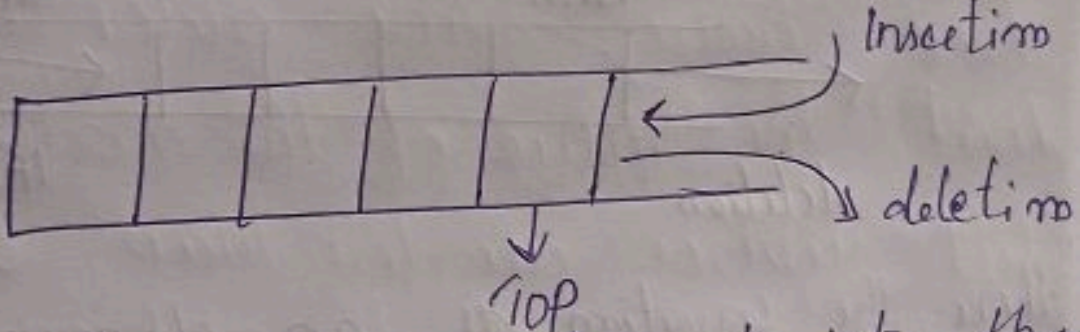
1. a) Stack

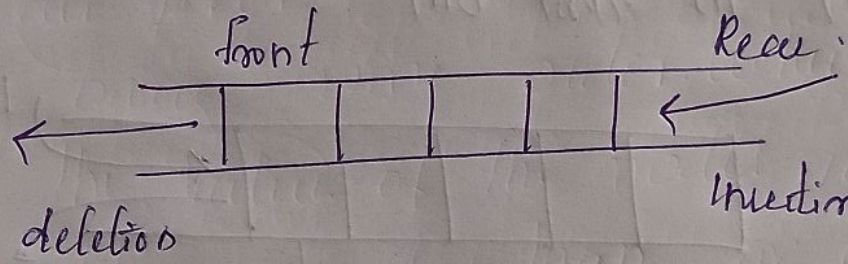
stack is a linear type data structure and also it is an ordered list elements where all insertions and deletions are made at the same end. and that same end is called 'TOP' of the stack. Stack is performed under LIFO principle last in first out.



The insertion of an element into the stack is called push operation, and deletion of an element from the stack is called pop operation. In stack, we have only one point to access the list it called TOP

Queue

A queue is also a linear data structure with the operations like insertion and deletion can take place at both ends. queue is follow the FIFO principle. first in first out. the element which is inserted at first in the list, is the first element to be removed from the list



Here the insertion of an element is called enqueue operation and the deletion of an element is called dequeue operation. The ~~insertion~~ deletion operation is at the front end and which can be denoted as FRONT. And the insertion is at Top of the queue is called REAR. In queue there are two pointers to access the list called front and rear.

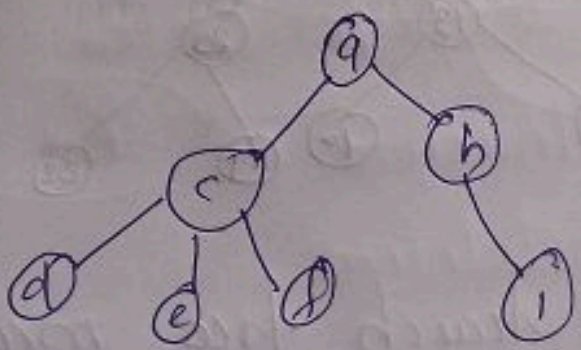
queue is used to solving problems having

sequential processing -

16 Compare Conliat binary tree and BST

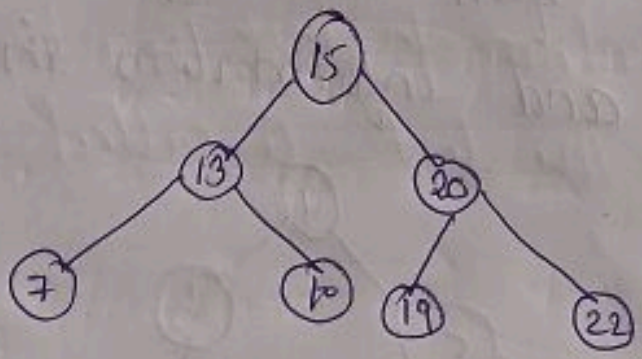
Conliat binary tree

Binary tree is a non linear datastructure which represents hierarchical relationships b/w data in a tree structure.
Each node must have at most two child nodes with each node being connected using edges. In Conliat-binary tree, there is no relative order to how the nodes be organized basically ~~dat~~ binary tree are used for fast and efficient access of data and information in a tree.



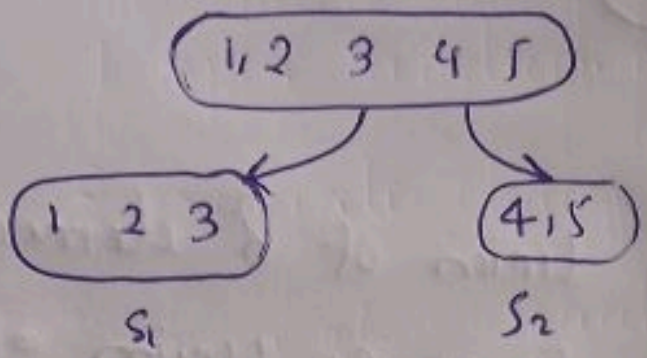
Binary search tree

Binary search tree is a type of tree data structure with some hierarchical relationship b/w elements. and all elements that are present in the binary search tree have some order. which means that the value of the nodes in the left subtree are less than or equal to the value of the root node, and the nodes to the right subtree have values greater than the value of root node.



The binary search tree mainly used for insertion, deletion and searching of

1. d) Two or more sets with nothing in common are called disjoint sets

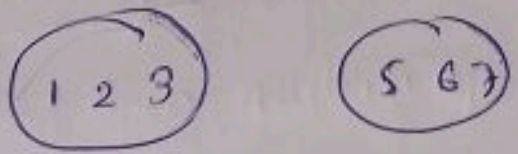


$S_1 \cap S_2 = \emptyset$ (disjoint set

usage of disjoint set,

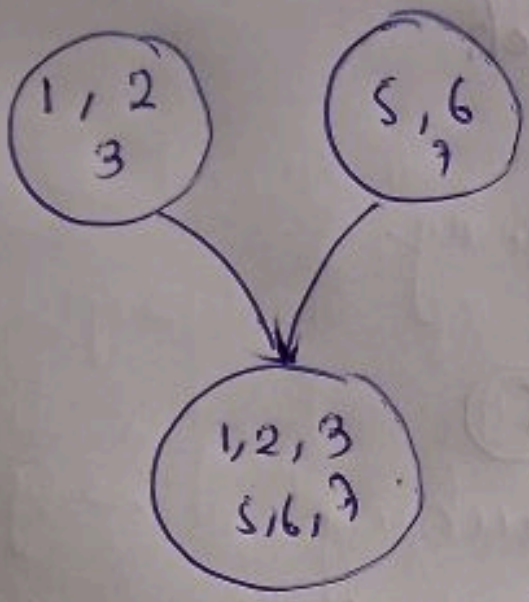
keeps track of the set that an element belongs to: It is easier to check given two elements, whether they belong to the same subset (AND operation)

$S_1 \pm = (S_2)$



at return time

② used to Merge 2 sets into one union operation

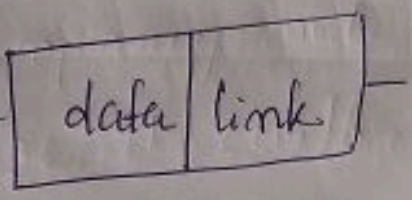


Union of 2 elements is
same as union of 2 [illegible]

2) Singly linked list

Singly linked is a linear data structure. And it is a sequence of data and the linked list is a sequence of data linked with each other. We can simply say, Singly linked list is a sequence of elements which every element has link to its next element in the sequence

In a singly linked list, each individual element is called node. The node have two parts data and link. The data stores the actual data to be stored. And the link which stores the address of the next node.



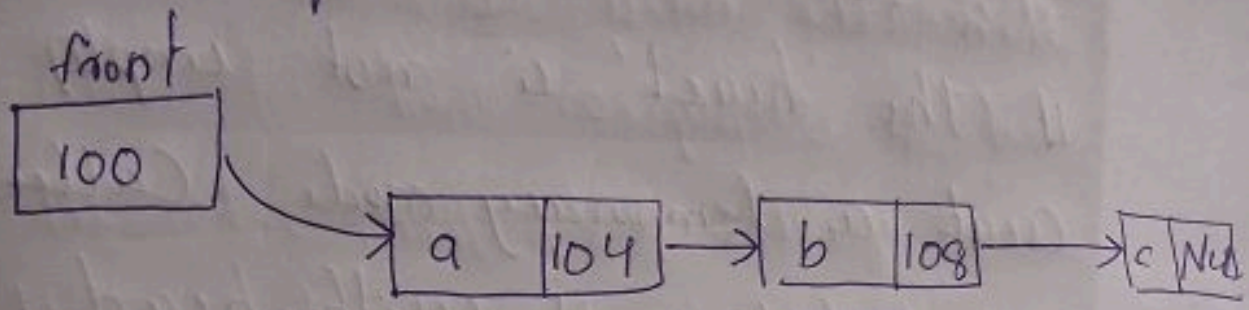
In singly linked list address of the first node is always stored in reference node known as front or head. And also the last element in the singly linked list have the link value NULL

~~for example there are 3 operations~~
~~are there to perform on~~

* Insertion
* deletion
* Display

that 3 operation are performed on Singly linked list.

for example, look at the below mentioned picture.

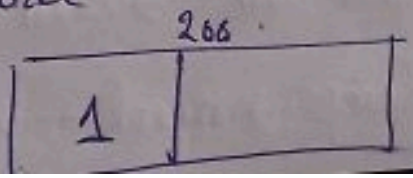


first, we are discussing about the Inserting operation. We can insert elements in three ways.

1. Inserting at beginning of the list
2. Inserting " end of the list
3. Inserting at a specific location

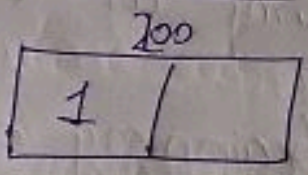
Inserting at beginning of the list and end

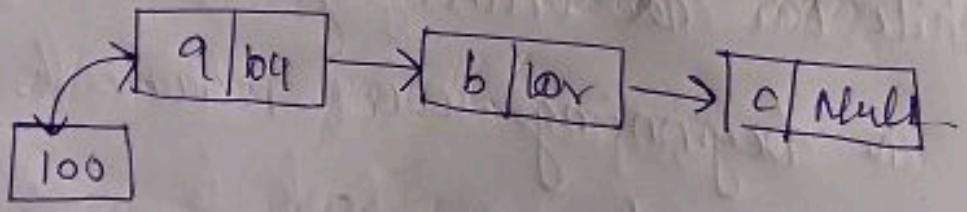
while we are inserting an element at the beginning we have to create a new node



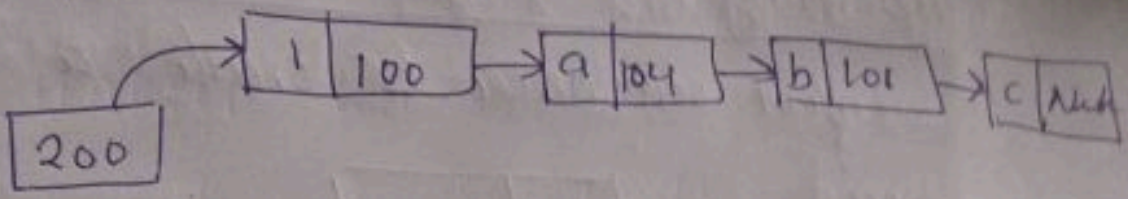
Then we have to check the reference node is empty or not: if the reference node(head) is empty then set, the address of the new node.
If the 'head' is not empty then creat a temporary node called 'temp' and initialize it with head.
keep moving the temp to its next node until it reaches the last node in the list (temp → next = Null)
finally set temp → next = Newnode

pictorial representation

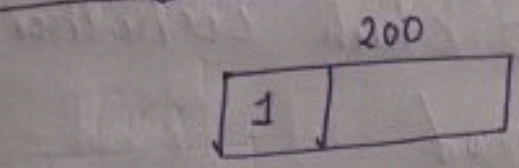


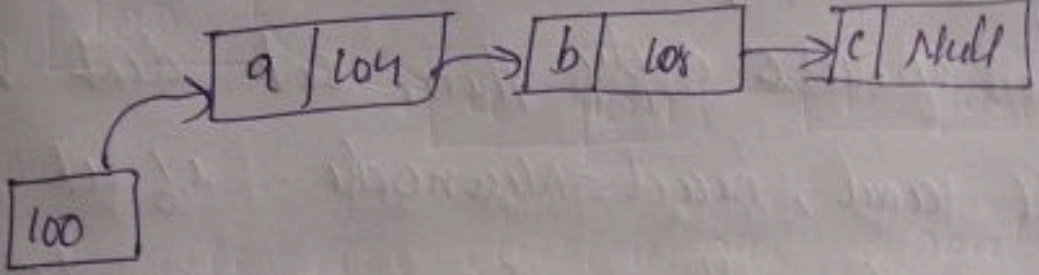


New list :

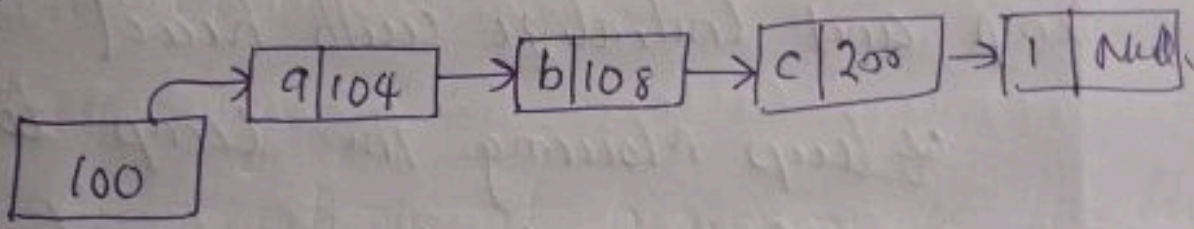


Inserting at end



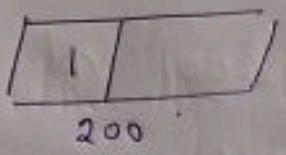


after ~~insertion~~ insertion



Inserting at Specific Locations

first we have to create a new node



Then we have to check whether the list is empty or not.

If it is empty then, set data value null and, head = Newnode. If it is not empty, then define a pointer temp and initialize with head

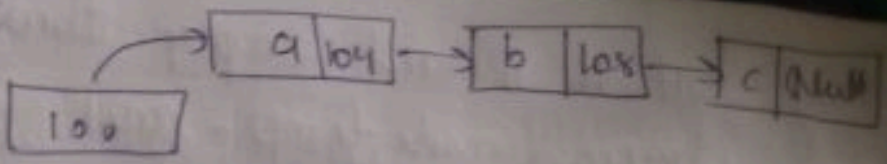
If temp Moving the temp to the next node until it reaches to the node after which we want to insert the newnode.

we have to check whether temp is reached last node or not. If it is reached the last node then display 'Insertion not possible'.

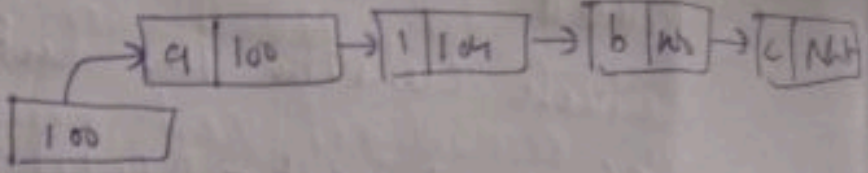
Then Set Newnode → next = temp → next

temp → next → newnode

before insertion:



after insertion:



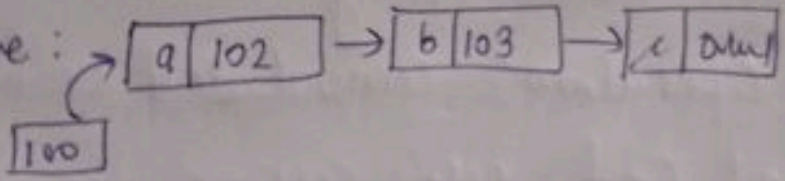
deletion

like insertion, deletion can be performed in 3 position.

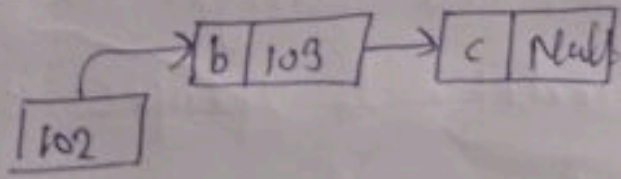
1. Deleting from beginning
2. " " end of the list
3. Deleting specific node.

1. Deleting from beginning

~~after~~ before:



after:

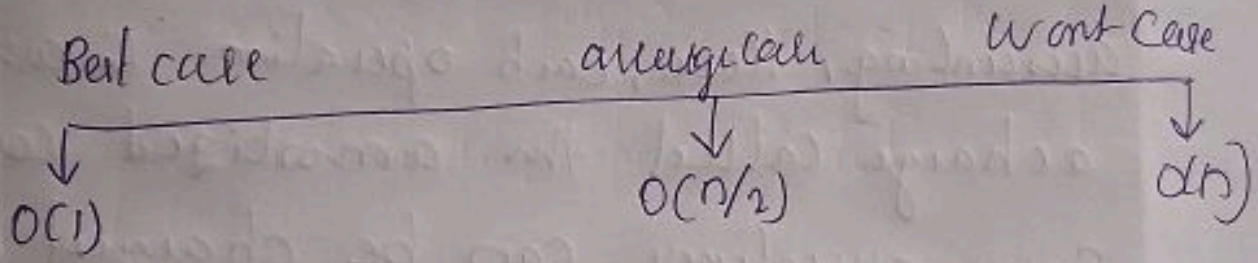


if the list empty then we can't delete the item/node. So we have to chack it first; if it is empty then print "Not possible"

If it is not empty then define a temp
ray node temp. and initialize it with
head.
check with the last having one node
(temp → next = NULL).
if it is true, then set head = NULL
and delete temp.
if it is false then set head = temp->
next, and delete temp

③ Discuss about amortized analysis and any two methods in detail.

Answer:) Amortized Analysis is used for algorithms and it is a method of analyzing the code associated with a data structure that averages the worst operation out over time. Amortized analysis guarantees the average performance of each operation in worst case.



There are three types of Methods are there in amortized cost

1. Aggregative
2. Accounting
3. potential

1. Aggregative analysis

In aggregative analysis, we show that

for all n, a sequence of n operation takes a worst case time $T(n)$.

In worst case the average or amortized cost per operation

$$\frac{T(n)}{n}$$

2) Accounting Method

It borrows ideas and terms from accounting, where, each operation is assigned a charge called the amortized cost. Some operations can be charged more or less than they actually cost. When an operation's amortized cost exceeds the actual cost, we assign the difference, called credit, to specific objects in the data structure.

④ There are two types of Collision resolution techniques are there

1. Open addressing
2. Separate chaining

In the process of searching, the given key is compared with many keys and each key comparision in known as probe. The efficiency of a Collision resolution technique is defined in term of the number of probe required to find a record with a given key

Open addressing

In open addressing, the key which caused the Collision placed in the hash table itself but a location other than its hash address.

There are three methods

1. linear probing
2. quadratic probing
3. Double hashing
