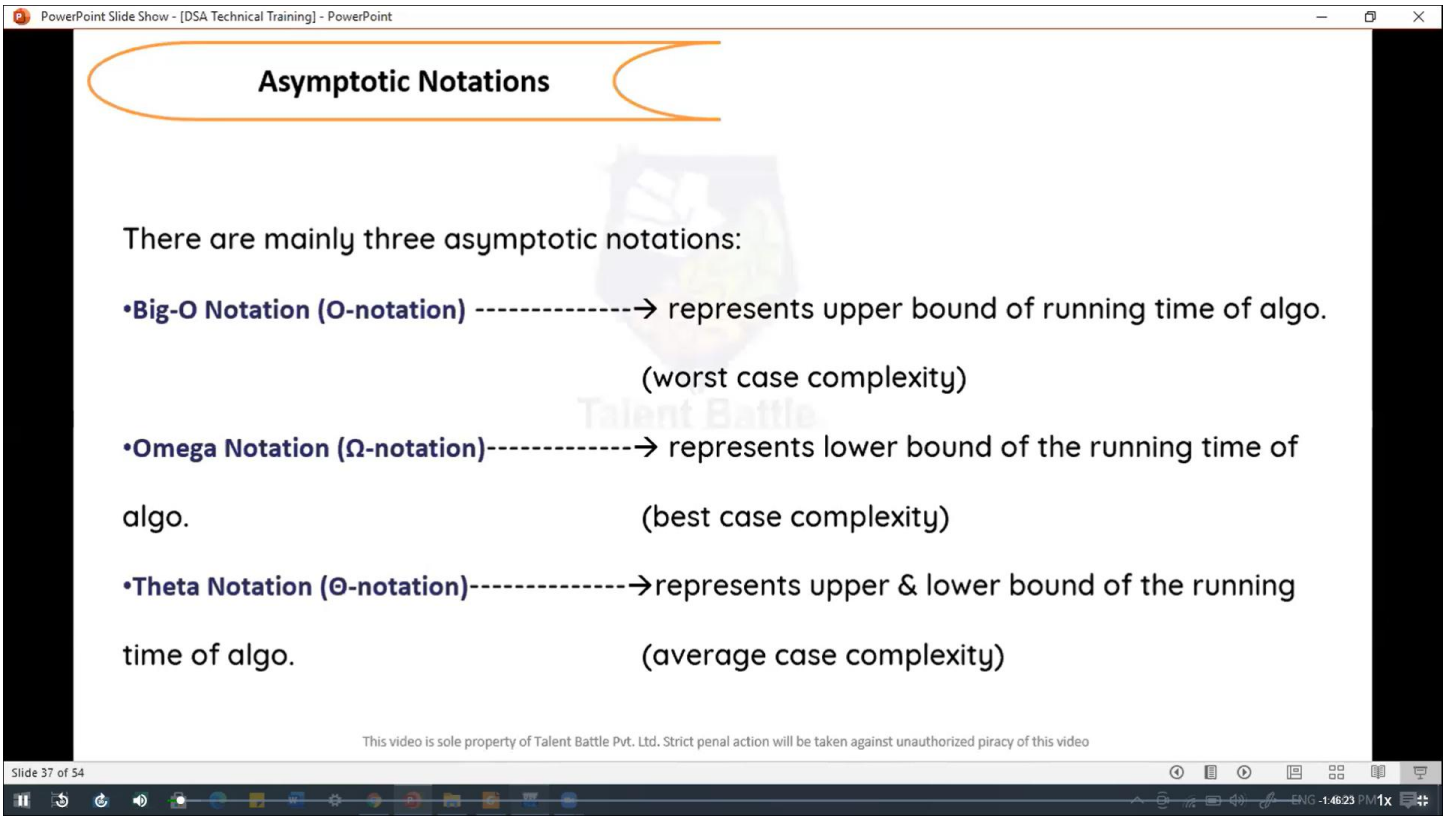
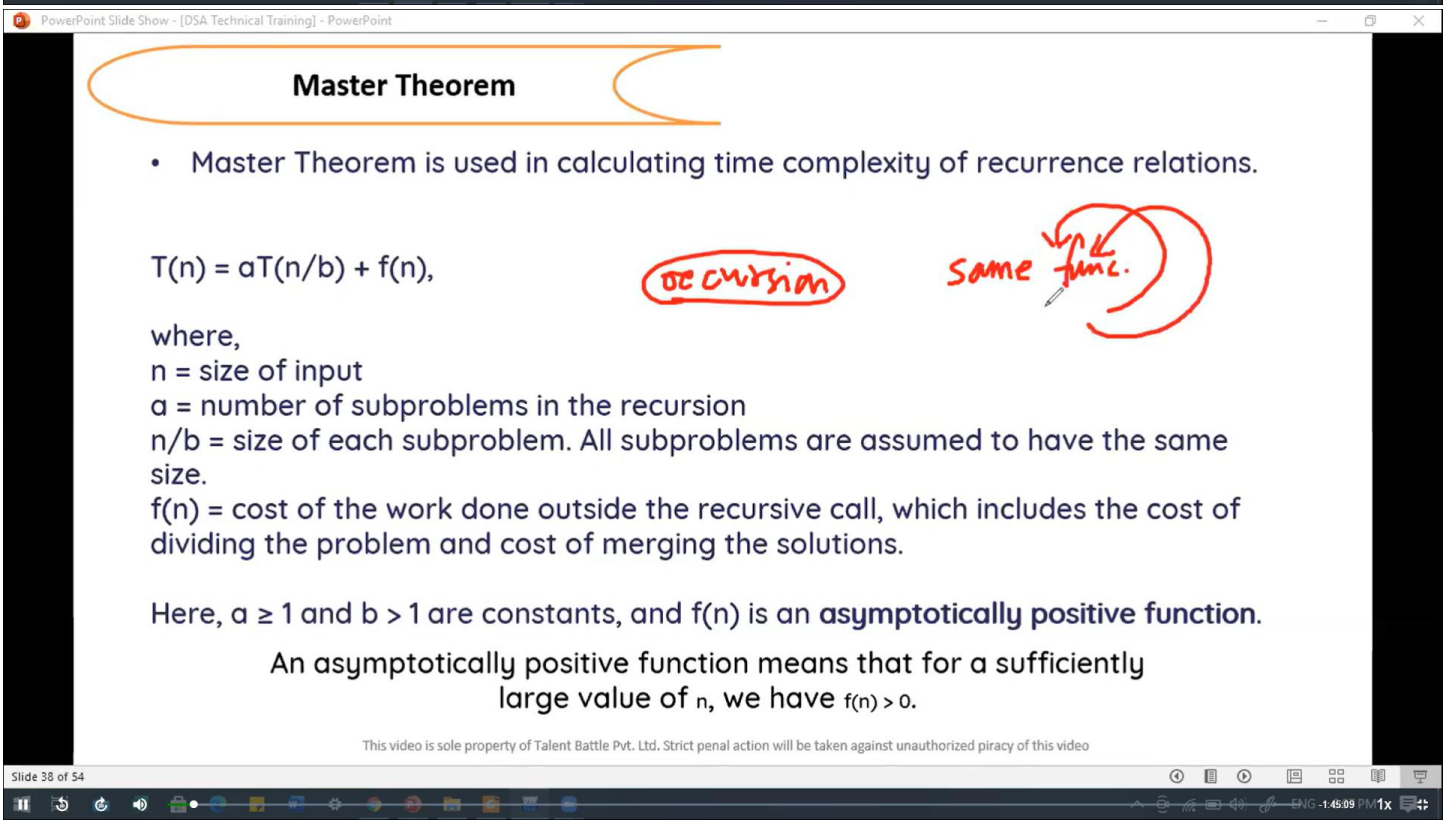
Day3

Stack Queue Part 1



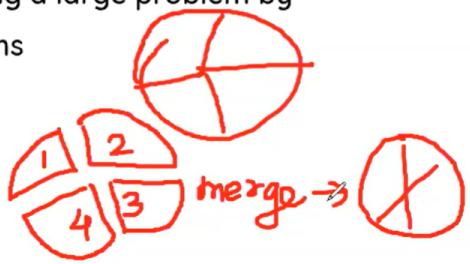


PowerPoint Slide Show - [DSA Technical Training] - PowerPoint

Divide and Conquer Algo.

A **divide and conquer algorithm** is a strategy of solving a large problem by

1. breaking the problem into smaller sub-problems
2. solving the sub-problems, and
3. combining them to get the desired output.



How Divide and Conquer Algorithms Work?

Here are the steps involved:

1.Divide: Divide the given problem into sub-problems using recursion.

2.Conquer: Solve the smaller sub-problems recursively. If the subproblem is small enough, then solve it directly.

3.Combine: Combine the solutions of the sub-problems that are part of the recursive process to solve the actual problem.

PowerPoint Slide Show - [DSA Technical Training] - PowerPoint

Divide and Conquer Algo.

Advantages:

•The complexity for the multiplication of two matrices using the naive method is $O(n^3)$, whereas using the divide and conquer approach (i.e. Strassen's matrix multiplication) is $O(n^{2.8074})$. This approach also simplifies other problems, such as the Tower of Hanoi.

•This approach is suitable for multiprocessing systems.

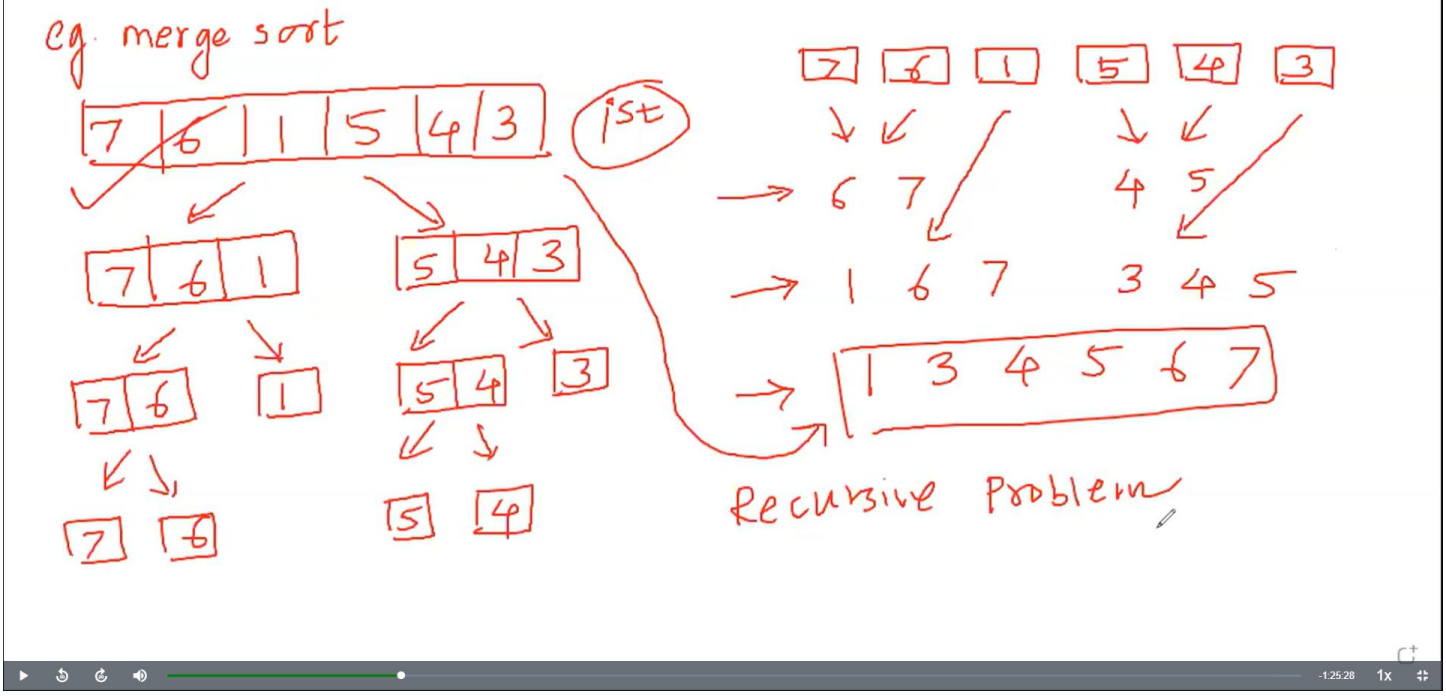
•It makes efficient use of memory caches.

PowerPoint Slide Show - [DSA Technical Training] - PowerPoint

Divide and Conquer Algo.

Applications:

- Binary Search
- Merge Sort
- Quick Sort
- Strassen's Matrix Multiplications
- Karatsuba Algorithm



$a = 2$

$T(n) = aT(n/b) + f(n)$

$n/b = n/2$

$f(n) = O(n)$

$T(n) = 2T(n/2) + O(n)$

$T(n) = O(n \log n)$

-1:22:37 1x

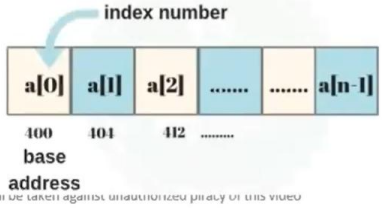
PowerPoint Slide Show - [DSA Technical Training] - PowerPoint

Array

Instead of creating separate variables to store data, it is an efficient organization to store data in a single variable called array

Definition: An array is defined as a collection of items stored at contiguous memory locations under the same name

Ex: int (a,b,c,d,e)can be grouped in a single variable as int a[5]: now five continuous memory location are assigned with the same name 'a'



ENG 6:30 PM

Array

```
#include<iostraem>
using namespace std;
int main()
{
int a[5]={6,9,1,5,3};              //direct declartion and initialisation
cout<<"elements are:\n";
for(int i=0;i<5;i++)               // iterate all the elements
cout<<"a["<<i<<"]="<<a[i]<<"\n";        //access though index
return 0;
}
```


Array: Practice Examples

1. C++ program to read array of size n and find the frequency of the given element.
2. C++ program to find the array type (even, odd or mixed)

Array: Imp Points

1. An array index cannot be negative or zero.
2. Unused space in the array is always filled with zeros.
3. An array elements are Garbage values before initialization of values.
4. Declaring an array without size is invalid.

STACK Data Structure

- A stack is a useful data structure in programming.
- It is just like a pile of plates kept on top of each other.

Think about the things you can do with such a pile of plates

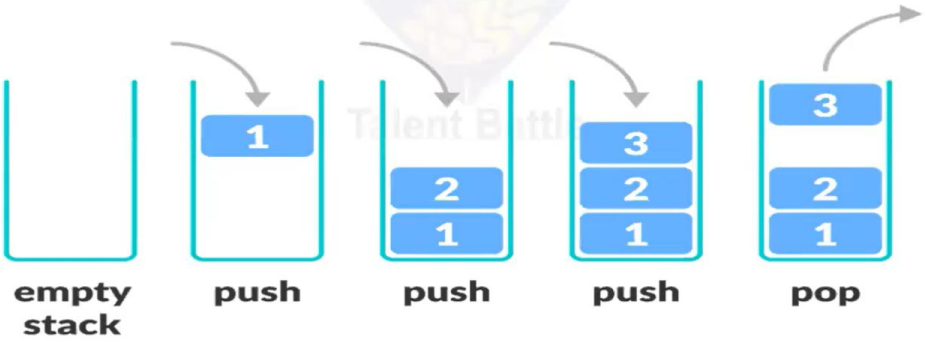
•Put a new plate on top
•Remove the top plate

If you want the plate at the bottom, you must first remove all the plates on top. Such an arrangement is called **Last In First Out** - the last item that is the first item to go out.

PowerPoint Slide Show - [DSA Technical Training] - PowerPoint

LIFO Principle of Stack

In programming terms, putting an item on top of the stack is called **push** and removing an item is called **pop**.



PowerPoint Slide Show - [DSA Technical Training] - PowerPoint

Basic Operations of Stack

A stack is an object (an abstract data type - ADT) that allows the following operations:

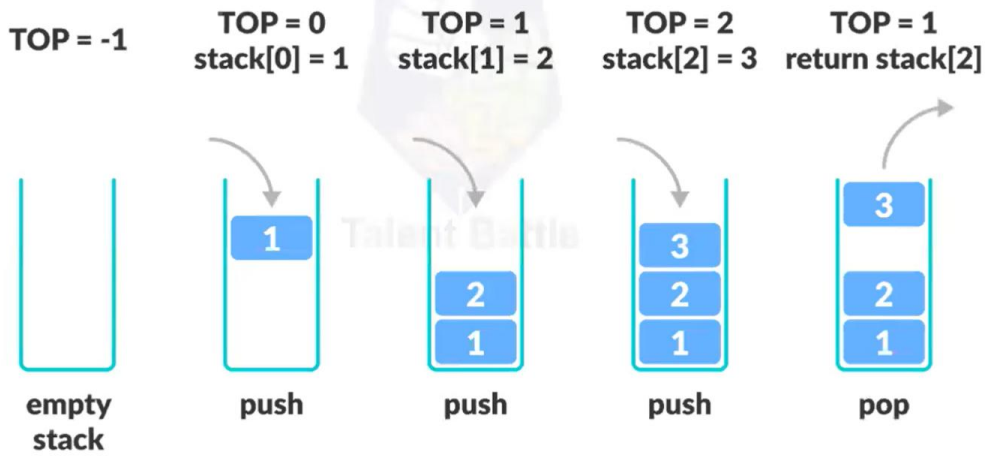
- **Push**: Add an element to the top of a stack
- **Pop**: Remove an element from the top of a stack
- **IsEmpty**: Check if the stack is empty
- **IsFull**: Check if the stack is full
- **Peek**: Get the value of the top element without removing it

Working of Stack Data Structure

The operations work as follows:

1. A pointer called TOP is used to keep track of the top element in the stack.
2. When initializing the stack, we set its value to -1 so that we can check if the stack is empty by comparing TOP == -1.
3. On pushing an element, we increase the value of TOP and place the new element in the position pointed to by TOP.
4. On popping an element, we return the element pointed to by TOP and reduce its value.
5. Before pushing, we check if the stack is already full
6. Before popping, we check if the stack is already empty

Working of Stack Data Structure



Stack Time Complexity

For the array-based implementation of a stack, the push and pop operations take constant time, i.e. O(1).

Stack Applications

Applications of Stack Data Structure

Although stack is a simple data structure to implement, it is very powerful. The most common uses of a stack are:

- **To reverse a word** - Put all the letters in a stack and pop them out. Because of the LIFO order of stack, you will get the letters in reverse order.
- **In compilers** - Compilers use the stack to calculate the value of expressions like 2 + 4 / 5 * (7 - 9) by converting the expression to prefix or postfix form.
- **In browsers** - The back button in a browser saves all the URLs you have visited previously in a stack. Each time you visit a new page, it is added on top of the stack. When you press the back button, the current URL is removed from the stack, and the previous URL is accessed.