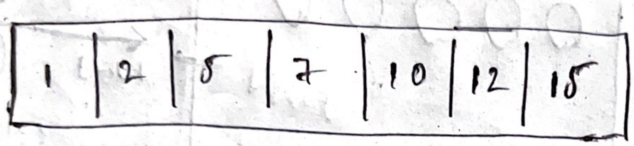
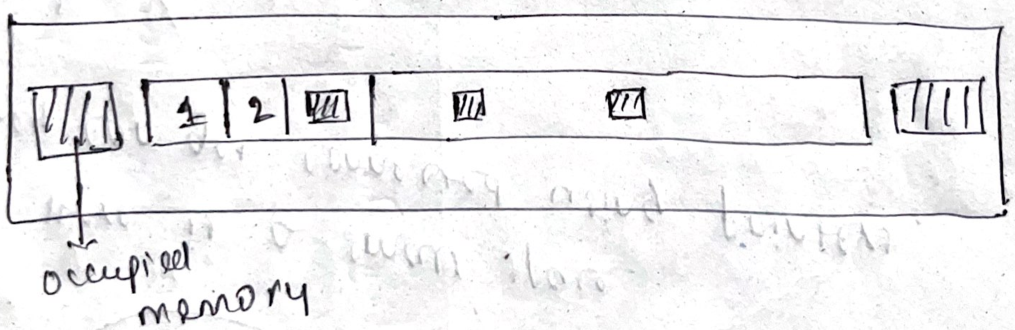
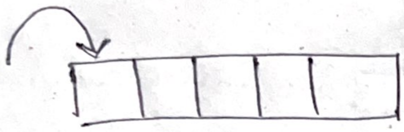


Memory of array

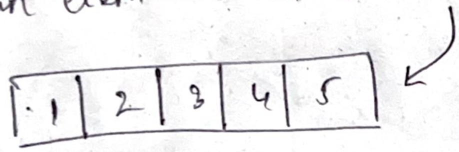


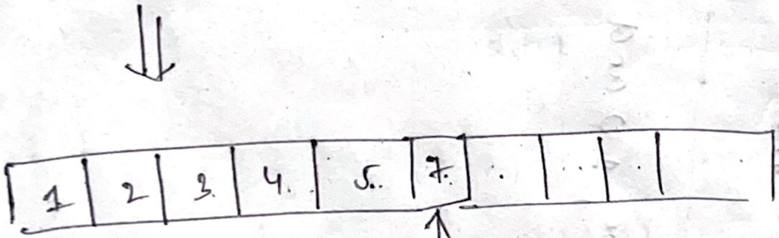
Vector A



size = 5

To add an element at 6^{th}. Adding element here.





Element added

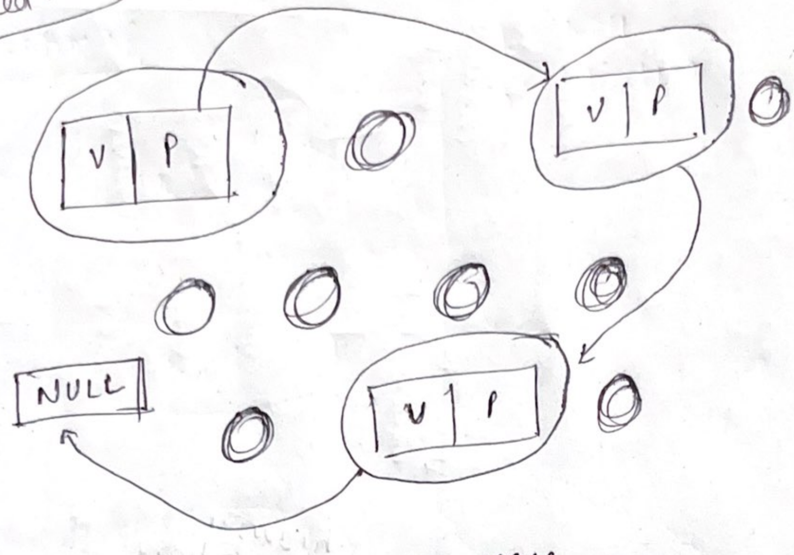
capacity doubled .. size increased by 1.

size = 6

capacity = 10

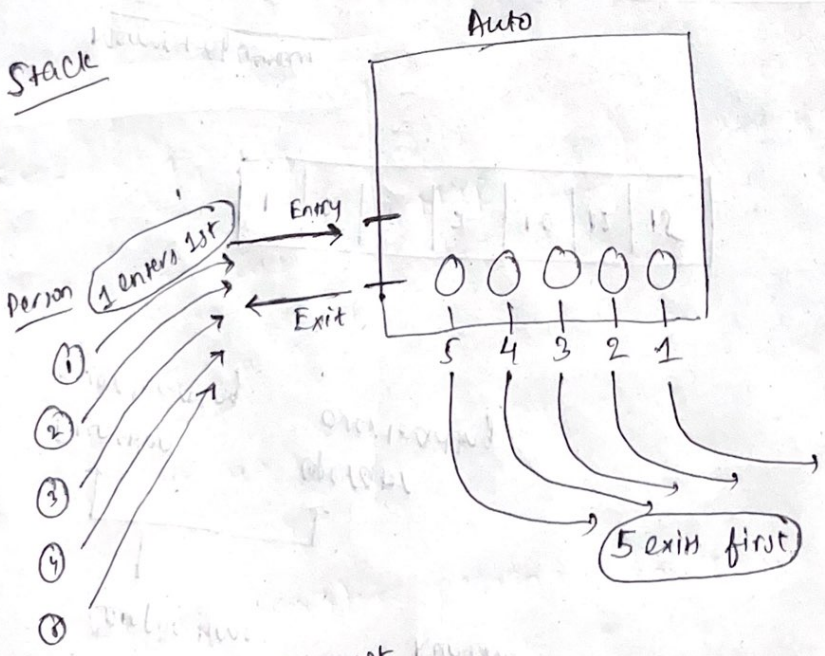
Linked list :

V = value
P = pointer



Even if there is a small space
they can get connected using pointers.

Stack



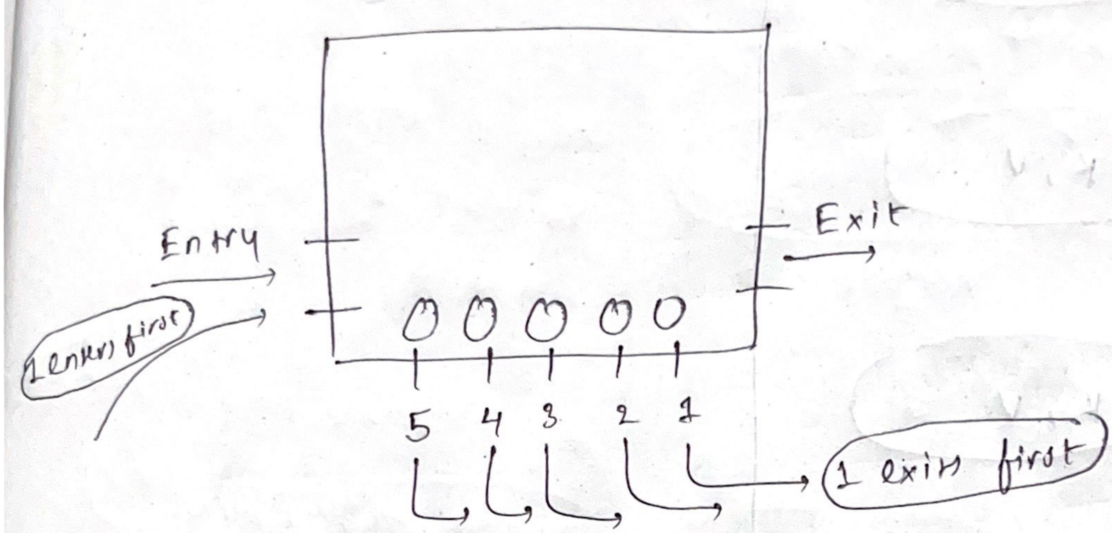
Concept

1st in Last Out (LIFO)

(4)

Queue

Same as stack but point of entry and point of exit are different



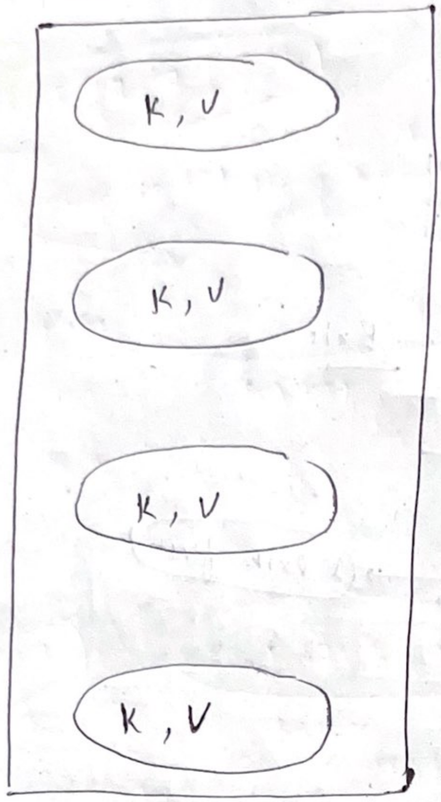
Concept.

1st in 1st out (FIFO)

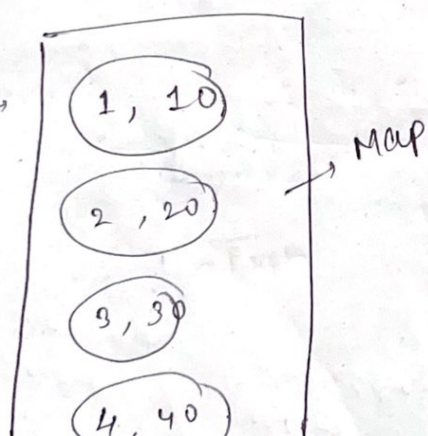
Hashmap.

K = key
V = Value

Pairs



Ex →



Find if key 4 is present in map. and tell value

map [4] = ?

How it is stored in memory ?

Normal Hashmap

4 , 40
3 , 30
2 , 20
1 , 10

stored in
an order

TC = $n \log n$ → best & worst

sorted in order of keys

unordered Hashmap.

2 , 20
4 , 40
3 , 30
1 , 10

we cannot say in which
index some key is present
as it is not sorted

TC = n^2 → worst
TC = n → Best.

Set
→ Basket

2, 2, 5, 7, 7, 6, 4

store in set

Basket

Step 1 — 2, 5, 7, 6, 4 → Unique values only stored in basket.

Basket

Step 2 — 2, 4, 5, 6, 7 → Elements get sorted in a specific order.

TC : nlogn

~~Multiset~~

~~2, 2, 5, 7, 7, 6, 4~~ → 2, 2, 4, 5, 6, 7, 7

Multiset

2, 2, 5, 7, 7, 6, 4

store in multiset → Basket

Step1 2, 2, 5, 7, 7, 6, 4 → Unique values not stored. Instead Multiple values stored

Step2 2, 2, 4, 6, 7, 7 → Elements gets sorted in increasing order.

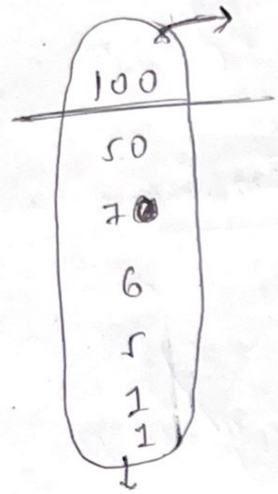
TC : nlogn.

Heap → basket

100, 50, 70, 6, 5, 1

store in heap →

Heap basket



Heap basket

step1 Unique values not stored Instead multiple values stored

step2 All elements are arranged according to priority

step3 To remove elements from heap the topmost element removed first

We cannot access other elements when topmost element is removed at once

step4 To acess 5 → remove 100 first
→ remove 50 first
→ remove 70 first
→ remove 6 first
→ remove 5 first → we get 5 by following above step 1 by 1

~~remove~~

This is MAX HEAP

Min Heap → basket

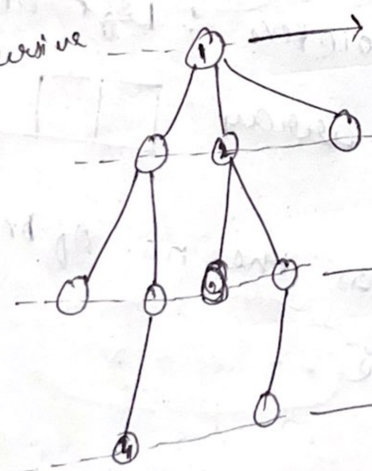
1
1
5
6
7
50
100

To accen 7
→ remove 1
→ remove 1
→ remove 5
→ remove 6
→ remove 7 → we have to follow it step by step 1 at a time

$\boxed{\text{Heaps are used in} \mid \text{DP.} \mid \text{Trees}}$

Example of use of heap. in Trees.

Ex → we have recursive tree



→ u want max element at this point push (1)

→ Push (2) to max heap

→ Push (3) to max heap

→ Push (4) to max heap

Max Heap

[(1) | (2) , (3) , (4)]

TC : nlogn

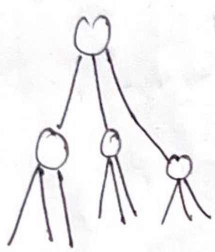
We can reduce it to n by customization.

Trees and Types of Trees

Binary tree

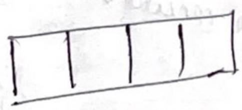
- Two pointers : Left and Right
- So we use 2 struct
  Each struct represents a node

n array tree



n = 3

Instead of 2 nodes at each point we have n nodes.

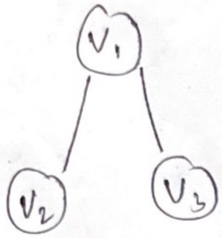
- nodes are stored in array
- We don't use Two pointers : Left and Right
- Instead we use an array 
- In this array pass the no. of branches reqd at each node

Here n = 3

Binary search tree

Relation :

$V_2 < V_1 < V_3$



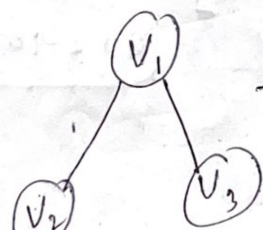
But for case :

1	100	100

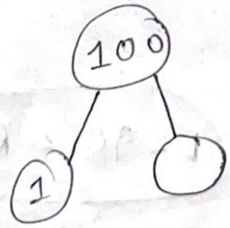
We need to place it in a BST.

Problem : V_1 and V_3 are equal.

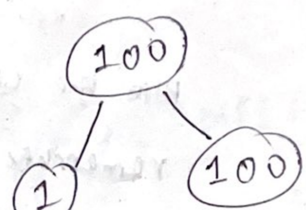
But we can place V_3 in this format :-



→ Reqd → $V_2 < V_1 < V_3$



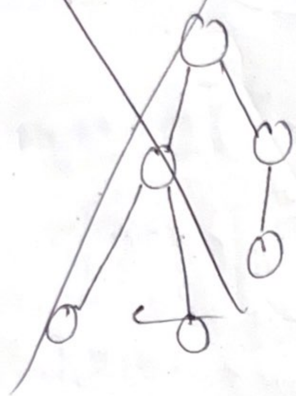
→ $V_2 < V_1$ → satisfied



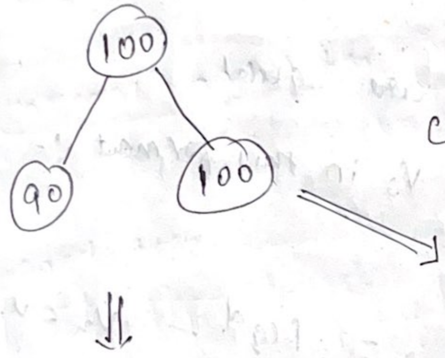
→ However even if $V_1 < V_3$ not satisfied

But always place the value to the right

~~Another example of B.S.T.~~



Another example of BST

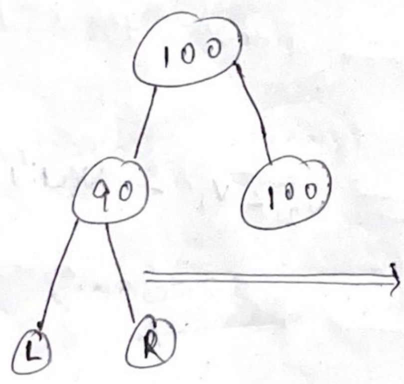


~~use to~~

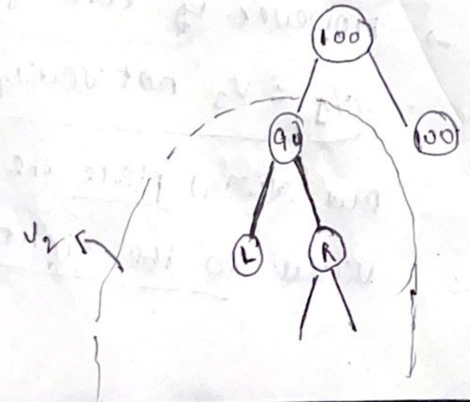
Condition :

$V_2 < V_1 < V_3$

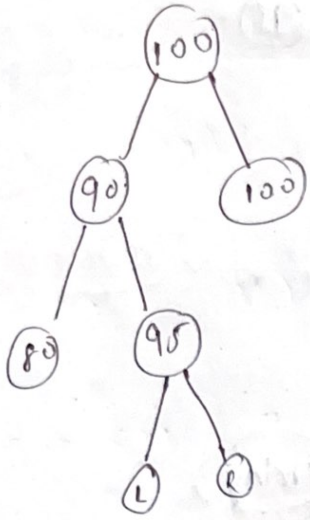
$90 < 100 < 100$



$L < 90 < R$

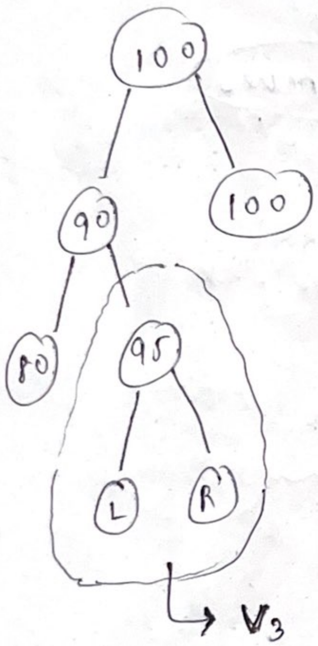


Also ~~the~~ we must remember that left part of tree ~~must~~ (V_1) must be less than 100.



95 → greater than 90 and

→ less than 100



Normal condition :

$L < 95 < R$

New condition :

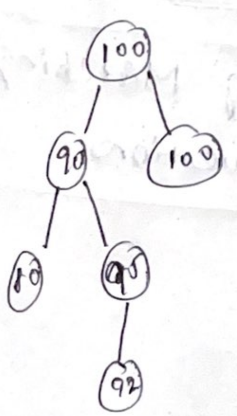
L is to right of 90.

so , L value must

be greater than 90

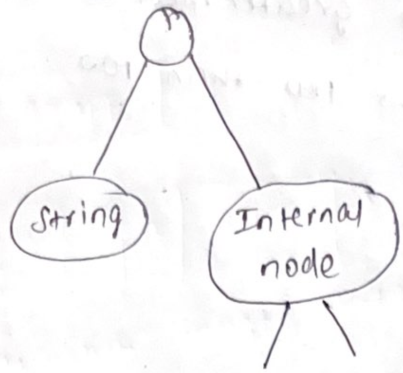
=) $90 < L < 95$.

=) $L = 92$ (Let)



=) COMPLETE BST

Cord Tree or Rope DS

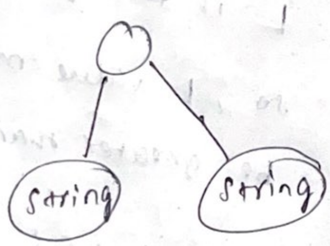


↳ 1 branch of tree will have value string

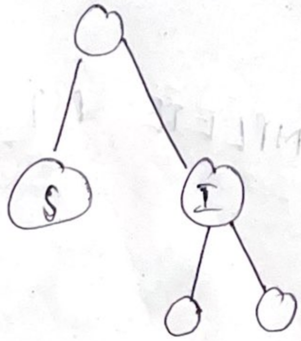
↳ other branch is of Internal node

Only internal node has sub branches

To end cord tree



To do branching



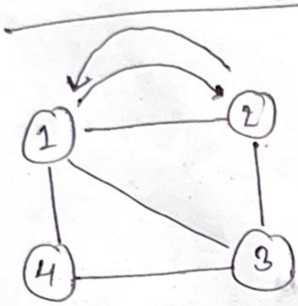
Used in

String Matching questions

Graphs

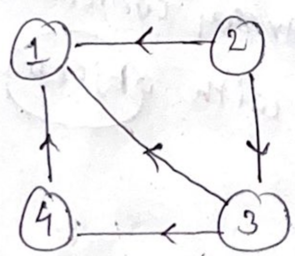
↳ Trees are derived from Graphs.

Non directed Graph



Cycles always exist

Directed Graph



Cycles may exist

Cycles may not exist