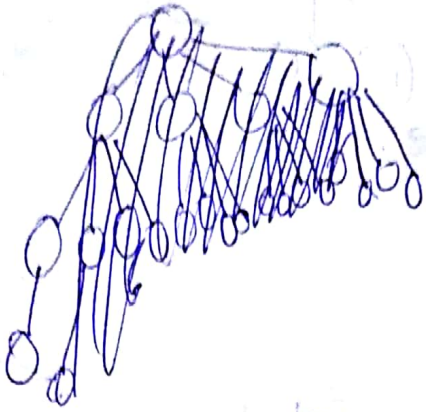
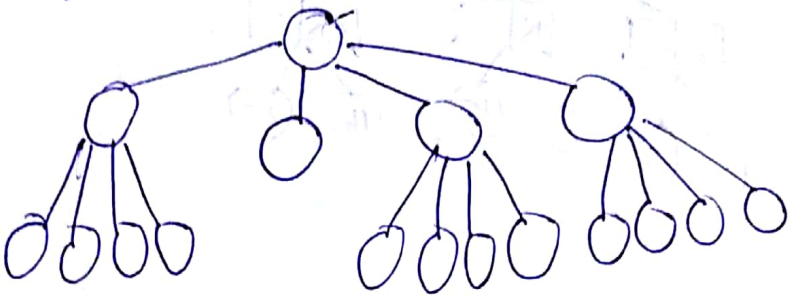
Quad Trees

→ Any node can have either 4 children or no children.

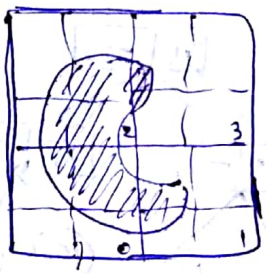


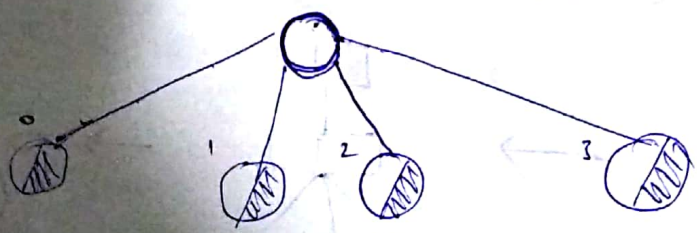


→ Oct Trees

→ Can have either 8 children or no children

→ Application of Quad tree





(no shaded region → no childr

Advantages: Storage of Image.

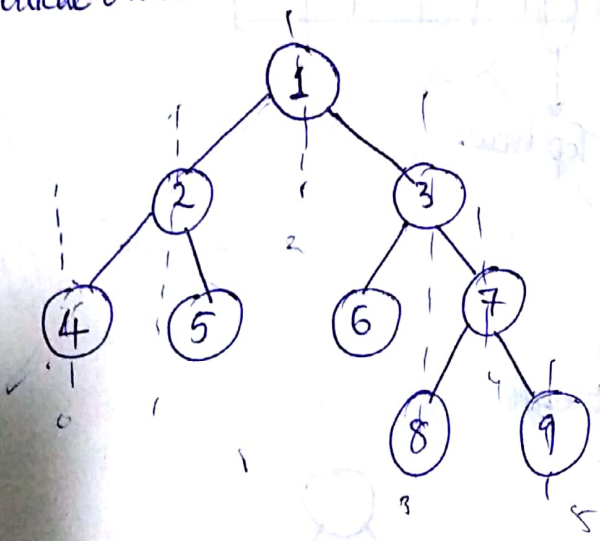
→ NOTE: Pixel, Voxel

1) First find the bounding square.
2) Divide the square into 4 equal size squares.
3) Check whether the squares are completely filled or partially filled or empty.
4) Any node representing a completely filled or empty square will not be expanded.
5) Further, divide all the partially filled squares into 4 equal squares.
6) If square size > pixel size goto step2.
7) At the leaf nodes, the 4 nodes representing squares than 50% filled are made completely filled, and the rest completely empty.

. . .

→ TUTORIAL

1) Print i) Top view of Binary Tree using Hashing.
ii) Vertical Order



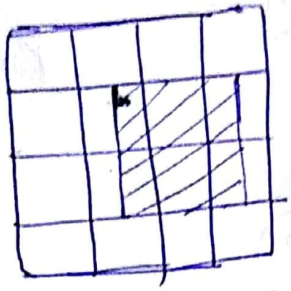
Top view: 4 2 1 3 7 9

Vertical order:
4
2
1 5 6
3 8
7
9

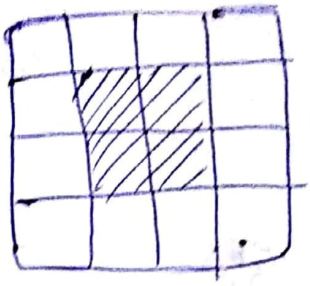
02/11/19

→ Quadtree (contd.)

→ Disadvantage



← Difficult to translate

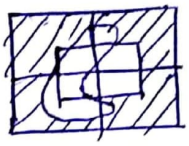


→ Advantage:- Storage ~~needs~~ → need to store the leaf infos only.

1) Area of the image = Σ area of each leaf × level (reversed) ?

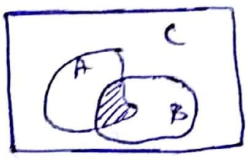
2) Complementation:- Make the filled leaves empty and vice versa.

3) Clipping :-

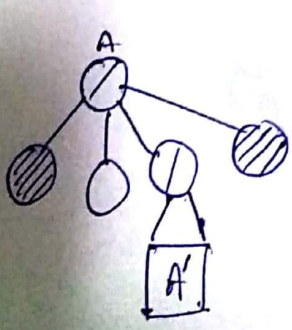


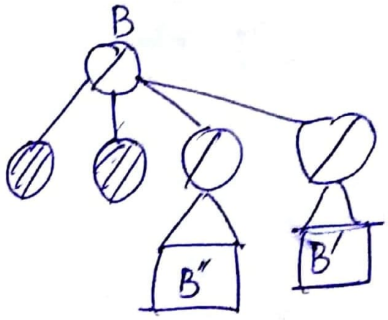
4) Boolean operations:-

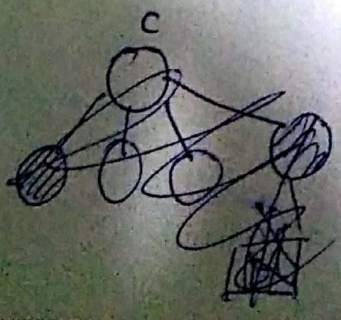
(a) Intersection :- $C = A \cap B$

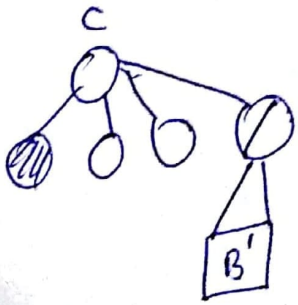


Cases :-









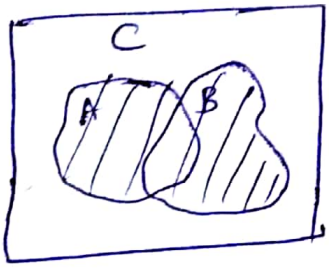
i) If at any equal level both A & B nodes are completely filled then include that node in C.

ii) If at any equal level A and B nodes are completely empty, the corresponding node in C will also be empty.

iii) When one of the nodes is completely filled and the other is semi-filled, attach the expansion of the semifilled node to C.

iv) If both nodes are semifilled, expand that corresponding nodes in C and go to ~~set~~ step 1.

(b) Union $C = A \cup B$



(c) Difference

$C = A - B = A \cap \overline{B}$