ARRAY AND STRINGS.

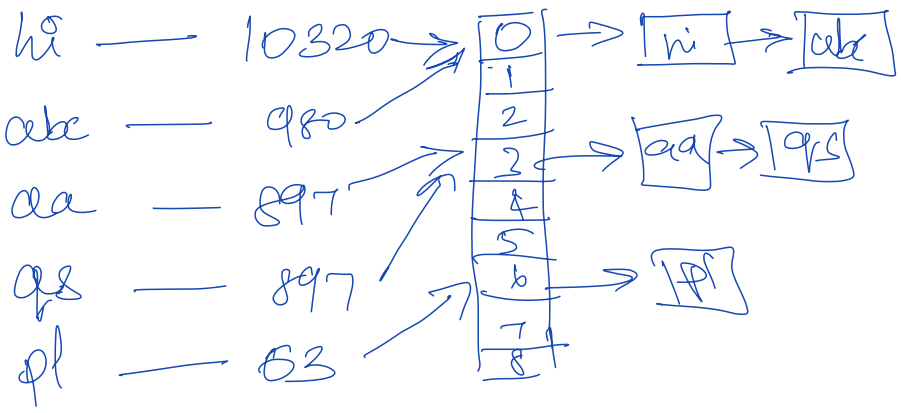
Hash Tables :-

1. Maps keys to values.
2. Highly efficient lookups
3. Infinite number of keys & finite ints.
    1. Compute key's hash code
    2. Map hashcode to index
    3. Every index has linked list of key and values

(x) two keys can have same same hash code

(x) two hash codes can be linked to same index.

eg :-



ALT :- we can implement hash tables with Balanced binary search tree.

Linked List

1. No constant time access to n^{th} element
2. Sequence of nodes
3. Each node points to next and previous node
4. Add or Remove from beginning.

eg.

ALT: Number of Linked List Problems rely on Recursion.

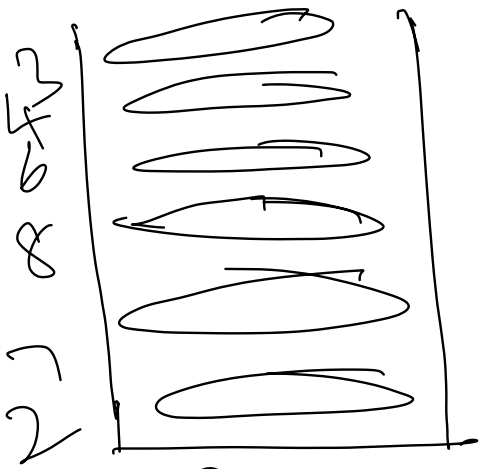
If a LinkedList cannot solve a problem. Recursion can be used.

eg: problems:

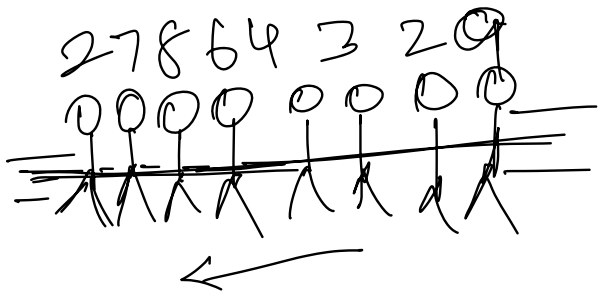
1. Remove duplicates from unsorted array.
2. To find k^{th} to last element of a singly link list
3. Delete a node in middle of a singly LL.
4. To check if a LL is palindrome
5. Circular LL, beginning of the loop.
6. Two 3-digit numbers stored in reverse of a Linked List add them and Return LL.

Stacks & Queue

Stacks - LIFO	Queues - FIFO
pop ()	add (i)
push (i)	Remove ()
peek ()	peek ()
isEmpty ()	isEmpty ()



Stacks



Queue

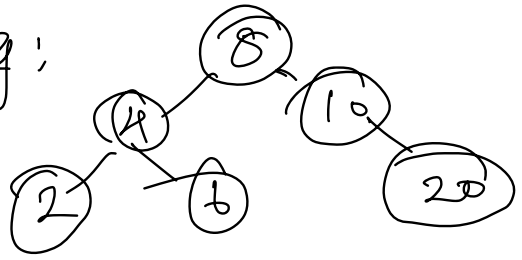
Trees & Graphs

Trees are node & child structures
Root node can have zero or more child

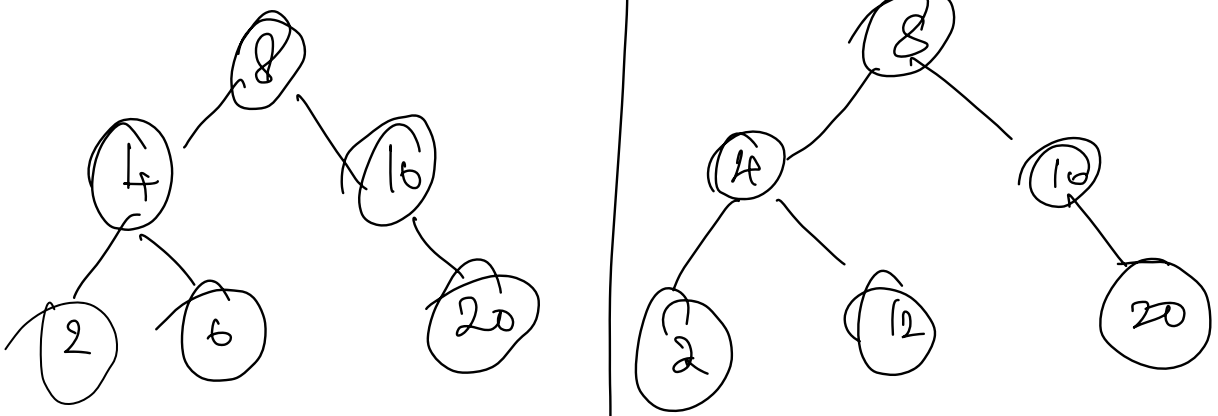
Binary Trees:

Can have only two child
node without child is leaf

eg:



Binary Tree vs Binary Search Tree



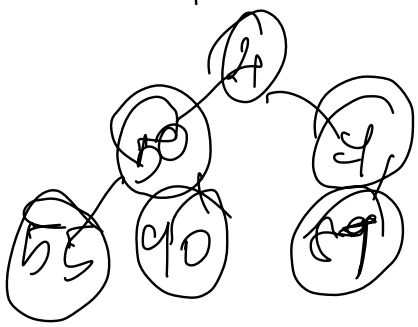
for each node
BST has all left descendant <= n < all right

ALT: Complete Tree vs non complete tree

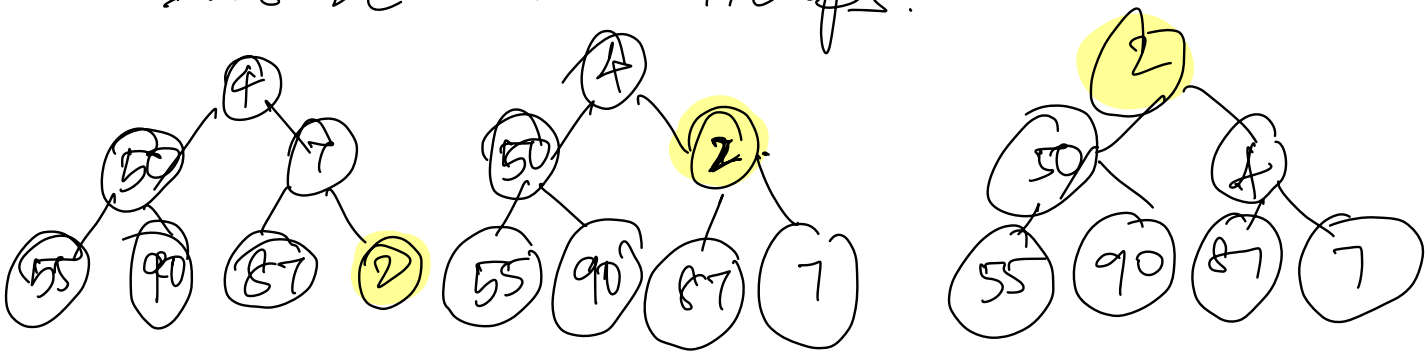
Heaps are complete BTrees.
two types Min and Max

Min Heaps

Each node smaller than its child.
Root is minimum element
Totally filled other than rightmost leaf.



Insert in Heaps:



Extract Min