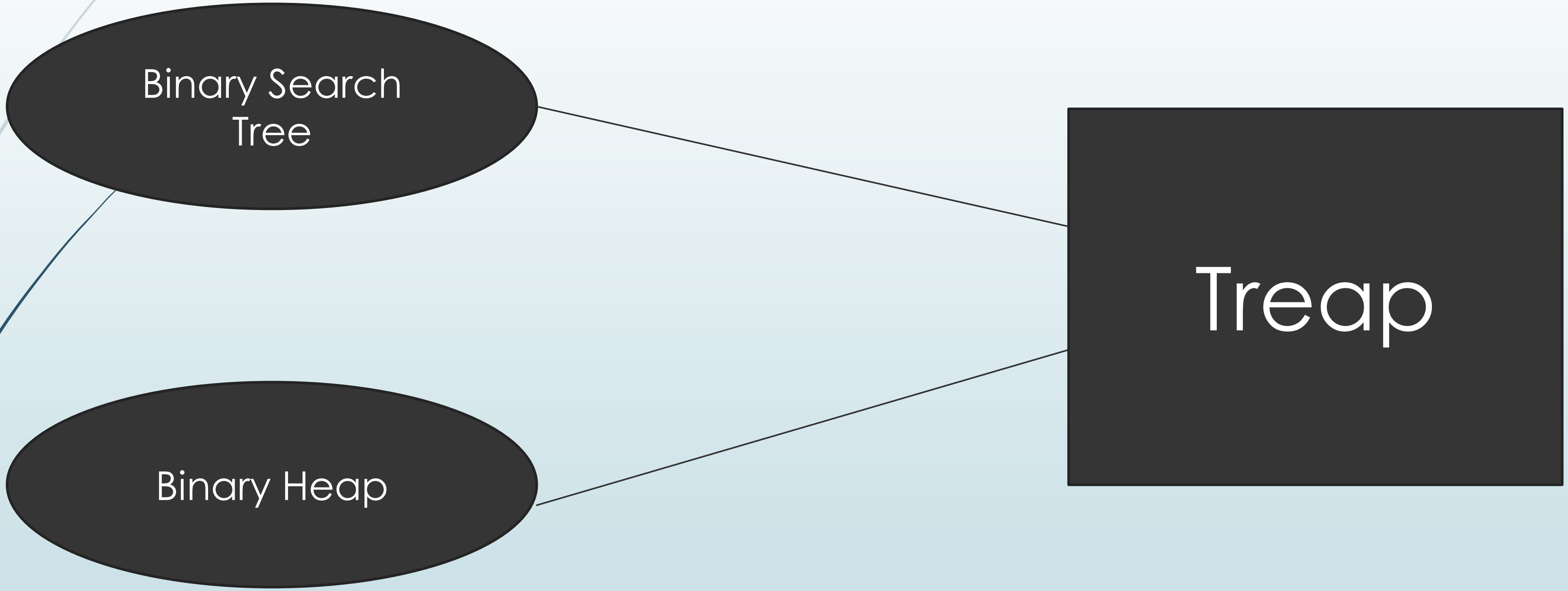
Treap

Pittsford Sutherland Programming Club

By Yizuo Chen, Yiyou Chen

Treap is another "Balanced binary search tree" – simplest one to write



Treap

- Use Binary search tree’s properties to make it easy to find certain elements.

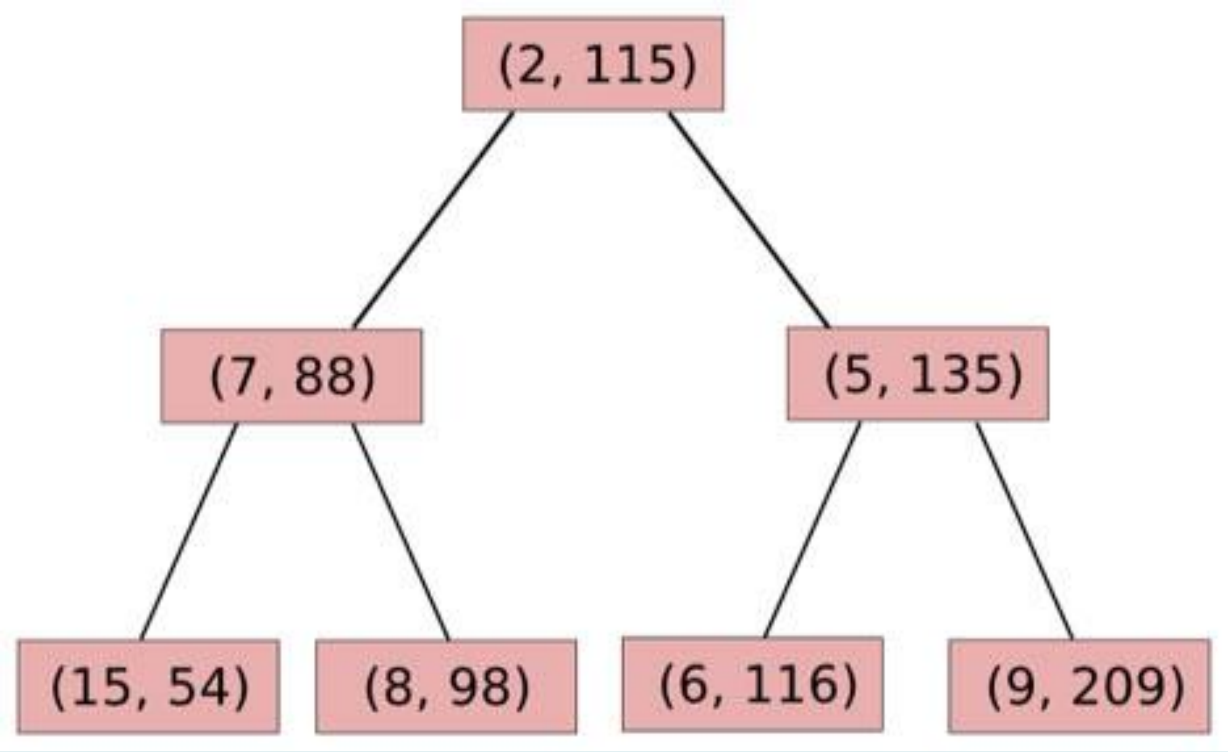
- Use Heap’s properties to make it “balance”.

How?

Inside the Treap

- Besides left child, right child, value, nodes in Treap have another elements called Priority value.
- The priority value for each node is given by a random number; however, these priority values must satisfy heap order.

Example:



In each node, the first value is the priority value, and the second value is the element value.

As you can see, Treap is a binary search tree + Min Heap

How to keep Treap “balanced”?

- Use rotations!

Insertion

- The same way Binary Search Tree Insertion to find the position to insert.
- When find the position to insert the node, compare the nodes' priorities with their children's. Use Rotation to keep parents' priorities less than their children's priorities.

Deletion

- The same way Binary Search Tree deletion to find the position of the node to remove.
- Compare the priorities of two children, use rotation to make the child with smaller priority to be the parent. (In order to satisfy the heap property)
- Remove the node from Treap.

Time Complexity: $O(\log n)$