Splay Tree

(1) Type of a BST
(2) Most of the operations are having O(log n)
(3) Not strictly balanced - Unlike AVL
(4) Easy to Implement
(5) Fast accessed to elements accessed recently.
(6) eg. Caches.

Find operation

- Like a standard BST search operation
- Rotations are performed when we find the given element we are looking for → it is going to be the root node. // Splaying
- Because in next search it can be accessed very fast even in O(1) time.

- There are 3 ways we can make it happen

(i) Zig-Zag | Zag-Zig situation

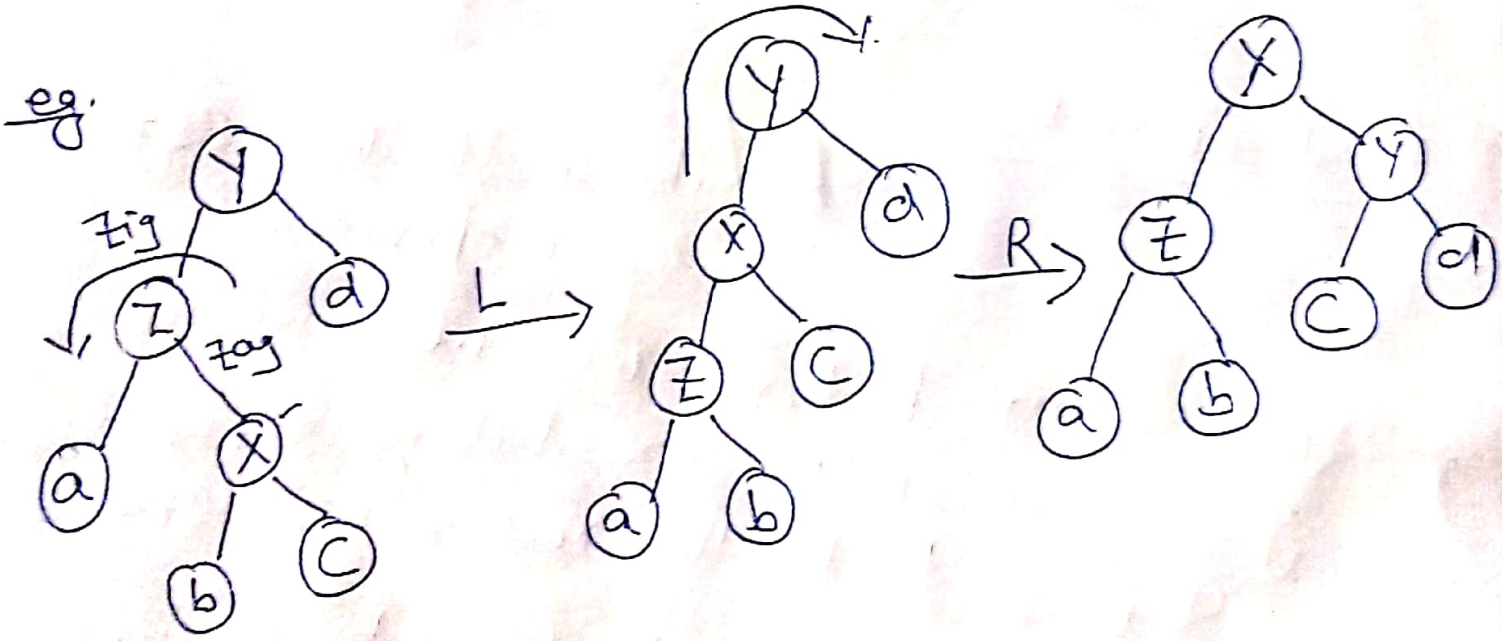
(ii) Zig-Zig | Zag-Zag

(iii) Zig | Zag situation

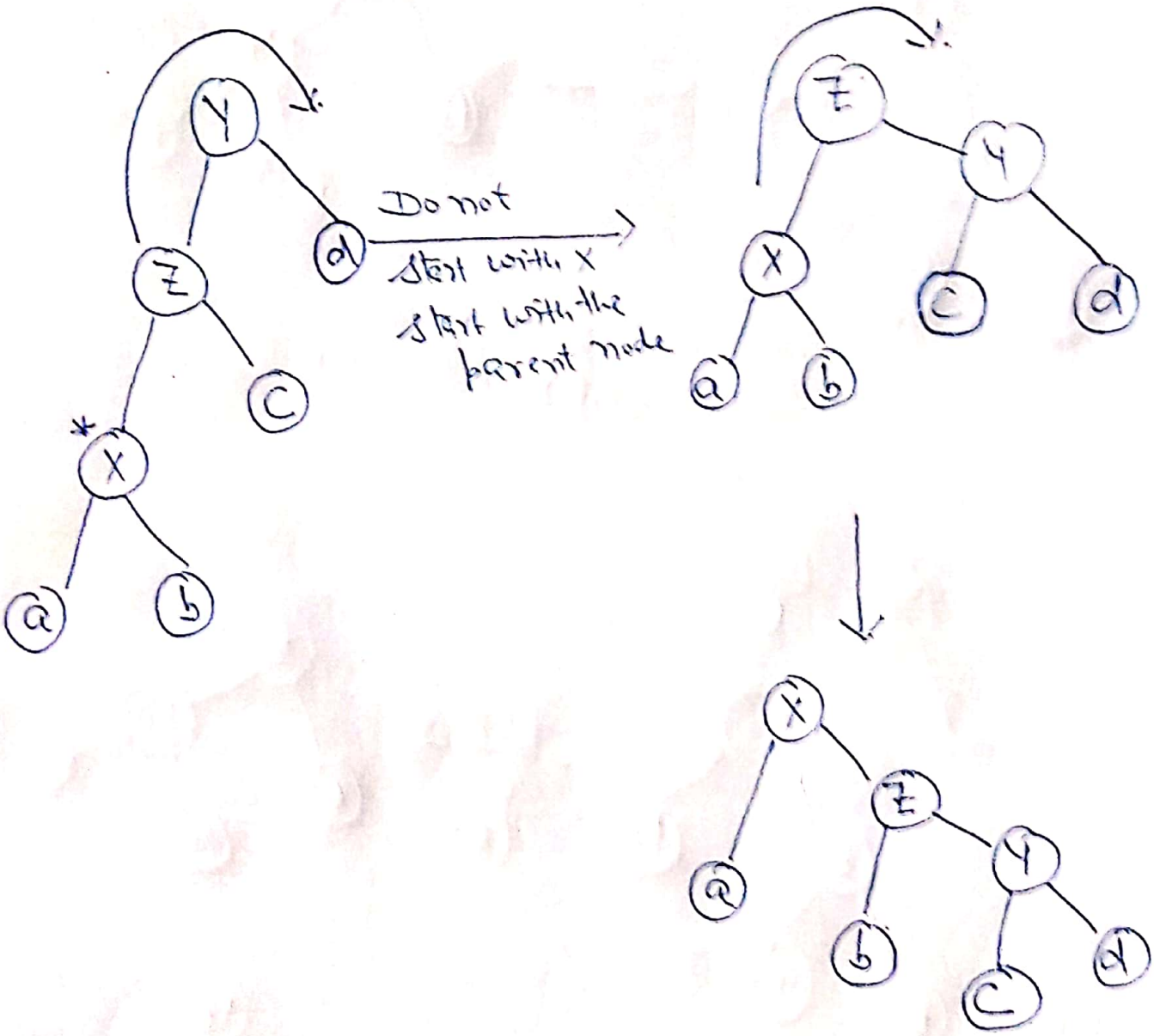
(i) Zig-Zag

- The given node x is a right child of a left child.
- (OR) the given node x is a left child of a right child.

eg.

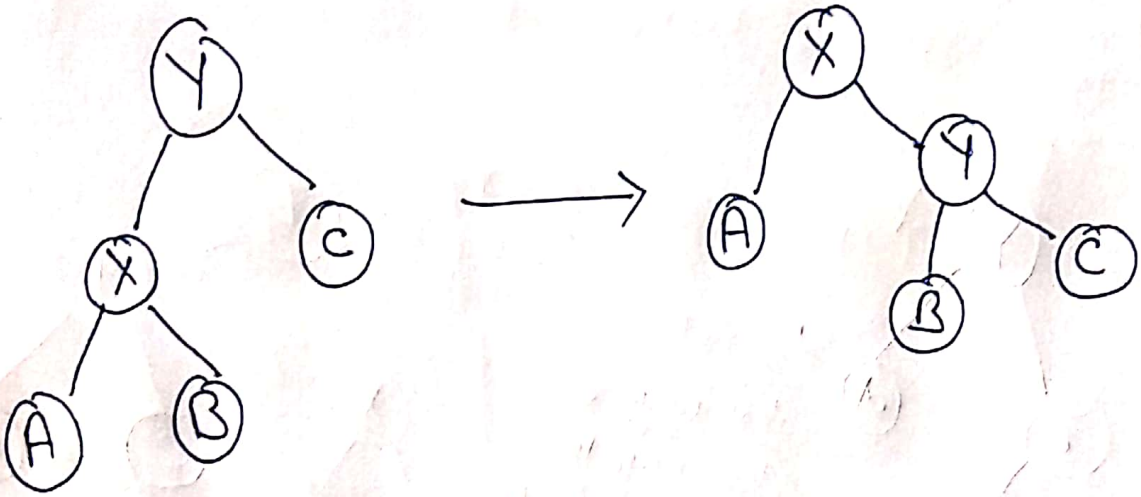


Zig-Zig

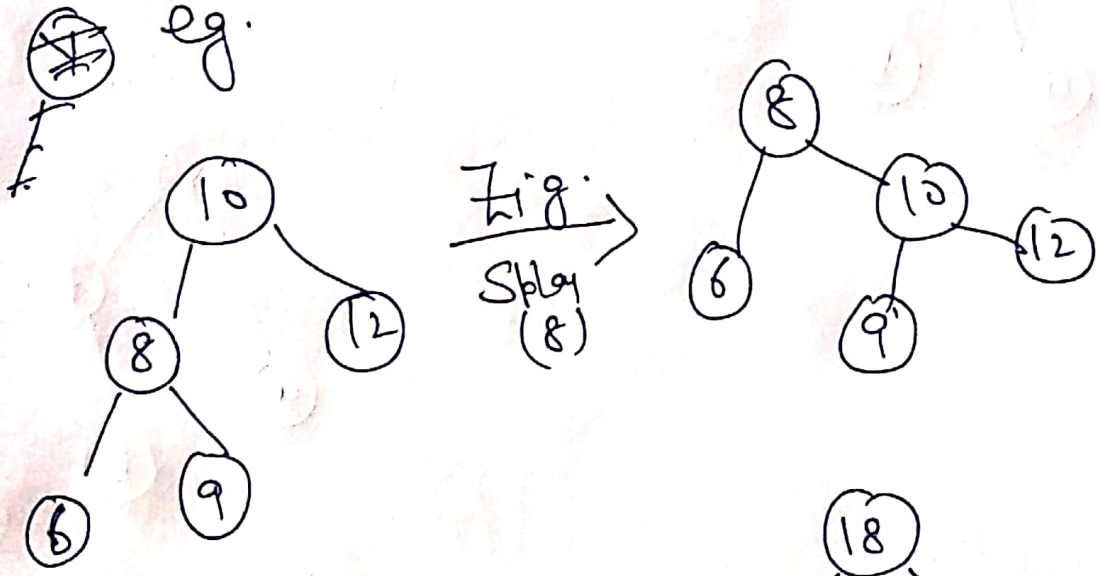


(iii) Zig Situation

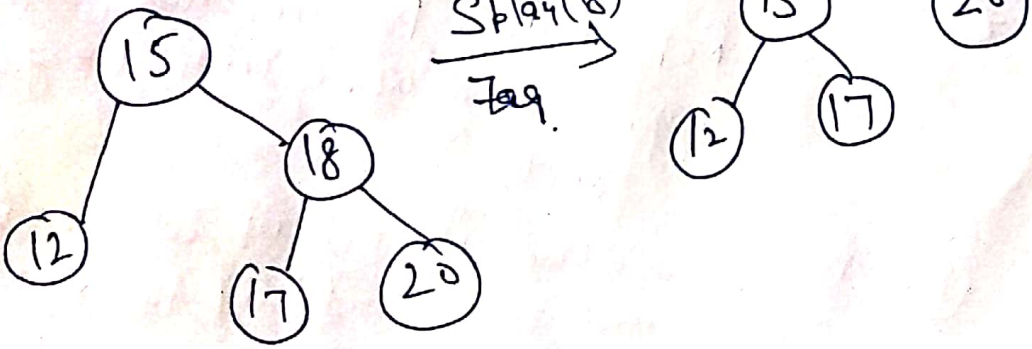
- Repeat the previous two steps again and again until we get to the root.
- Sometimes we end up at the left/Right ~~and right~~ child of the root. Then we have to make a single right/left rotation accordingly
- X is just the child of the root



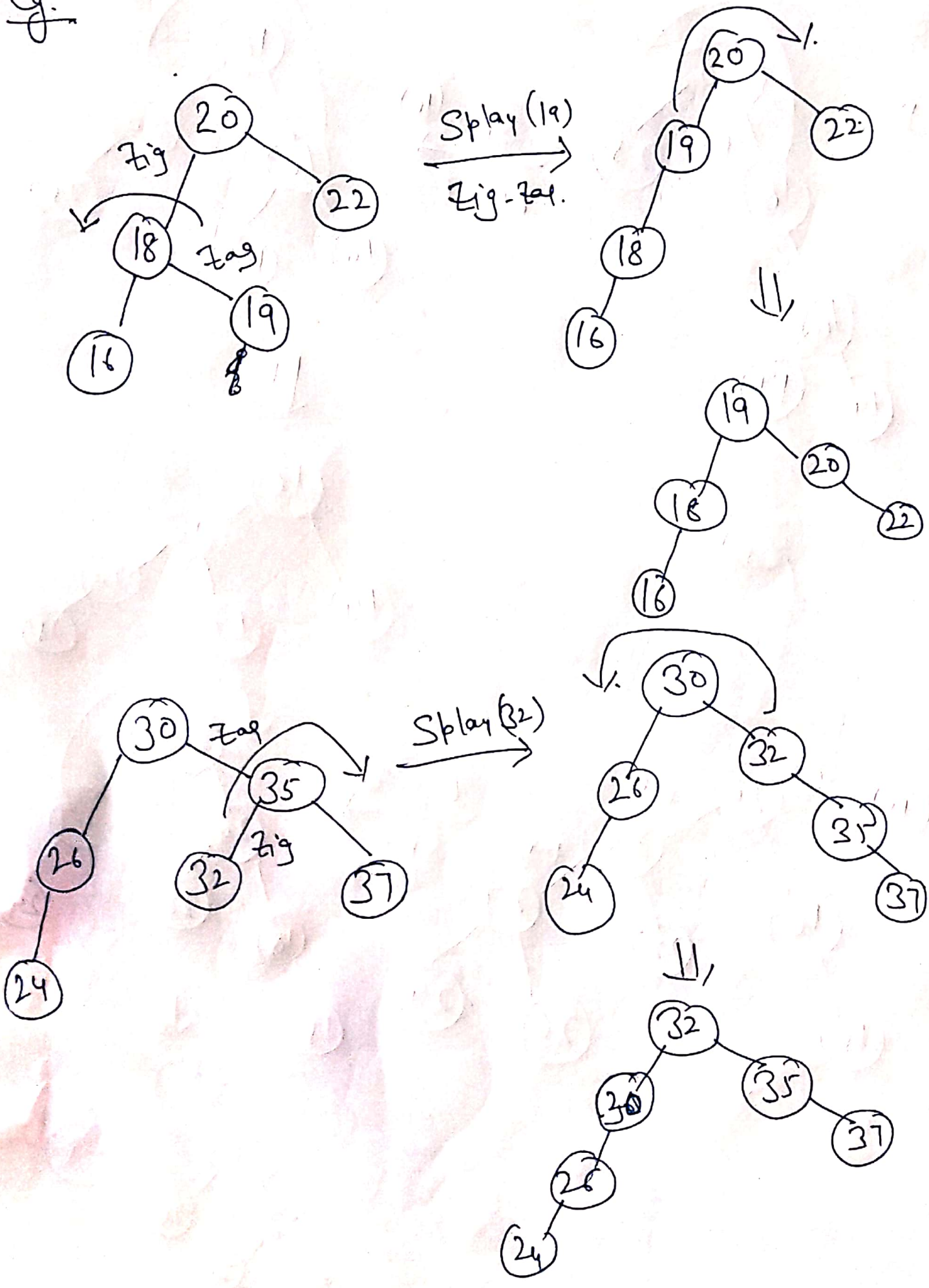
(OR)



(OR)

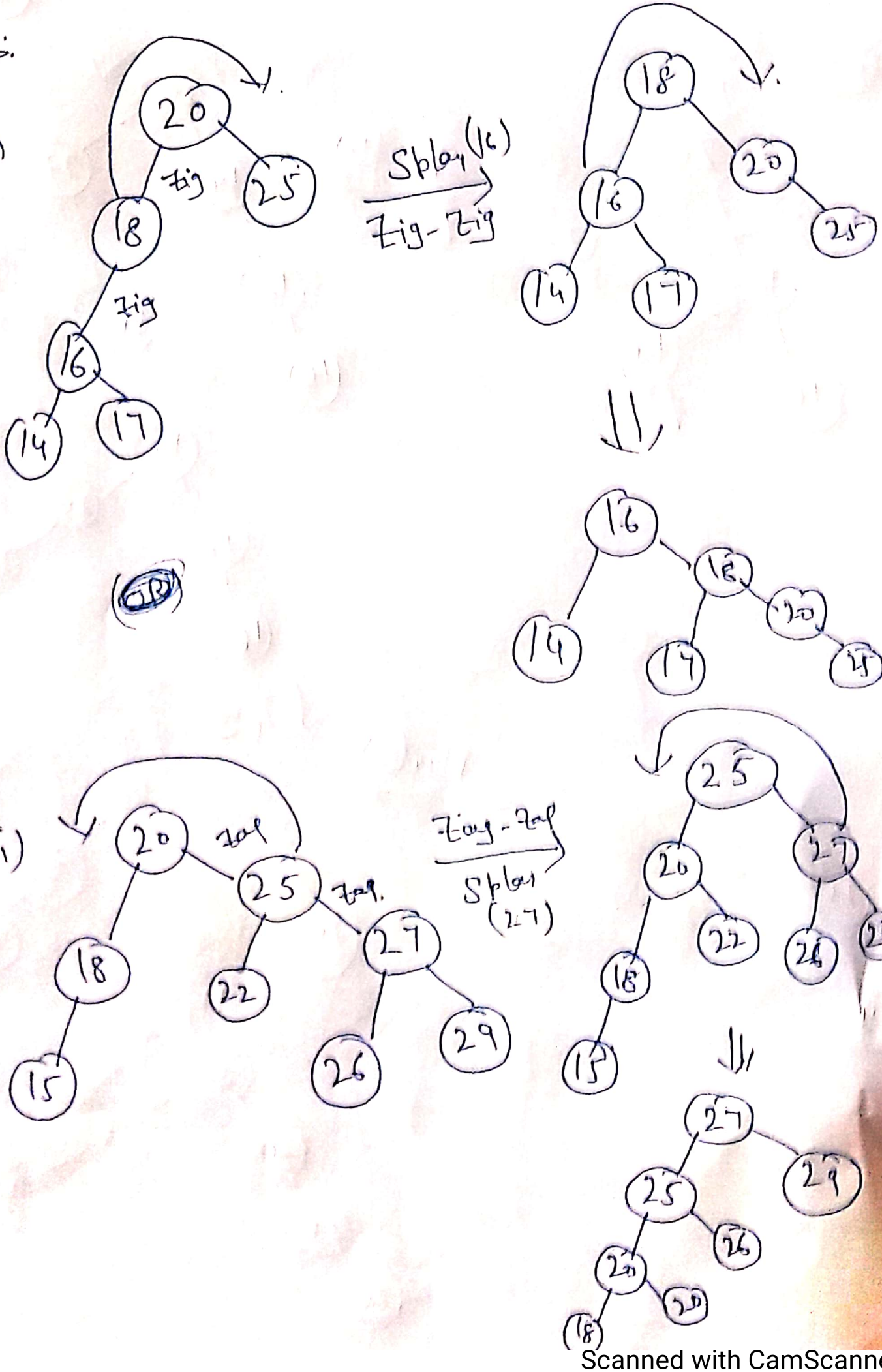


eg.



Eg:

(i)



(ii)