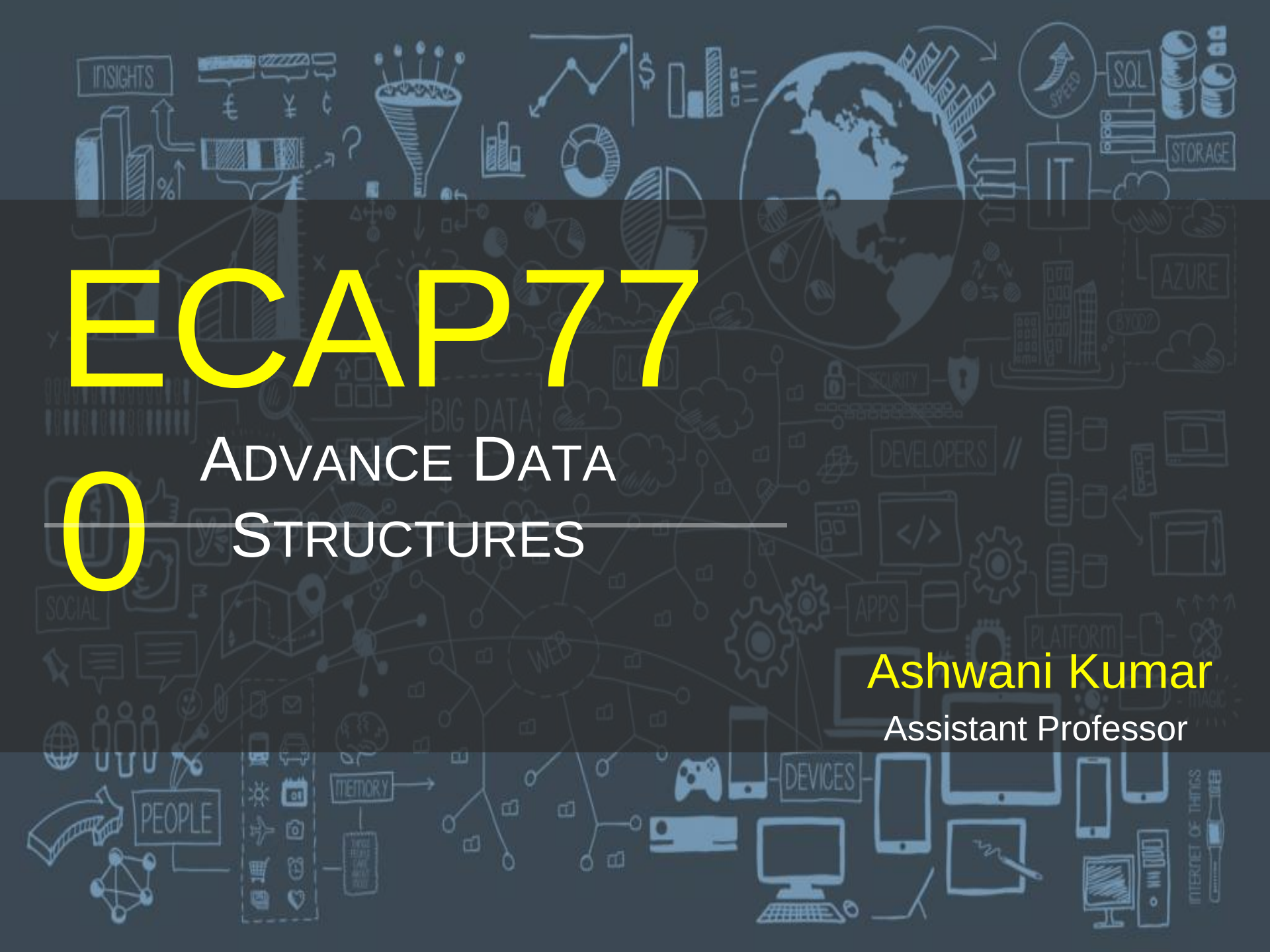


ECAP770

Advance Data Structures

Ashwani Kumar
Assistant Professor

Learning Outcomes

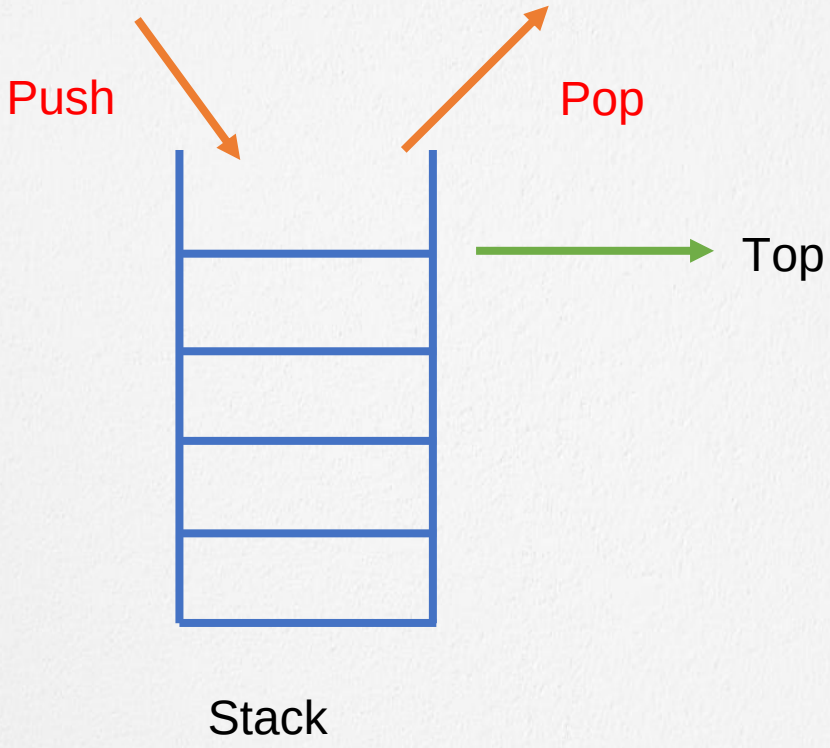
After this lecture, you will be able to

- understand implementation of stacks
  - Using Array
  - Using linked list

Stack

- A Stack is a linear data structure that follows the LIFO (Last-In-First-Out) or FILO(First In Last Out) principle.
- Stack has one end, insertion and deletion can be done from the one end known as the top of the stack.

Stack



Stack implementation

- Using array
- Using linked list

Stack implementation using array

- In array implementation, the stack is formed by using the array.
- All the operations regarding the stack are performed using arrays.

Adding an element onto the stack

Adding an element into the top of the stack is referred to as push operation.

- Increment the variable **Top** so that it can now refer to the next memory location.
- Add element at the position of incremented top. This is referred to as adding new element at the top of the stack.

Algorithm – for push operation

```
begin
    if top = n then stack full
    top = top + 1
    stack (top) : = item;
end
```

Implementation of push operation

```
void push (int val,int n) {
    if (top == n )
    printf("\n Overflow");
    else
    {
    top = top +1;
    stack[top] = val;
    }
}
```

Deletion of an element from a stack

Deletion of an element from the top of the stack is called pop operation. The value of the variable top will be decremented by 1 whenever an item is deleted from the stack. The top most element of the stack is stored in another variable and then the top is decremented by 1.

Algorithm – for pop operation

```
begin
    if top = 0 then stack empty;
    item := stack(top);
    top = top - 1;
end;
```

Implementation of pop operation

```
int pop ()
{
    if(top == -1)
    {
        printf("Underflow");
        return 0;
    }
    else
    {
        return stack[top - - ];
    }
}
```

Visiting each element of the stack

- Peek operation involves returning the element which is present at the top of the stack without deleting it.
- An underflow condition can occur if we try to return the top element to an already empty stack.

Algorithm

```
PEEK (STACK, TOP)

Begin

    if top = -1 then stack empty

    item = stack[top]

    return item

End
```

Implementation of Peek algorithm

```
int peek()
{
    if (top == -1)
    {
        printf("Underflow");
        return 0;
    }
    else
    {
        return stack [top];
    }
}
```

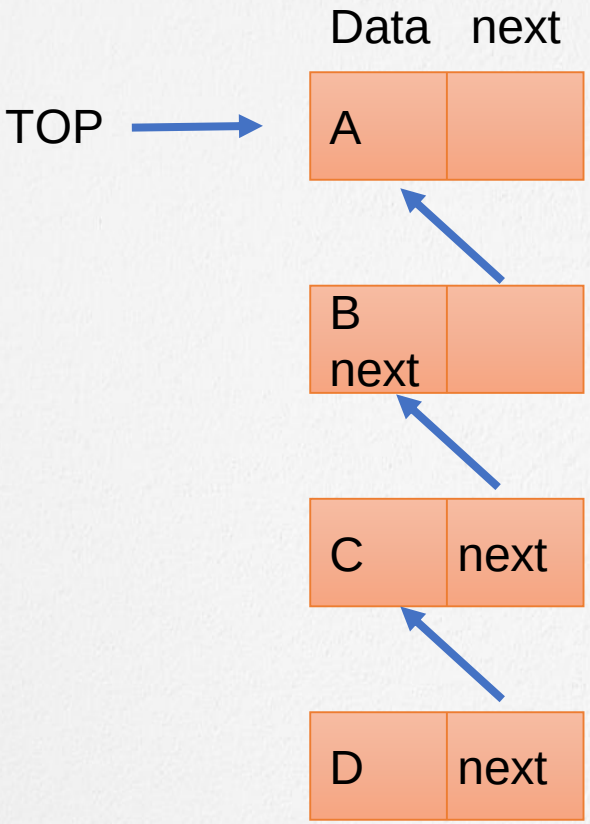
Stack Implementation using linked list

In Linked list memory is allocated dynamically. However, the time complexity in both the scenario is the same for all the operations, i.e. push, pop and peek.

Stack Implementation using linked list

- In linked list implementation of the stack, we need to create nodes and the nodes are maintained non-contiguously in the memory.
- Each node contains a pointer to its immediate successor node in the stack.
- Stack is said to be overflown if the space left in the memory heap is not enough to create a node.

Stack Implementation using linked list



Adding a node to the stack

Adding a node to the stack is referred to as a push operation.

- Create a node first and allocate memory to it.
- If the list is empty then the item is to be pushed as the start node of the list. This includes assigning value to the data part of the node and assign null to the address part of the node.

Adding a node to the stack

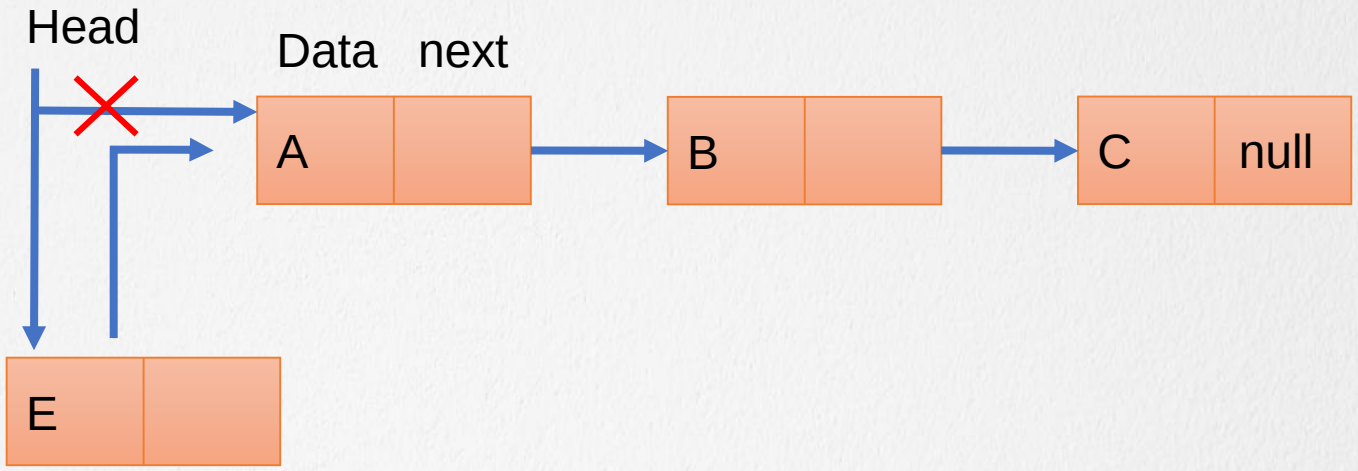


Fig. 1

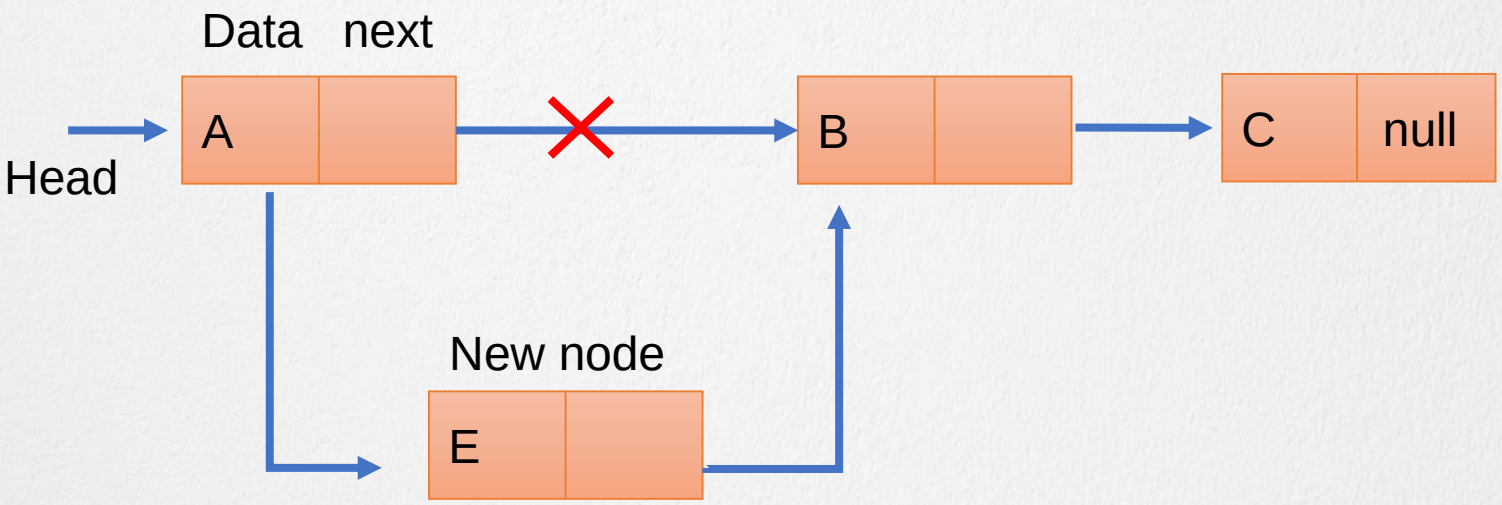


Fig. 2

Implementation of Push Operation

```
void push ()
{
    int val;
    struct node *ptr =(struct
node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("not able to push the
element");
    }
else
    {
        printf("Enter the value");
        scanf("%d",&val);
        if(head==NULL)
        {
```

Implementation of Push Operation

```
ptr->val = val;
        ptr -> next = NULL;
        head=ptr;
    }
    else
    {
        ptr->val = val;
        ptr->next = head;
        head=ptr;
    }
    printf("Item pushed");
  }
}
```

Deleting a node from the stack

- Deleting a node from the top of stack is referred to as pop operation.
- Underflow condition is checked. The underflow condition occurs when we try to pop from an already empty stack.
- The stack will be empty if the head pointer of the list points to null.

Deleting a node from the stack

- In a stack, the elements are popped only from one end, therefore, the value stored in the head pointer must be deleted and the node must be freed.
- The next node of the head node now becomes the head node.

Implementation of Pop Operation

```
void pop()
{
    int item;
    struct node *ptr;
    if (head == NULL)
    {
        printf("Underflow");
    }
    else
    {
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped");
    }
}
```

That’s all for now…