Lect - 16

## Insertion in Singly Linked List :-

Before Starting the Insertion operation. Let's discuss some new terms :-

- Memory Allocation ; Garbage Collection

⇒ Mechanism which provides unused memory space for the new nodes.

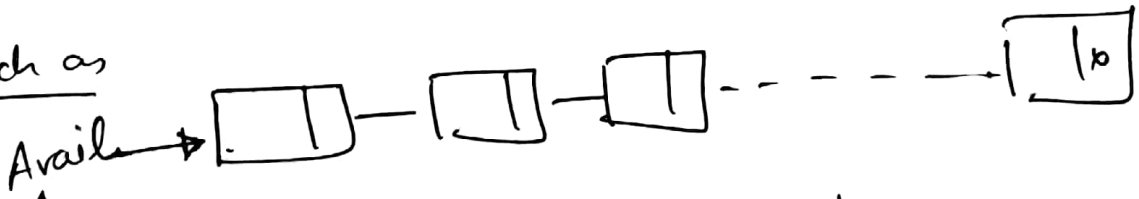
⇒ Memory space of deleted nodes becomes available for future use.

⇒ Together with the linked list in memory, a special list is maintained which consist of unused memory cells.

⇒ This list is called list of available space or free - storage list or free pool.

⇒ It has its own pointer.

such as




↑ pointer point at first node of free pool.

## Garbage Collection :-

⇒ The operating system of a computer may periodically collect all the deleted space onto the free - storage list. Any technique which does this collection is called garbage collection.

⇒ Garbage collection may take when there is only some minimum ~~amount~~ amount of space or no

space at all left in the free-pool.

or

⇒ when CPU is idle & has time to do the collection

⇒ This process of garbage collection is invisible to the programmer.

## Overflow & Underflow :-

Overflow :- New data to be inserted into a Data structure but there is no available-space

ie free-pool is empty.

If AVAIL = NULL & there is an insertion.

so overflow will occur.

Underflow :- When one wants to delete data from data structure that is empty.

ie when start = NULL & there is a deletion.

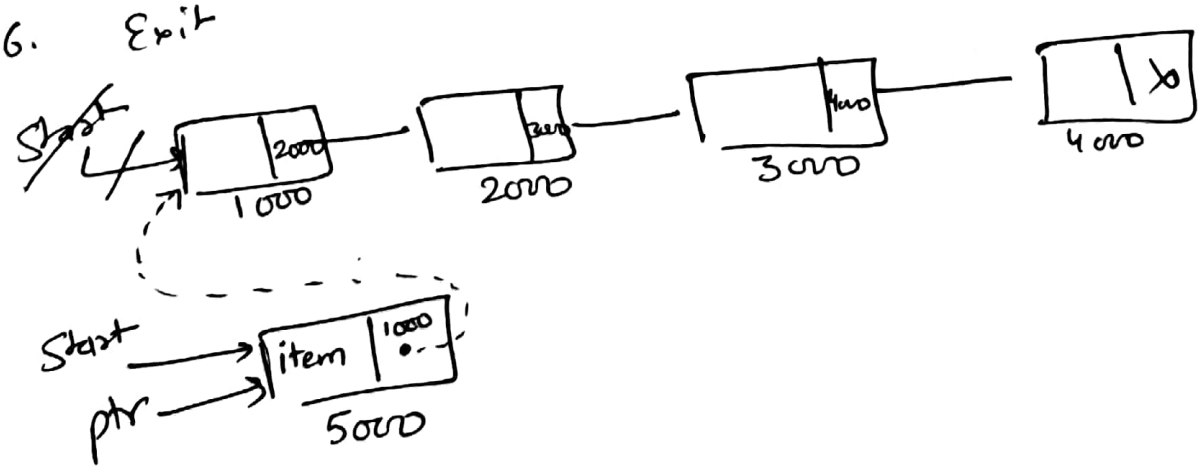
so underflow will occur.

## Insertion in a Linked list :-

### At Beginning

This algorithm will insert item as the first node in the given list.

1. If AVAIL = NULL
   then: Write overflow & Exit. || No free space is available to create a new node
2. Set Ptr = AVAIL
   and AVAIL = LINK[AVAIL] || Else we will remove the first node from avail list.
3. Set INFO[Ptr] = item || copy the new data into the new node ptr.
   || In program this line can be written as
   ptr → info = item.
   || But algorithm → is not represented. We show it with help of [ ] square bracket.
4. Set LINK[Ptr] = Start || new node now points to original first node of list.
5. Set Start = ptr || Changes Start so it points to the new node.
6. Exit



Now we will take all case i.e insert at Beg, at End or after a given node.

- We are considering that our list is sorted like list
- We perform searching to find the location where we want to do the insertion.

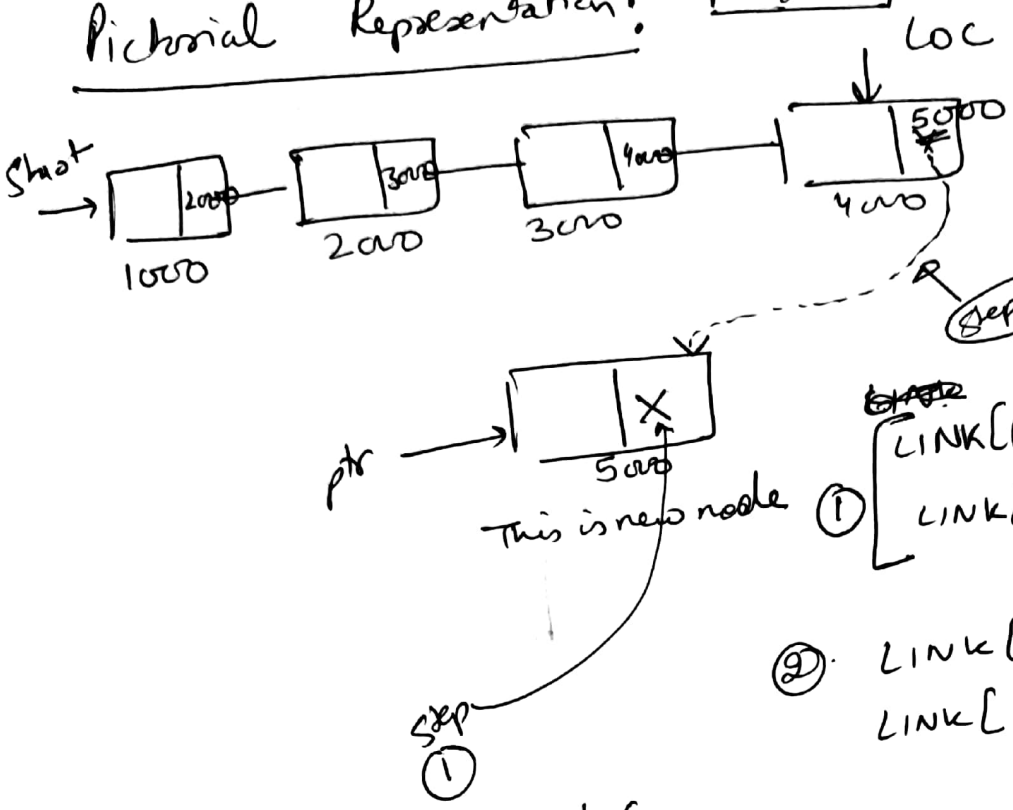
FIND is the name of algorithm.

```
FIND ( )
{
1. If start = NULL                          || when list is empty
   then Set LOC = NULL & return                so we return to calling
                                               function as LOC=NULL

2. If item < INFO[start]                    || special case i.e
   then set LOC = NULL & return                the item we want to
                                               insert is smaller than
                                               the info of first node.
                                               so we have to insert at
                                               Beg. so we return LOC
                                               =NULL

3. Set SAVE = start                         || initializes the pointers.
   and ptr = LINK[start]

4. Repeat step 5 & 6 while ptr ≠ NULL       || loop begins.
5. If ITem < INFO[Ptr]
   then set LOC = SAVE & return

6. Set SAVE = ptr                           || update the pointers.
   and ptr = LINK[ptr]

   End of loop
7. Set LOC = SAVE
8. Return                                   || return to calling function
```

Now we will call the FIND algorithm for location & then accordingly we will insert the item.

```
insert_location ( )
{
1. Call FIND ( );
2. If AVAIL = NULL
   then write overflow & Exit  || when avail list is empty i.e no free space available for creating a new node.
3. Set ptr = AVAIL
   and AVAIL = LINK[AVAIL]  || Removing the first node from Avail list & updating the Avail pointer
4. Set INFO[ptr] = item || copies new data into new node.
5. If LOC = NULL then: || Insert at first node
   Set LINK[ptr] = start || Making new node as first node of list
   and start = ptr || start pointing at first node of list.
   Else
   Set LINK[ptr] = LINK[LOC] || Inserting after node with location LOC
   and LINK[LOC] = ptr
   End of if
6. Exit
```

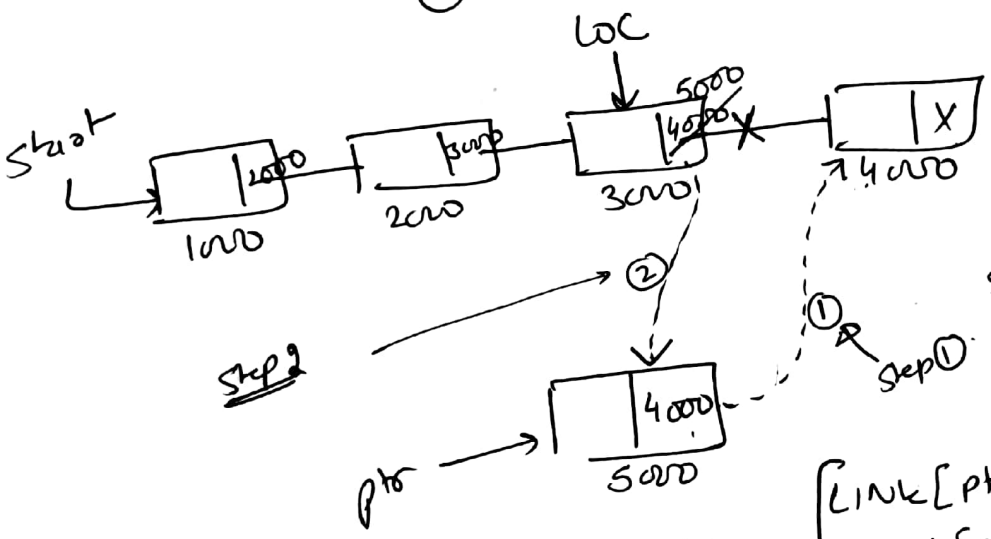
## Pictorial Representation:




If loc is pointing at last node of original list

① LINK[ptr] = LINK[loc] (Null)
LINK[ptr] = NULL

② LINK[loc] = ptr
LINK[loc] = 5000




Say loc is pointing at 3rd node. So we have to insert after 3rd node

① i.e [LINK[ptr] = LINK[loc]
LINK[ptr] = 4000

② [LINK[loc] = ptr
LINK[loc] = 5000

So I had tried to show you through pictorial representation of insertion.