Lecture 22

Stacks

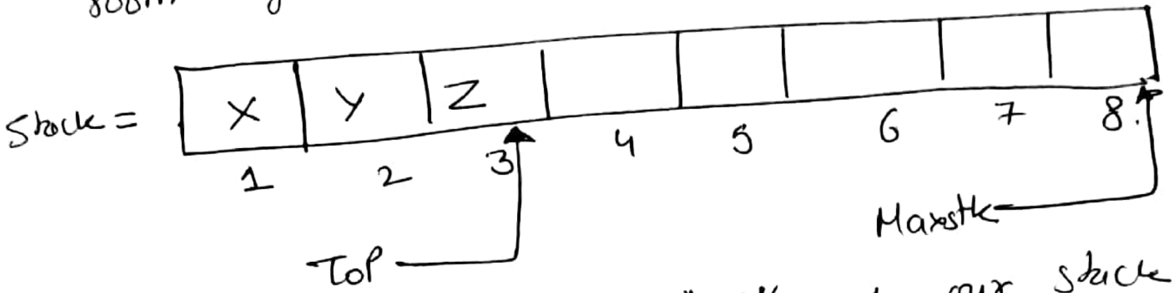
A stack is a linear structure in which items may be added or removed only at one end. This means, that the last item to be added to a stack is the first item to be removed.

- Stacks are also called last-in-first-out (LIFO)
- Everyday example of such a structure is : a stack of dishes, a stack of folded towels etc
- An element may be inserted or deleted only at one end called top of stack.
- Basic operation associated with stacks are
  - ⇒ Push : term used to insert an element
  - ⇒ Pop : term used to delete an element.

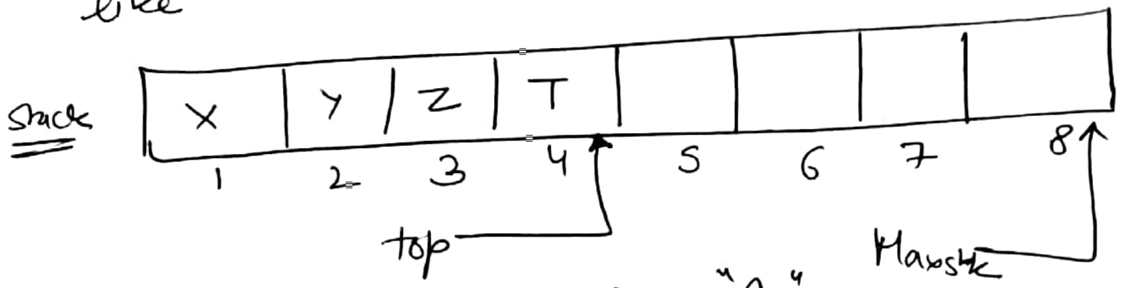
Array Representation of stack

- Top : a pointer variable which contain location of the top element of the stack
- Maxstk : a variable which gives the maximum no. of elements that can be held by the stack
- If TOP = 0 or TOP = NULL :→ Indicate that the stack is empty

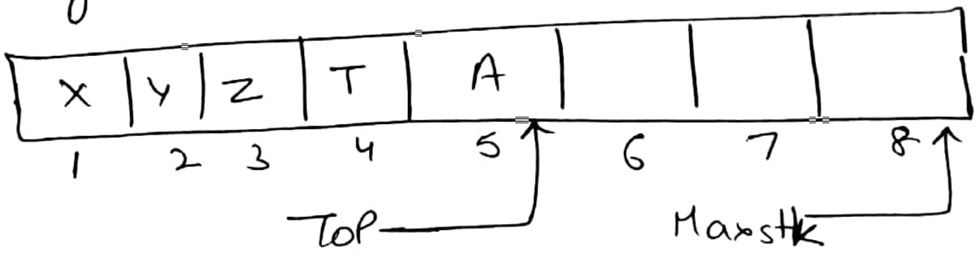
say Maxstk = 8.
and say TOP = 3
This mean stack has 3 elements and there is room for 5 more items in the stack.



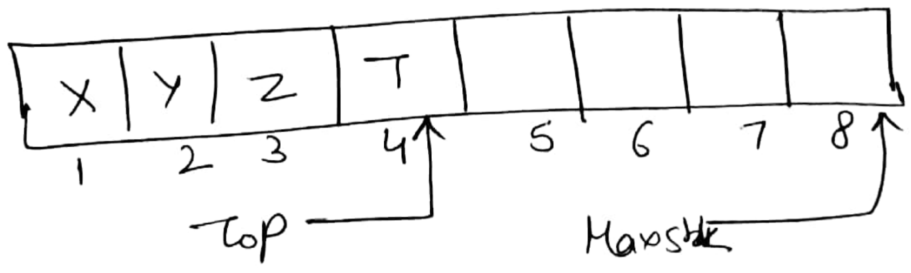
Say We want to push "T" so our stack will look like



Say we want to Push "A"



Now If we want to Pop an item ∴ the item which was inserted last will be the first to be deleted



After Pop : our stack look like above i.e $\boxed{A}$ is deleted.

① The follawing procedure pushes an Item onto a stack

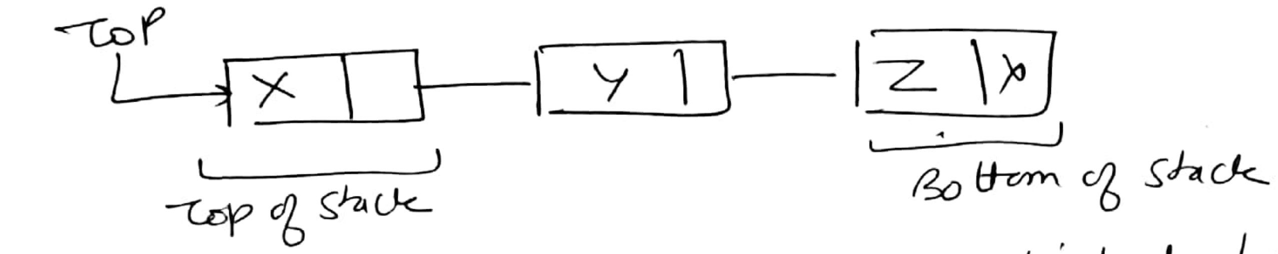
```
PUSH (      )
{
1. If ToP == MAXstk then        || stack is already filled.
   Print : overflow & return
2. Set ToP = top+1  || Increase top by 1
3. Set stack [ToP] = item  || Insert item in new Top position.
                            || Here stack is the name of the array.
4. Return
```

② The following procedure ~~pro~~ deletes the top element of stack and assigns it to the variable item.

```
Pop (   )
{
1. If Top == 0   then :        || stack is empty.
   Print ; Underflow.and return
2. Set ITEM = Stack [Top] || Assign top element to variable item.
3. Set Top = Top-1  || Decrease top by 1
4. Return.
```

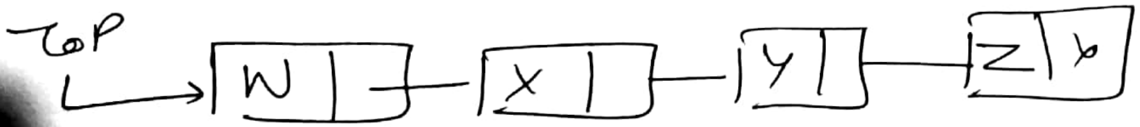
Linked Representation of Stacks

- Commonly termed as linked stack
- INFO field of the nodes hold the elements of the stack
- LINK field hold pointers to the neighbouring element in the stack
- ~~Start~~ START pointer behaves as the TOP pointer variable of the stack
- NULL pointer of the last node in the list indicate the bottom of stack



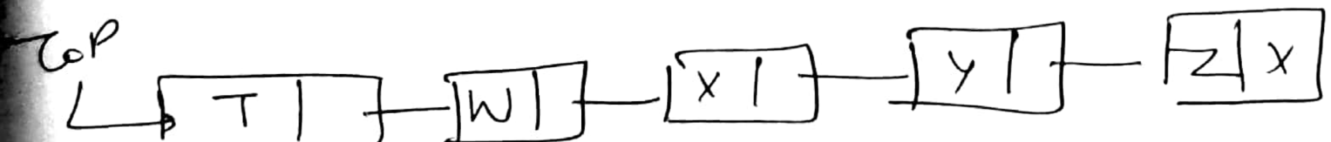
A Push operation into stack is accomplished by inserting a node into the front or start of the list.

Push "W" into stack



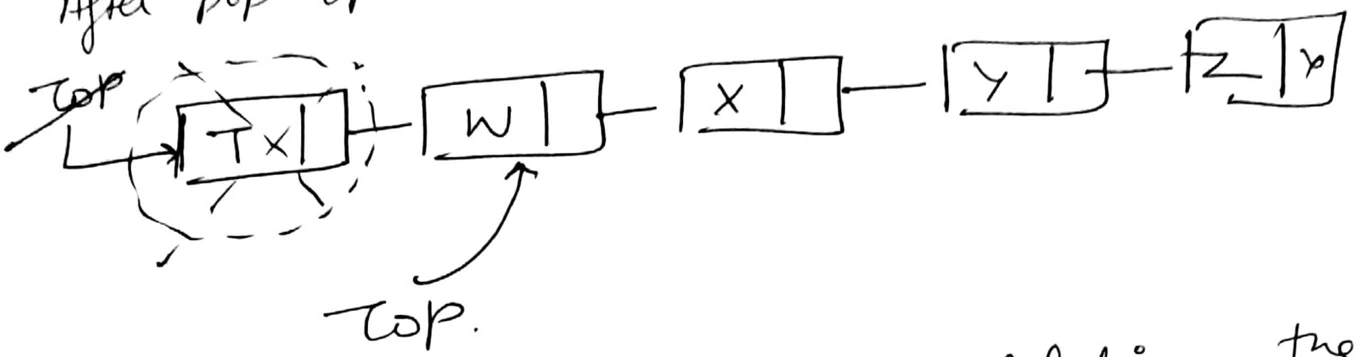
ie insertion at the beginning of the list.

Push "T" into stack



Pop from stack

After pop operation stack will look like



Pop operation is undertaken by deleting the node pointed by Top pointer.
i.e deletion at Beginning of the list.

The following procedure pushes an item into a linked ~~list~~ stack

```
Push_Linked Stack (  )
{
1.  If AVAIL = NULL, the write        || checking for
        Overflow & Exit               || availability of space

2.  Set ~~New~~ Temp = AVAIL          || You can simply create
        and AVAIL = LINK [AVAIL]      || the node through malloc()
                                      || or new

3.  Set INFO [Temp] = item            || copies item into new node

4.  Set LINK [Temp] = Top             || new node points to original
                                      || top node in the stack
5.  Set Top = Temp                    || reset top to point to new node
                                      || at the top of stack

6.  Exit
```

The following procedure deletes the top element of a linked stack & assign it to the variable item

```
Pop-linked Stack (    )
{
1. If Top = = NULL then                  || when stack is empty.
   Write: Underflow & Exit.
2. Set ITEM = INFO [TOP]                 || copies the top element of stack into item
3. Set TEMP = TOP →                      || To remember the old value of the Top pointer in Temp
   and TOP = LINK [TOP] →                || Reset Top to point to the next element in stack
4. Free (temp) ;
5. Exit.
```