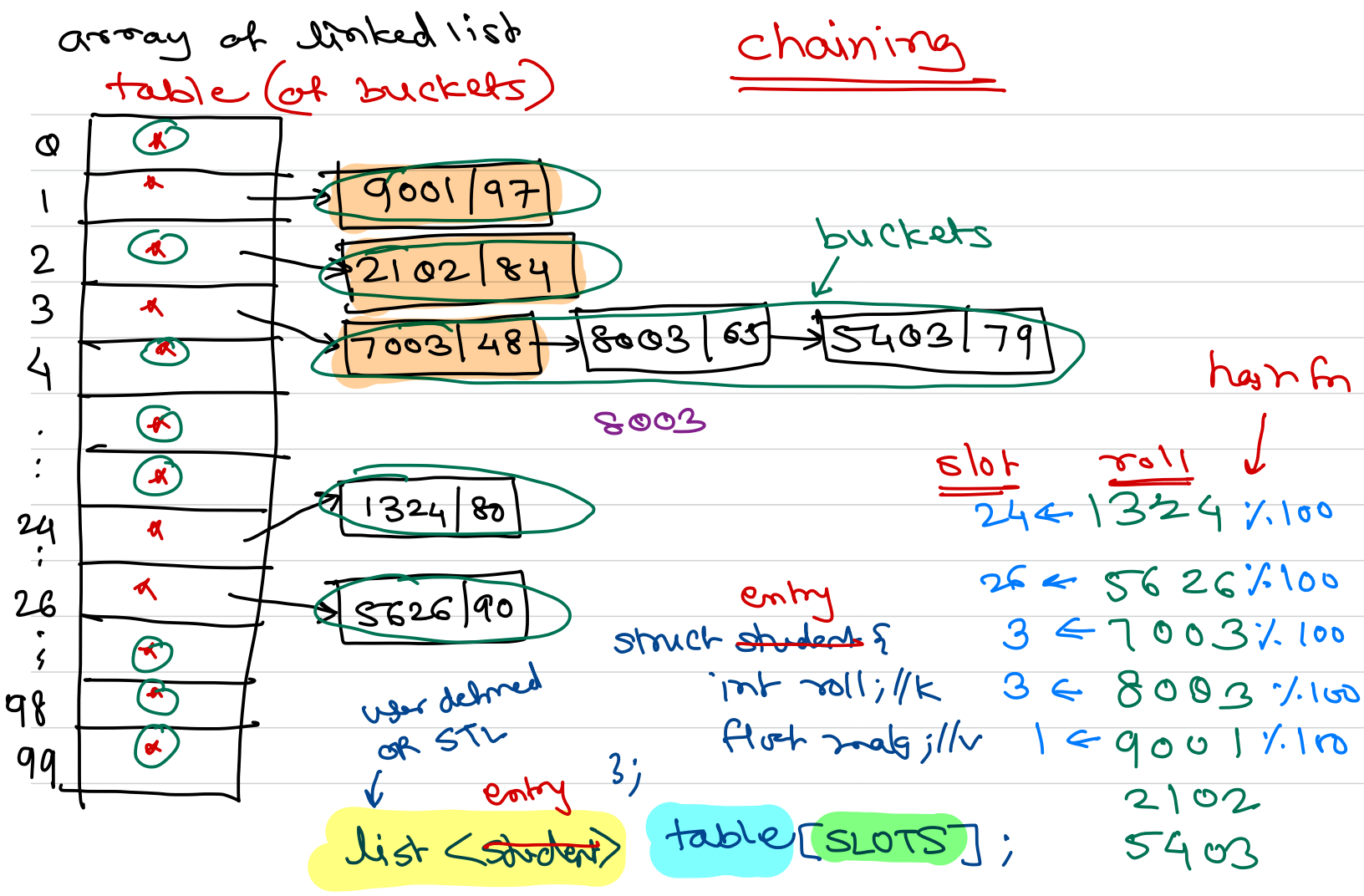
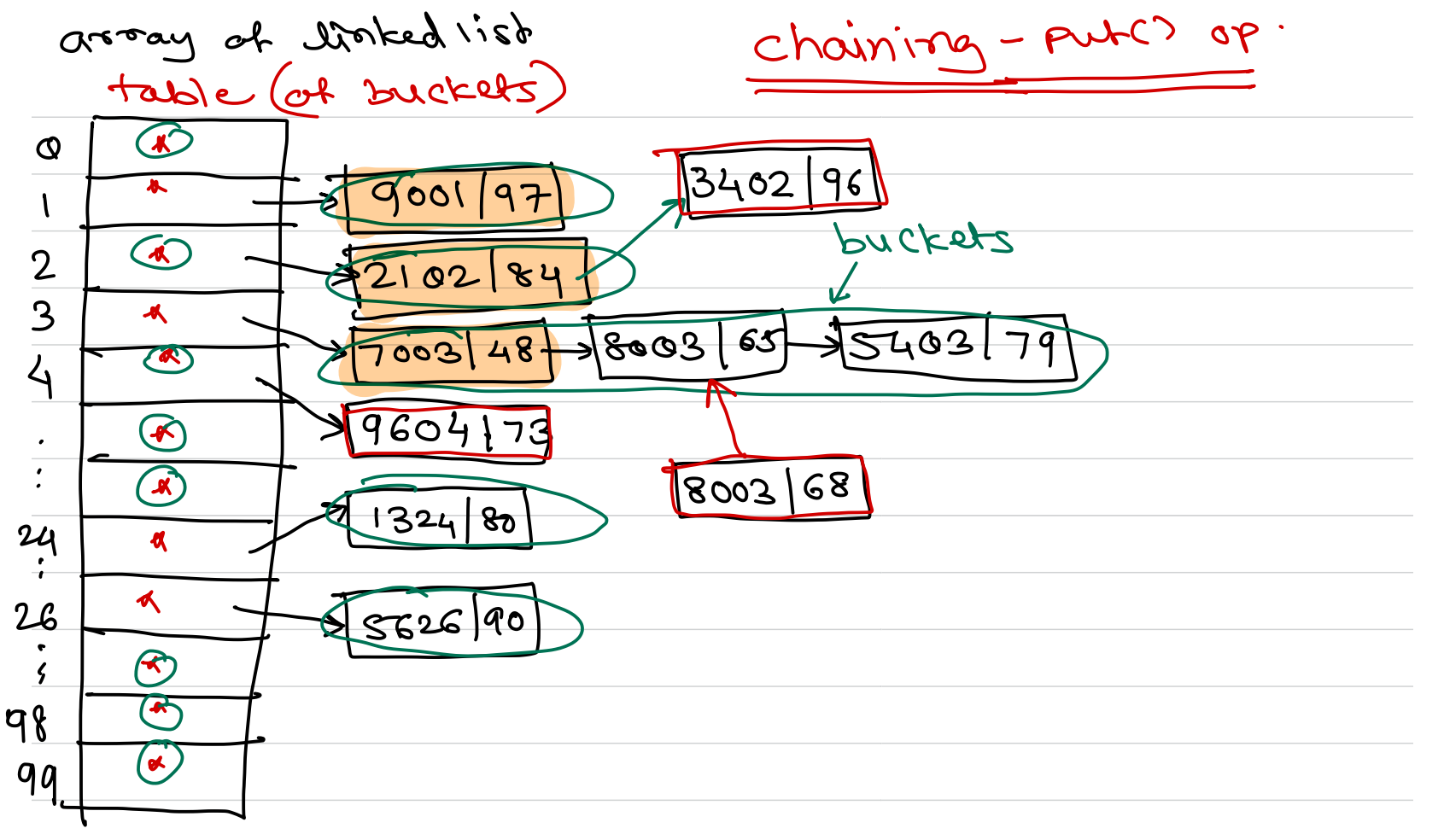
Data Structure & Algorithms

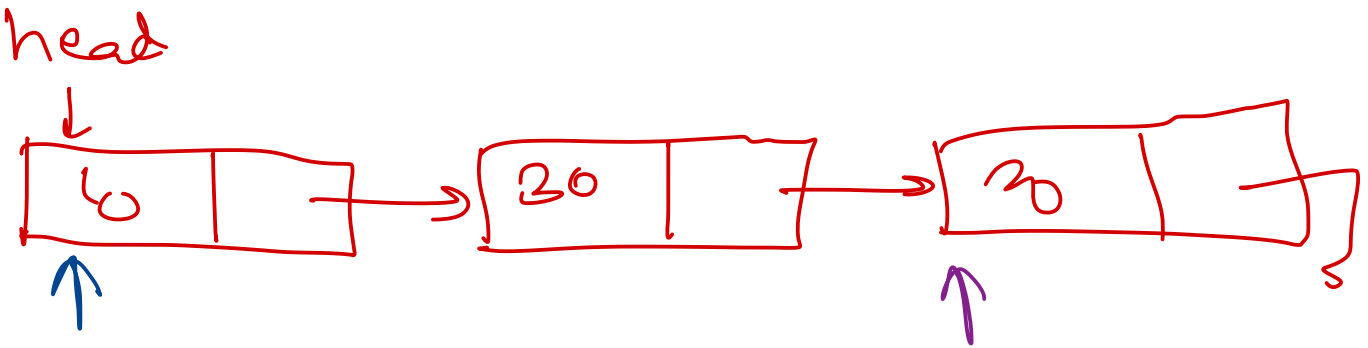
Sunbeam Infotech


array of linked list
table (of buckets)
chaining
0
1
2
3
4
24
26
98
99
9001 | 97
2102 | 84
7003 | 48
8003 | 65
5403 | 79
1324 | 80
5626 | 90
buckets
8003
hash fn
slot
roll
24 ← 1324 % 100
26 ← 5626 % 100
3 ← 7003 % 100
3 ← 8003 % 100
1 ← 9001 % 100
2102
5403
entry
struct student {
int roll; //k
float marks; //v
};
user defined
or STL
list <student>
entry
table[SLOTS];

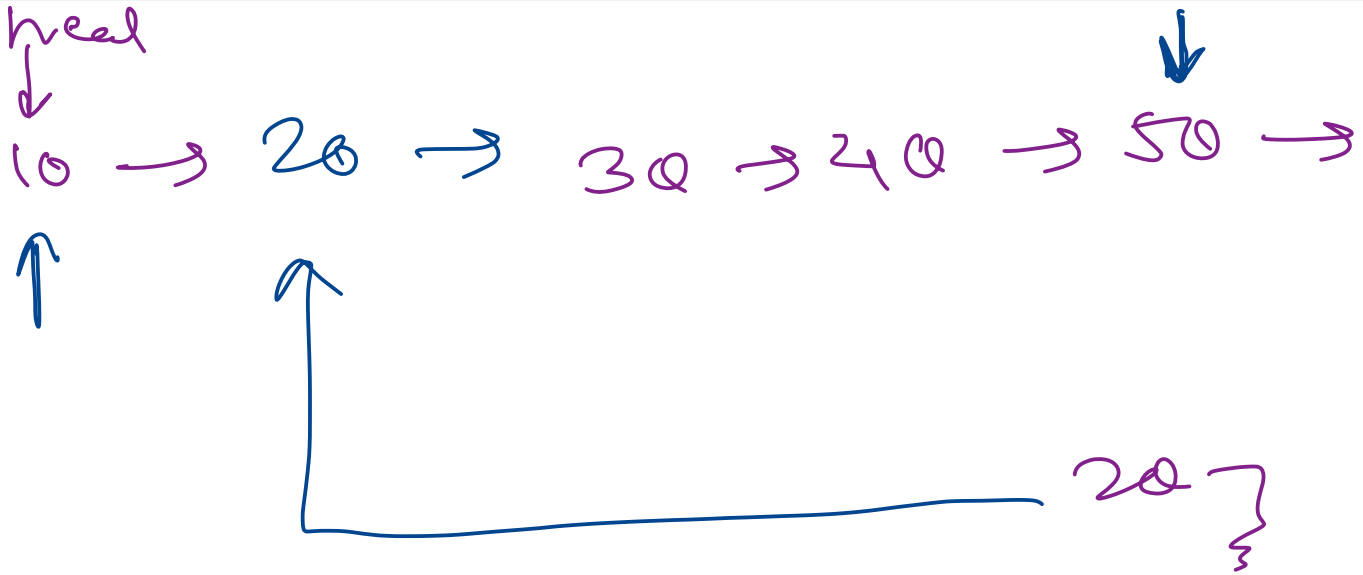

array of linked list
table (of buckets)
chaining - put() op.
0
1
2
3
4
24
26
98
99
9001 | 97
2102 | 84
7003 | 48
9604 | 73
1324 | 80
5626 | 90
3402 | 96
8003 | 65
5403 | 79
8003 | 68
buckets

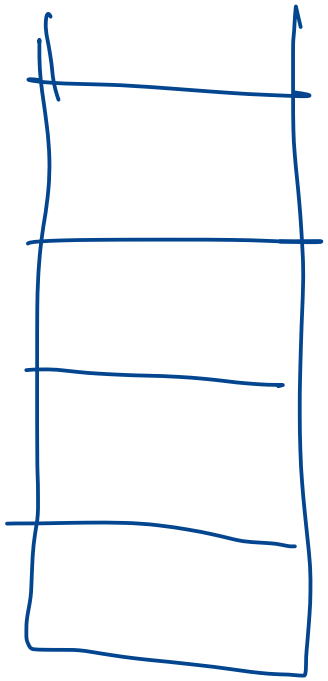
- sorting → selection or bubble or merge

```
node *i, *j;
for ( i = head ; i != NULL ; i = i->next ) {
    for ( j = i->next ; j != NULL ; j = j->next ) {
        if ( i->data > j->data )
            Swap ( i->data , j->data );
    }
}
```

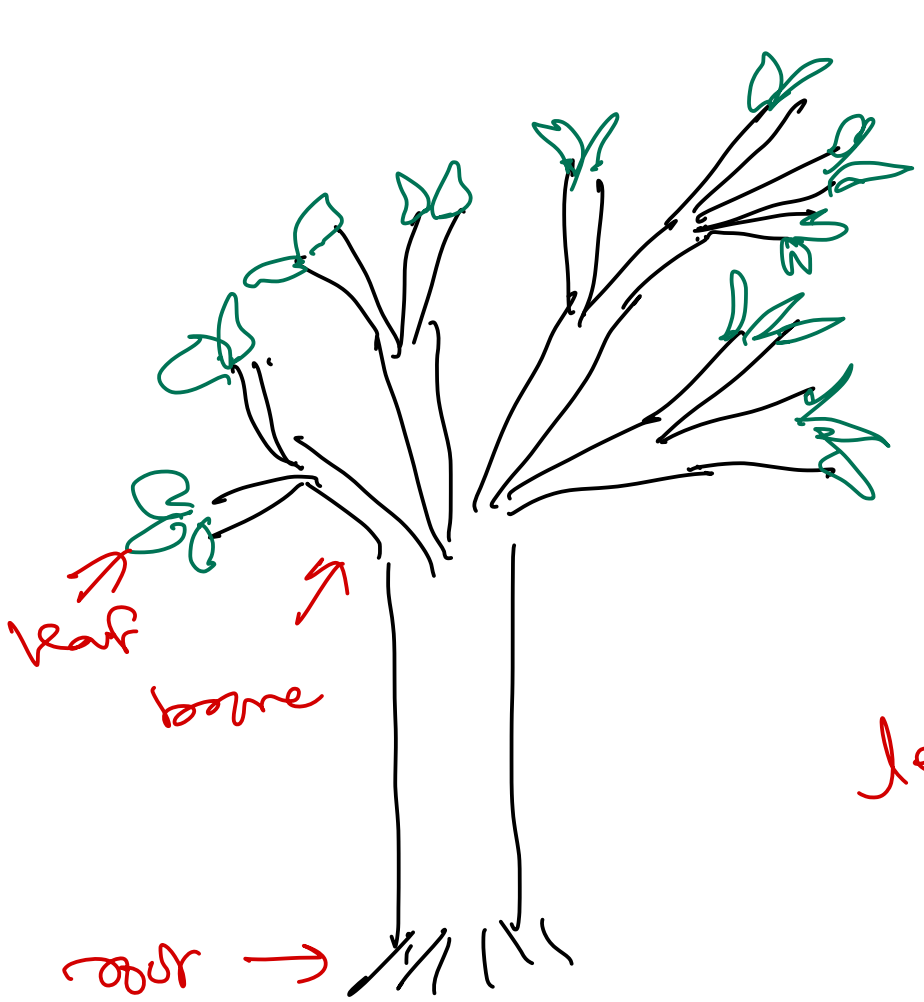


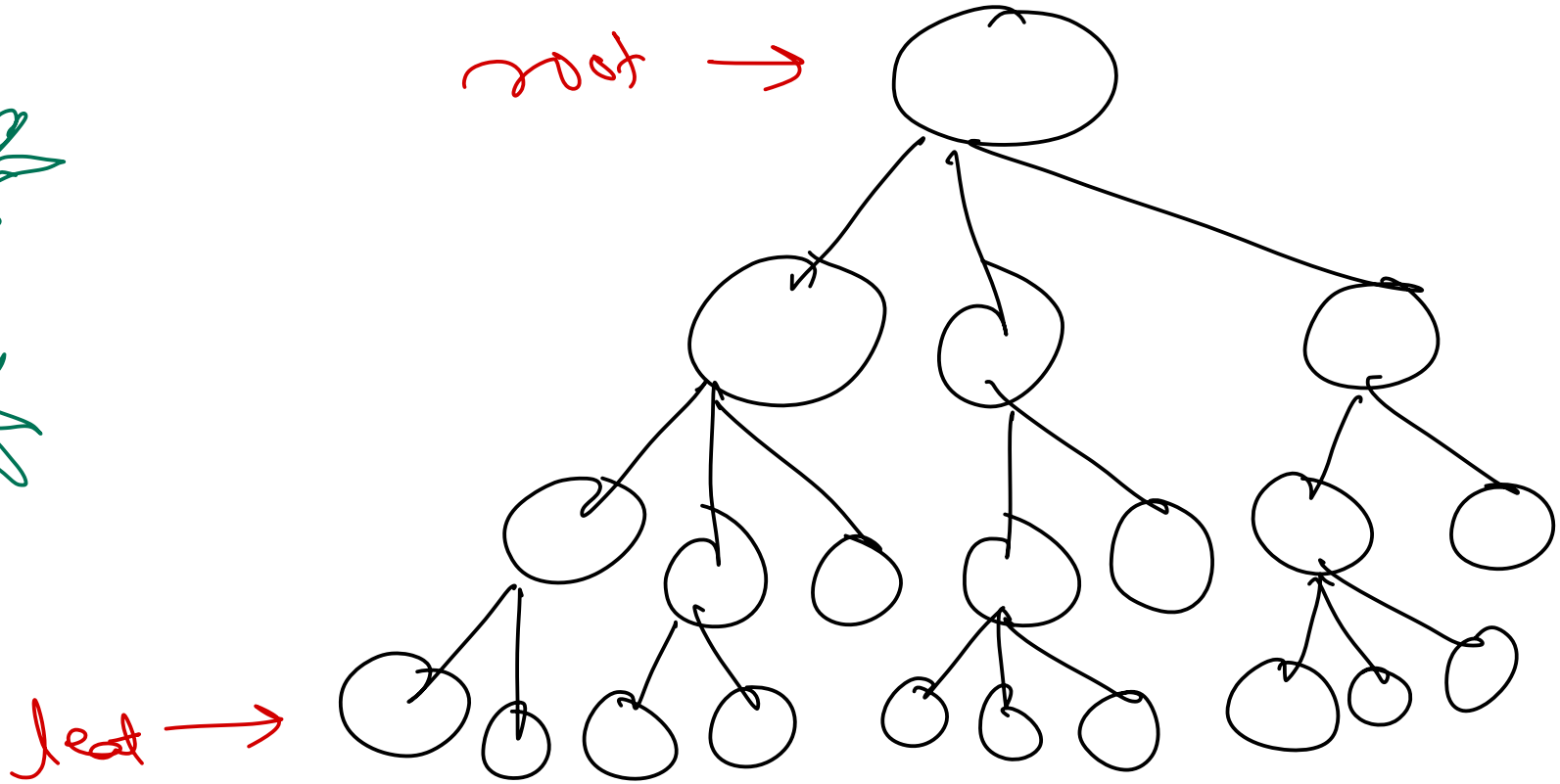
Linked List





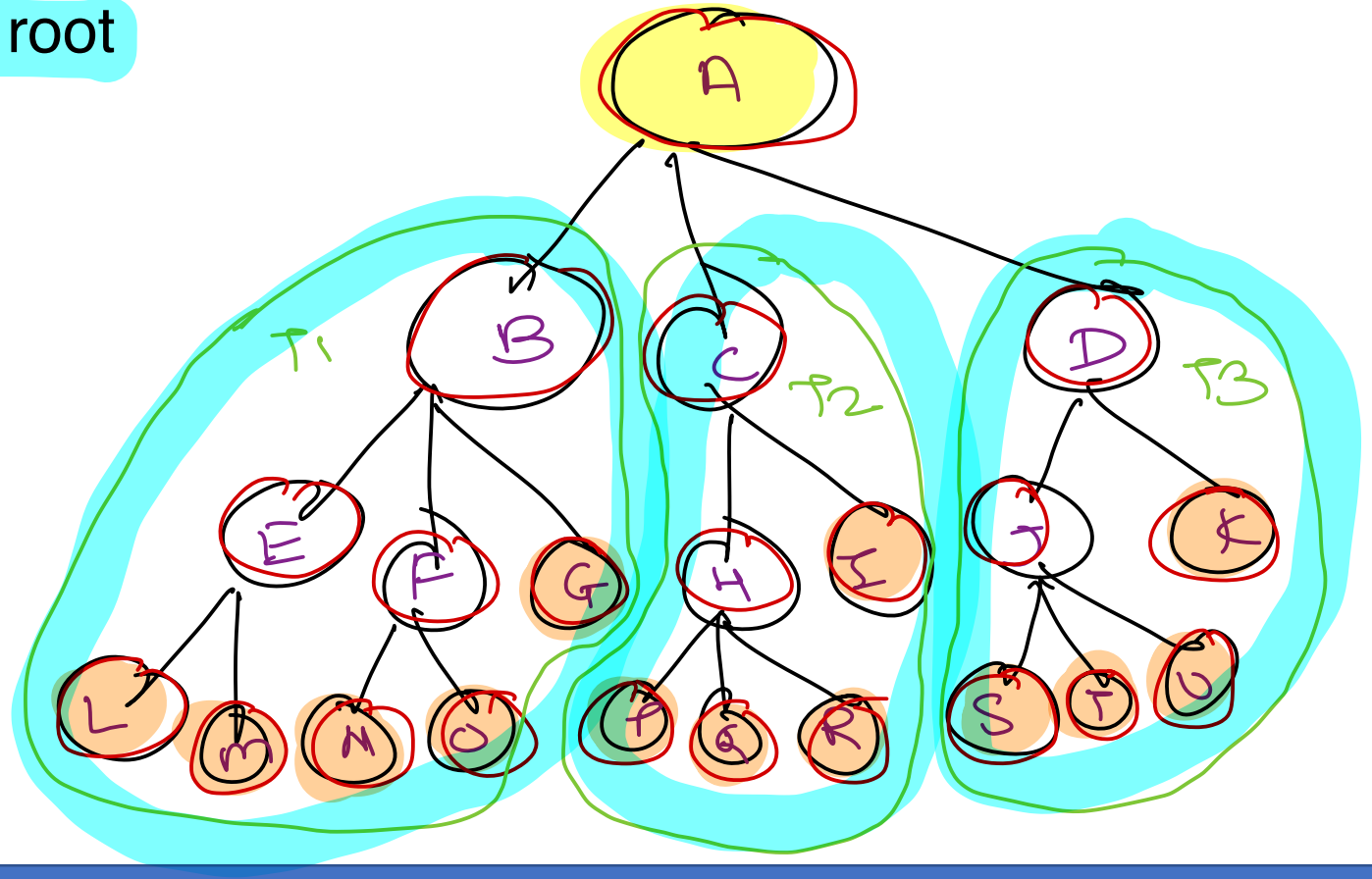
Linked List





Tree Definition

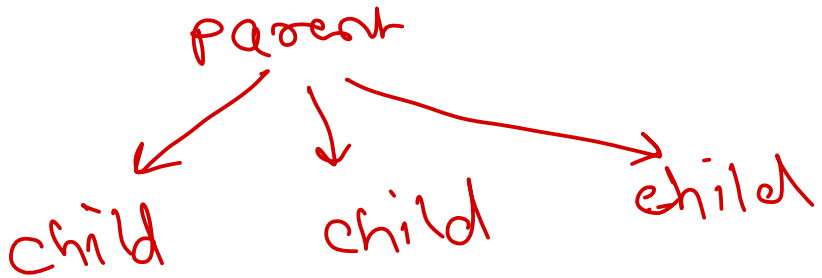
- **Tree** is a finite set of nodes with one specially designated node called the "**root**" and the remaining node are partitioned into disjoints sets T1 to Tn, where each of those sets is a TREE.
- T1 to Tn are called **sub-trees** of the root



Tree terminologies

pointers

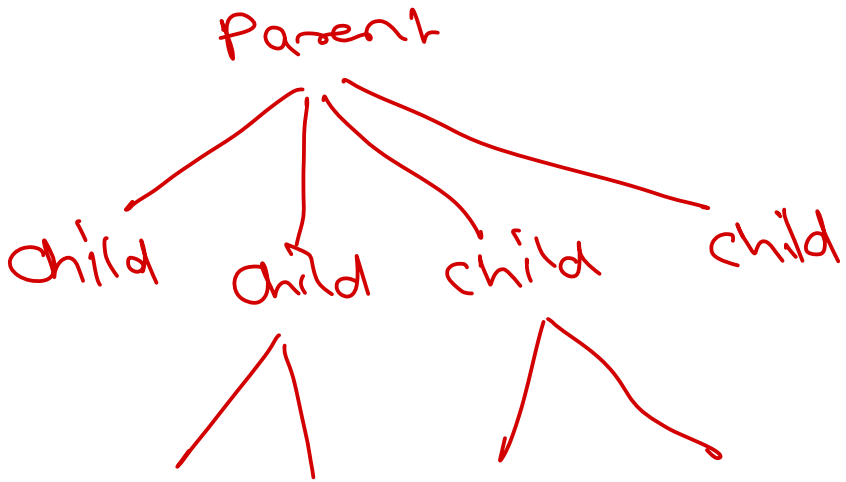
- **Node**: A item storing information and branches to other nodes
- **Null Tree**: Tree with no node (empty tree).
- **Leaf Node:** Terminal node of a tree & does not have any node connected to it
- **Degree of a Node:** No of sub trees of a node (count of child nodes)
- **Degree of a tree:** Degree of a tree is maximum degree of a node in the tree



Tree terminologies

child

- **Parent Node:** node having other nodes connected to it
- **Siblings:** Children of the same parents
- **Descendants:** all those node which are reachable from that node
- **Ancestor:** all the node along the path from the root to that node

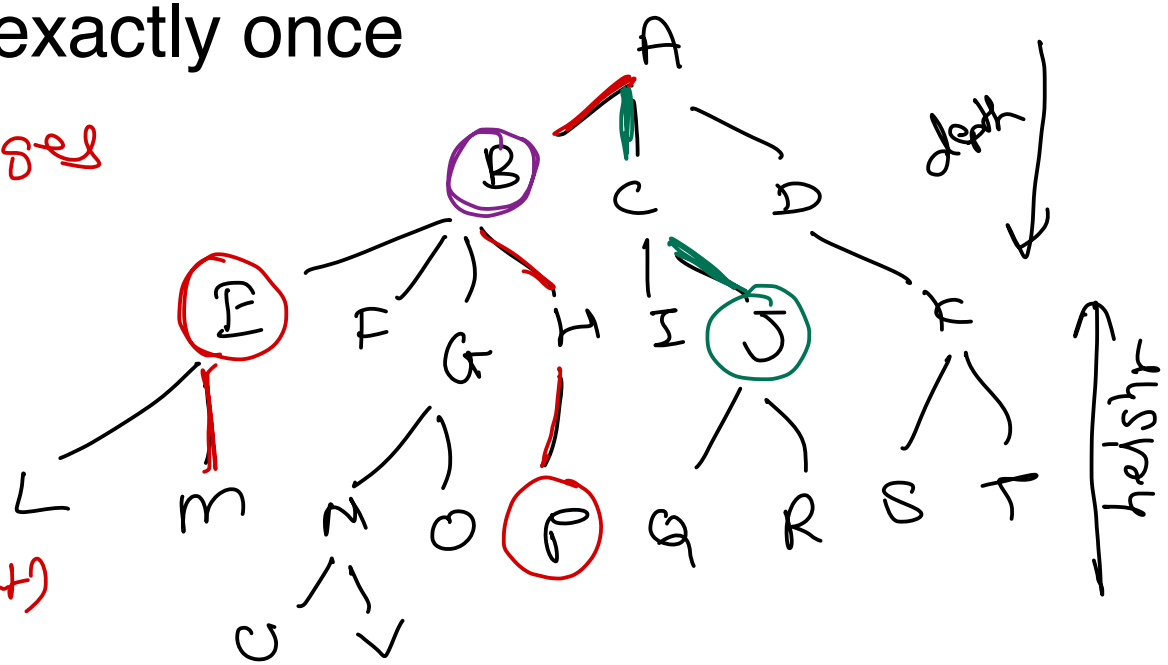


Terms used in Trees

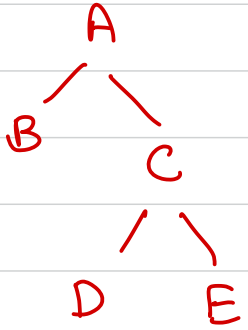
- **Level of a Node:**
  - Indicates the position of the node in the hierarchy
  - Level of any node is level of its parent +1
  - Level of root is 1
- **Depth/Height of a tree:** maximum level of any node in the tree.
- **Traversal** : Visiting each node of tree exactly once

Depth of a node = num of edges from root to that node.
= level of node − 1

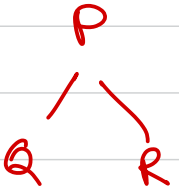
Height of a node = num of edges from bottom most leaf (descendent)



height of tree



h = 3



h = 2

Z

h = 1

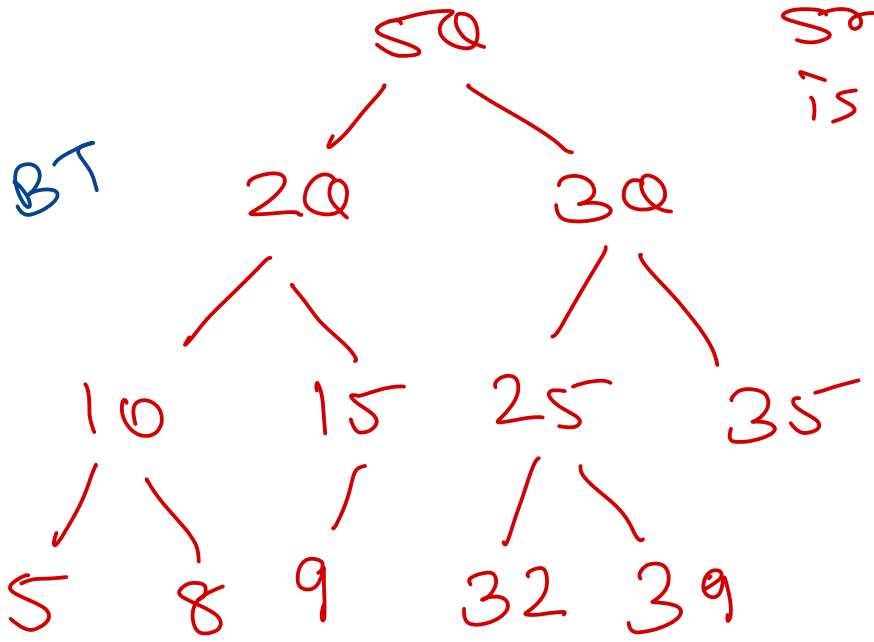
h = 0

height of null tree = 0

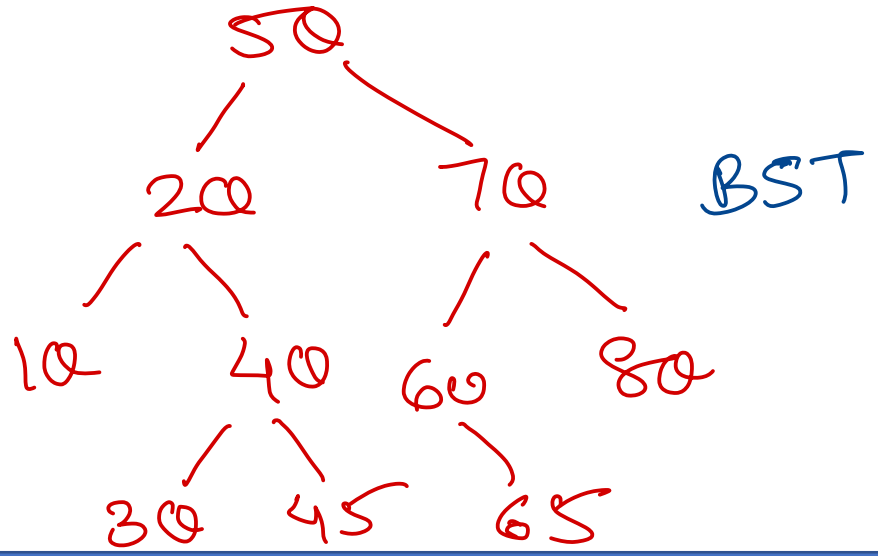
Types of Trees

→ 2-tree (degree = 2)

- **Binary Trees:** It is a finite set of nodes partitioned into three sub sets:- Root, Left sub tree, Right sub tree (each node have max 2 child nodes).
- **Binary Search tree:** A binary search tree is a binary tree in which the nodes are arranged according to their values. each left child node is smaller than node & right child node is greater/equal to the node.



BT



BST

Binary Tree Traversal

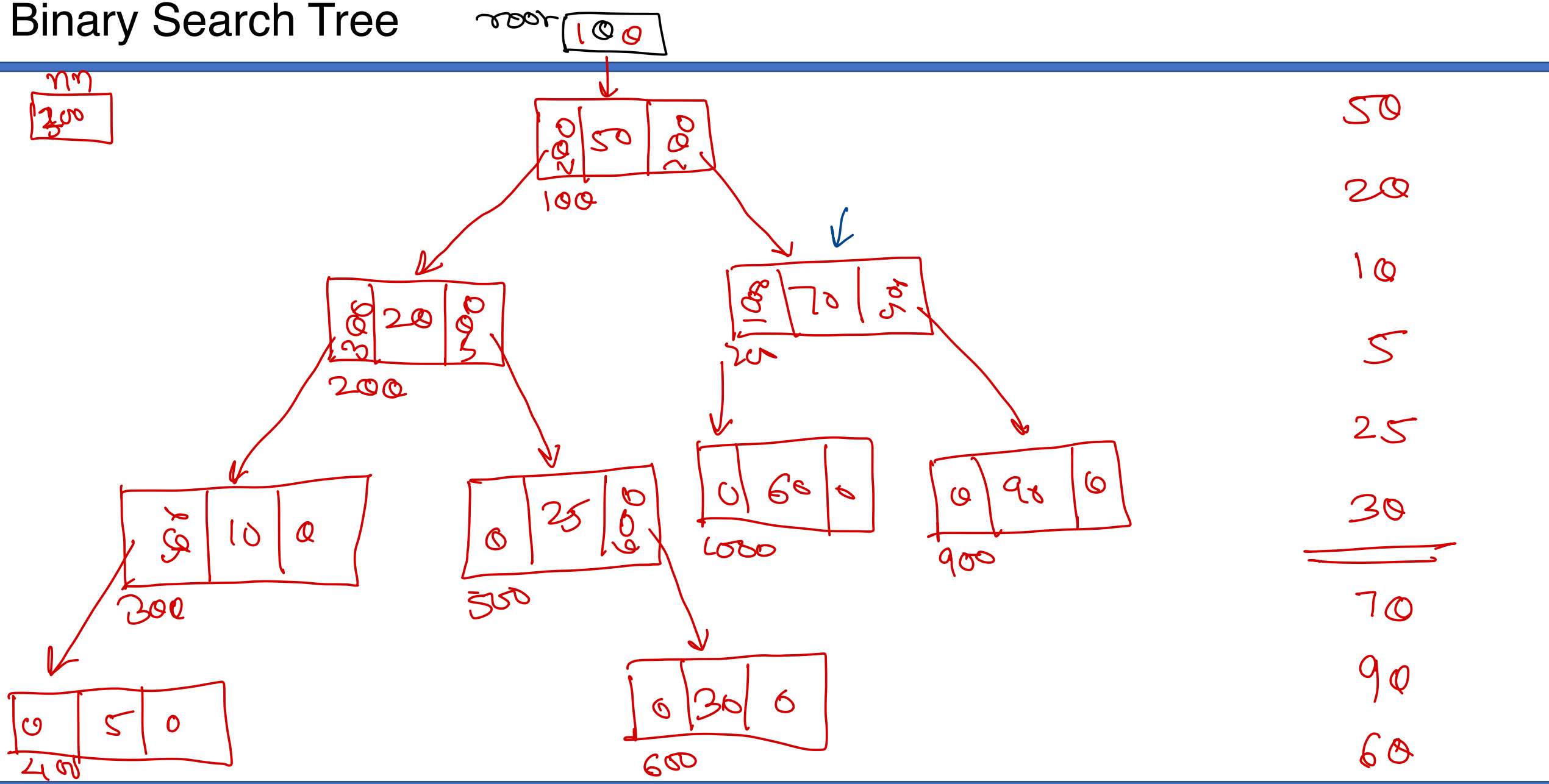
- **In-order** → L V R
- **Pre-Order** → V L R
- **Post-Order** → L R V
- The traversal algorithms can be implemented easily using recursion.
- Non-recursive algorithms for implementing traversal needs stack to store node pointers.

Binary Search Tree – creation

```
class node {
private:
        int data;
        node *left;
        node *right;
public:
        node();
        node(int val);
        friend class bstree;
};
```

```
class bstree {
private:
        node *root;
public:
        bstree() {
                root = NULL;
        }
        void add(int val);
        void inorder();
        void preorder();
        void postorder();
};
```

Binary Search Tree



Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>