DSA LAUNCHPAD 4.0

Introduction to DSA

Presentors:

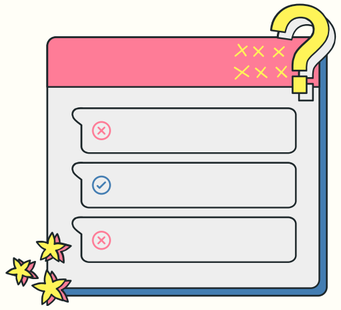
Pankaj Bhosale

Pratiksha Brahmne

Shantanu Kantak

Sumit Padadune

1 Long Challenge
1 Short Challenge



Quiz

DSA Launchpad
Champion Quiz Master

Contents

- Introduction & Basics (Pointers, Recursion,Datatypes, Function)
- Arrays & Strings
- Linked Lists
- Searching & Sorting Algorithms
- Stacks, Queue, DeQueue
- Trees (Implementation & Traversal)
- Basic Graph Algorithms

Data Structure

It is a way of arranging data on a computer,so that it can be accessed and updated efficiently.

types of data structure

Linear

- Array
- linked list
- stack
- Queue

Non Linear

- tree
- Graph

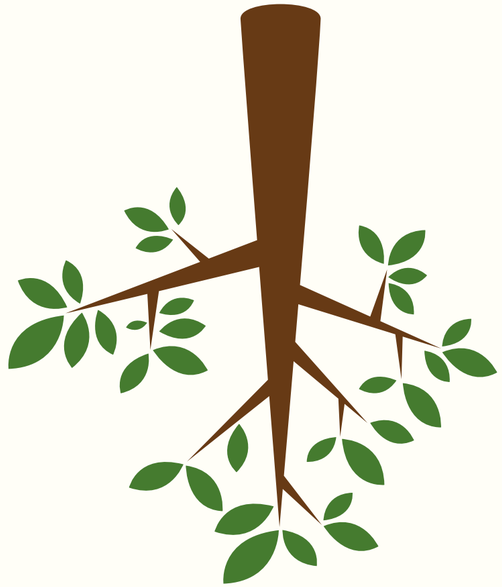
Stack

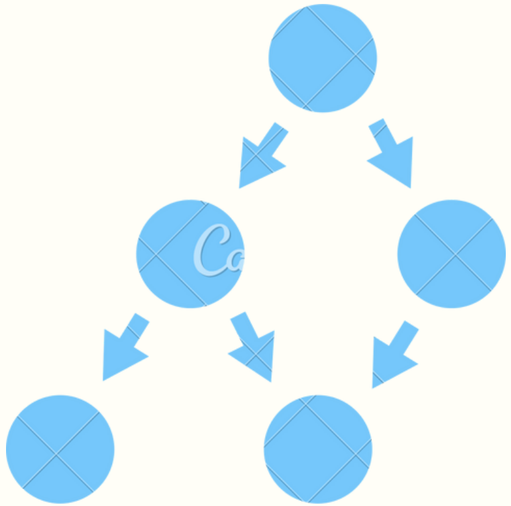
last In first out

Queue

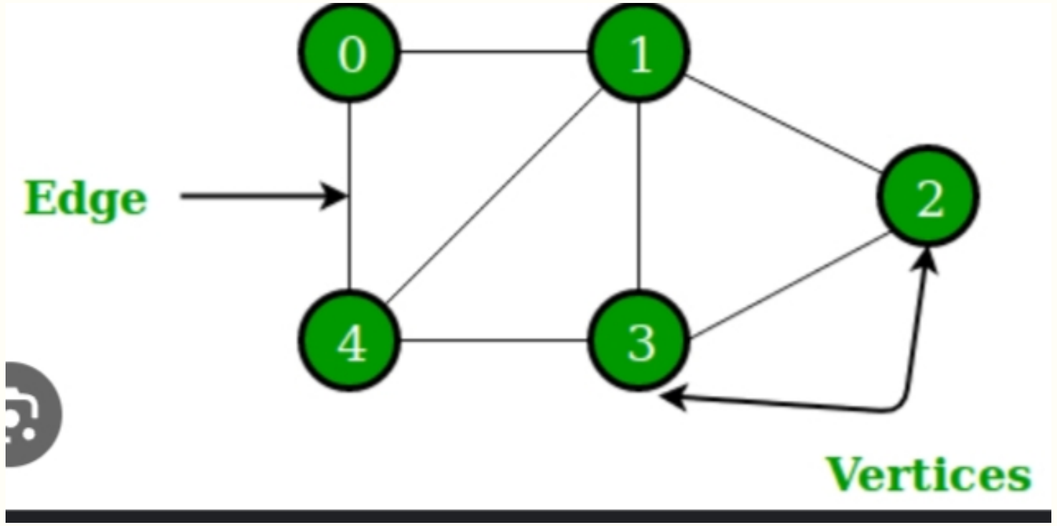
First In First Out

Tree



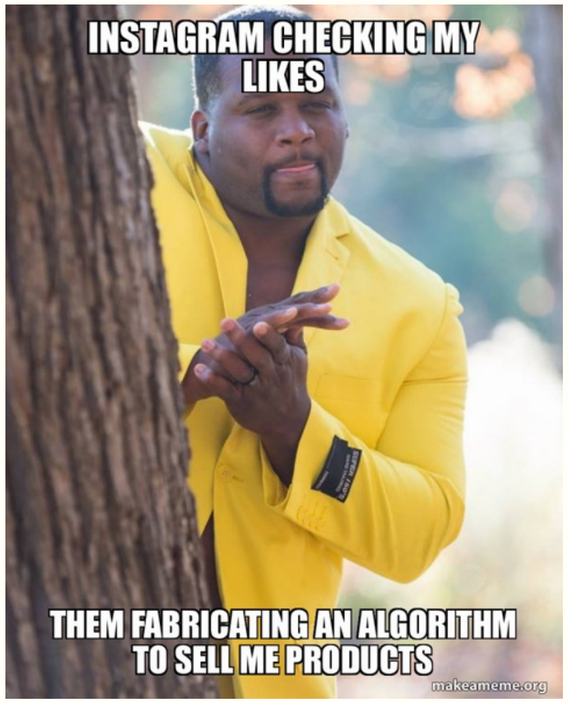


Graph



Algorithms

An algorithm is a set of well-defined instructions to solve a particular problem.
It takes a set of inputs and produces the desired output.

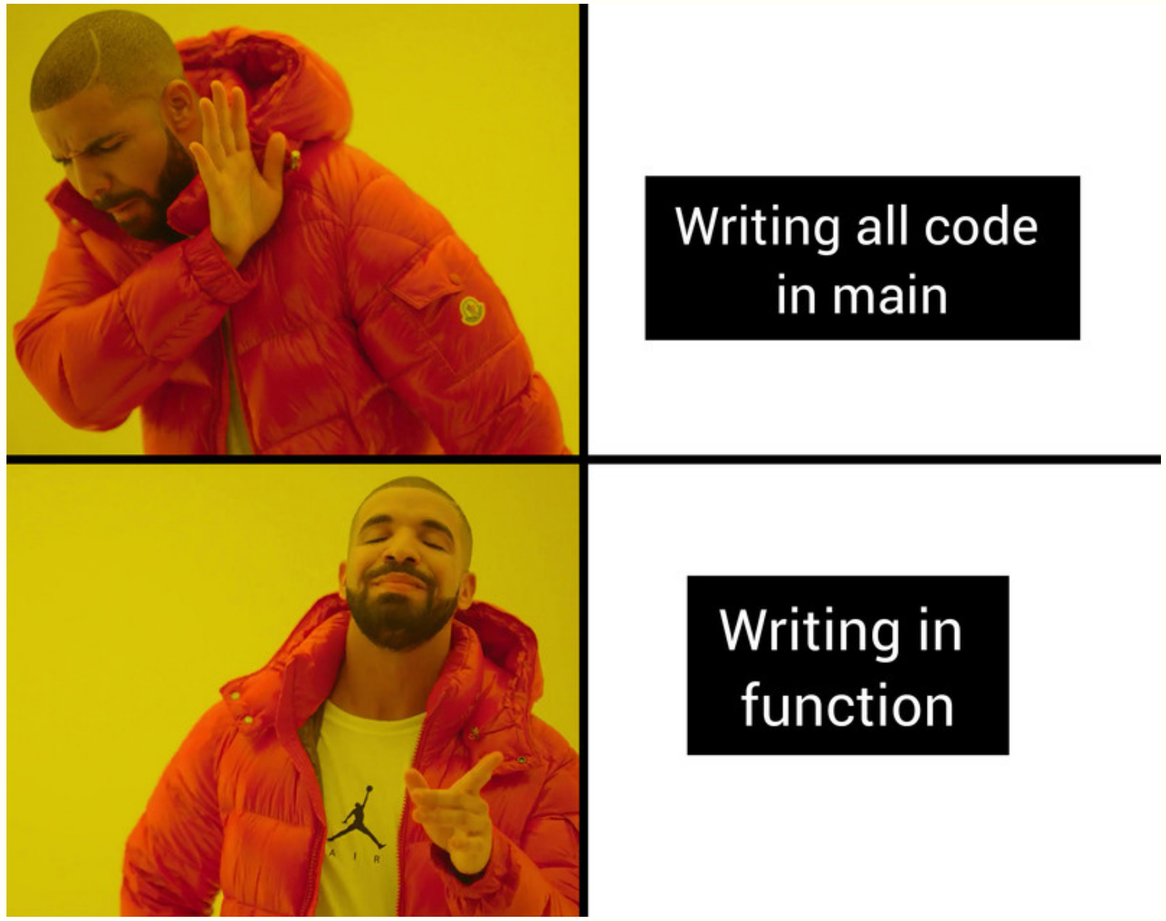


Functions

What is Function

Use of Function

Types of Function



What is Function?

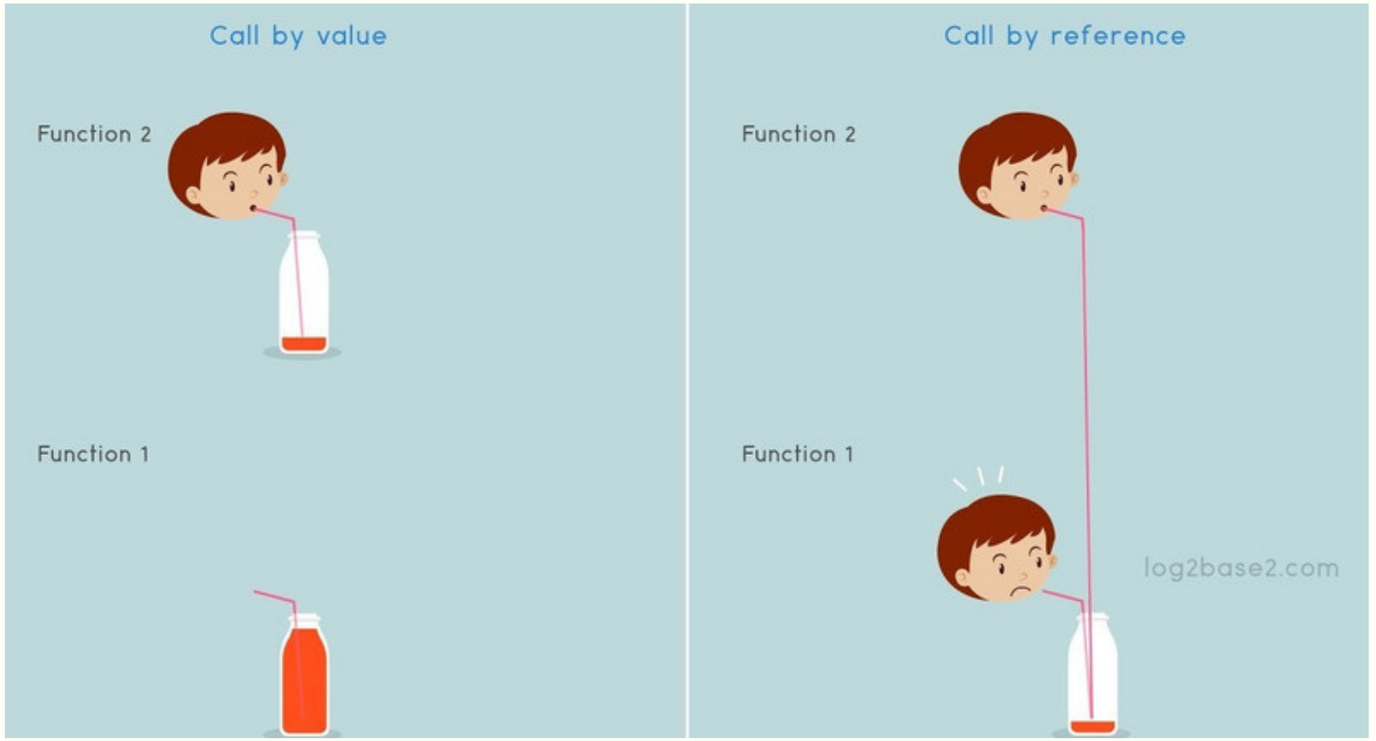
```
int add(int a,int b)
{
  //code
  return output
}
```

Self contained block which performs certain task

- Readability
- Reusability
- Debugging

How are functions called?

Pass by value Pass by reference



Pass By Value

```
x
void func(int a, int b)
{
    a += b;
    cout << "In func, a = " << a << " b = " << b << endl;
}
int main(void)
{
    int x = 5, y = 7;

    // Passing parameters
    func(x, y);
    cout << "In main, x = " << x << " y = " << y;
    return 0;
}
```

Pass By Reference

```
x
include <iostream>
using namespace std;
void swapnum(int &i, int &j)
{
    int temp = i;
    i = j;
    j = temp;
}

int main(void)
{
    int a = 10, b = 20;

    // passing parameters
    swapnum(a, b);

    cout << "a is " << a << " and b is " << b;
    return 0;
}
```

Guess The Output?

```
include <iostream>
using namespace std;

void param(char a, int b)
{
   cout<<a<<" "<<b<<endl;
}
int main(void)
{
    int a = 10, b = 20;
    string s;
    s.push_back('A');
    param(a,s);
    return 0;
}
```

Scope Of Functions

There are 4 types of scopes available in C++:

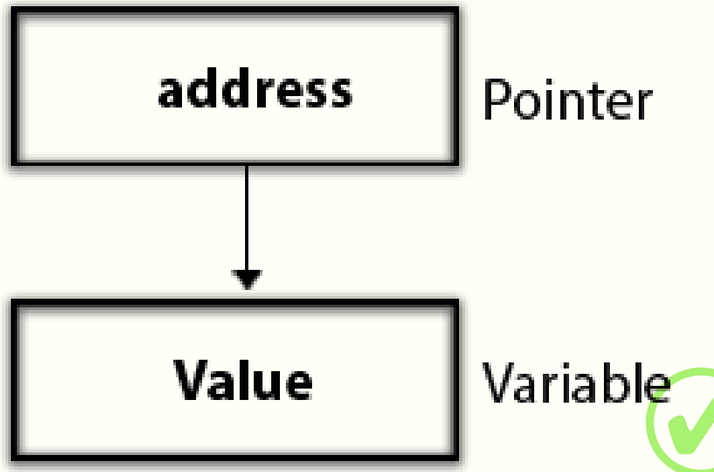
Local scope
Function scope
Global scope
Class scope

Pointers

Size of Pointer variable = 8 bytes.

Pointer may point to any datatype

can hold address of any variable of the datatype it is pointing to.



OUTPUT?

```
x
 #include<bits/stdc++.h>
 #include<iostream>
 using namespace std;
 int main(){
      int num = 25;
     int *p = &num;
     cout << &num << " " << p << *p <<endl;

     (*p)++;
     cout << num << " ";
   // Same Syntax
     int *n = 0;
     n = &num;
 return 0;
 }
```

```
x
#include<bits/stdc++.h>
 using namespace std;
 int main(){
     int k = 25;
     int *p = &k;

     int a = *p;
     cout<<"Value of a & k before = " ;
     cout<< a<< " " << k << endl;
     a++;
     cout<<"Value of a & k after = ";
     cout<< a << " " << k << endl;
 return 0;
 }
```

OUTPUT?

x

```
#include <iostream>
using namespace std;

int main(){
    int k = 29;
    int *p = &k;
    int a = *p;
    cout << "Adress of k = " << &k << " " << p << endl;

    cout << ++a <<" "<<"K = "<<k<<endl; //Increment value of a
    p += 65;

    cout << p <<" "<<(*p)<<endl; //first increment p and display *p;

}
```

OUTPUT?

```
x
include <iostream>
using namespace std;

int main(){
        int arr[4] = {2,7,8,9};
        cout << arr << endl;
        cout << &arr[0] << " ";
        cout << *arr << " ";
        cout << *(arr)+1 << endl;
        cout << *(arr+1) << endl;
}
```

OUTPUT?

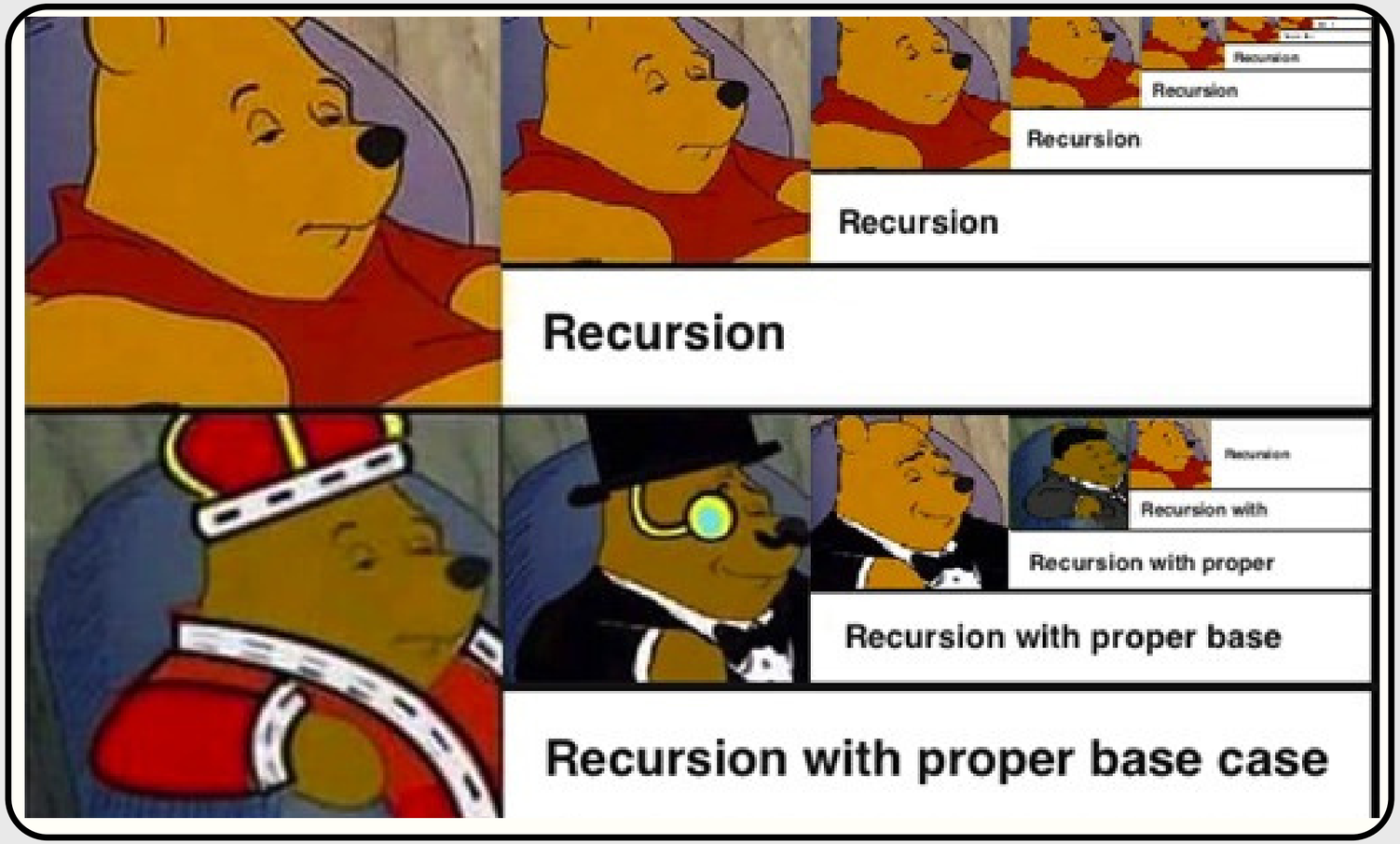
```
x
#include<bits/stdc++.h>
 using namespace std;
 int main( ){
   string s = "ABC";
   const char *ptr = s.c_str( );
   cout<< *ptr;
   // Or
   char *p = &s[0];

   cout << ptr <<" "<< p <<endl;
 return 0;
 }
```

...

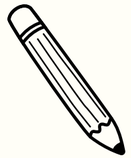
Recursion

Function calling itself?



Recursion

- Function calls itself directly or indirectly is called recursion
- Recursion needs a base case to stop its further execution



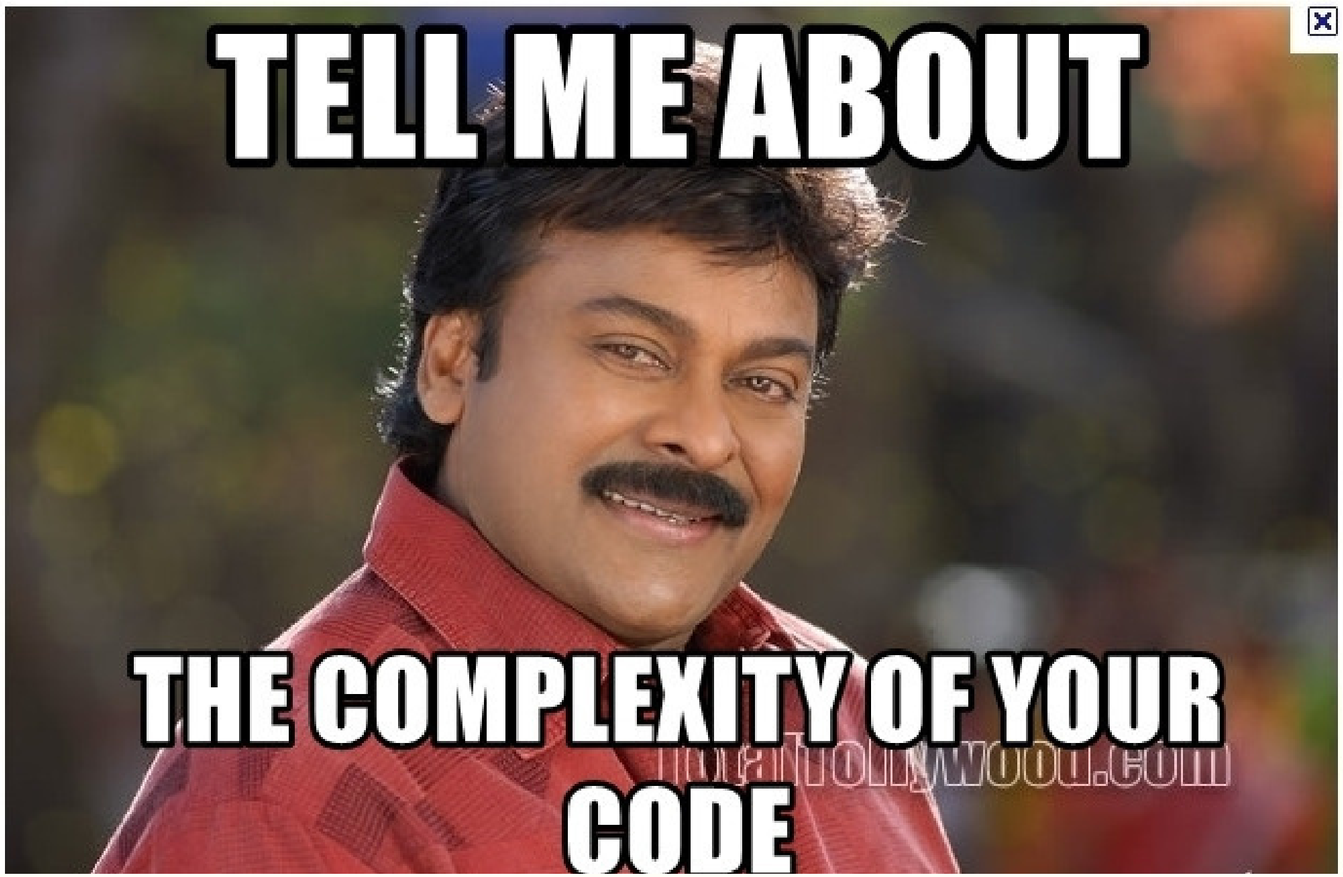
- Recursion uses stack memory for its execution
- If base case is not proper that may cause stack overflow

At each call of the function local copy of the function is created and when base case is reached these answers are returned

BackTracking

...

Algorithmic Complexities



Notations

\mathcal{O}	$\mathcal{O}$	Worst-Case
\Omega	Ω	Best-Case
\Theta	Θ	Average-Case

Time Complexity

- Amount of time required for the execution of an algorithm
- Highly depends on the size of processed data
- Effectiveness of an algorithm Evaluation of performance

Whenever time complexity is not given we can follow by this

n<=12 -O(n!)

n<=25 -O(2^n)

n<=100 -O(n^4)

n<=500 -O(n^3)

n<=10^4 -O(n^2)

n<=n^6 -O(nlogn)

n<=10^8 - O(n)

Types

O(1)	Constant time complexity
O(logn)	Logarithmic time complexity
O(n)	Linear time complexity
O(n^2)	Quadratic time complexity
O(n^3)	Cubic time complexity

Space Complexity

- Amount of memory a program uses in order to achieve its execution .
- Space complexity is auxiliary and input space helps evaluate a solution

Thank you

WCE ACM Student Chapter