# DYNAMIC PROGRAMMING
# (Very very imp.)
## Day 1

**Fibonacci Series:a series of numbers where each no(known as fibonacci no) is the sum of two preceding numbers..**

**series: 0 1 1 2 3 5 8 13..**

**Mathematically:**

**fib(i)=fib(i-1)+fib(i-2)**

**provided:**

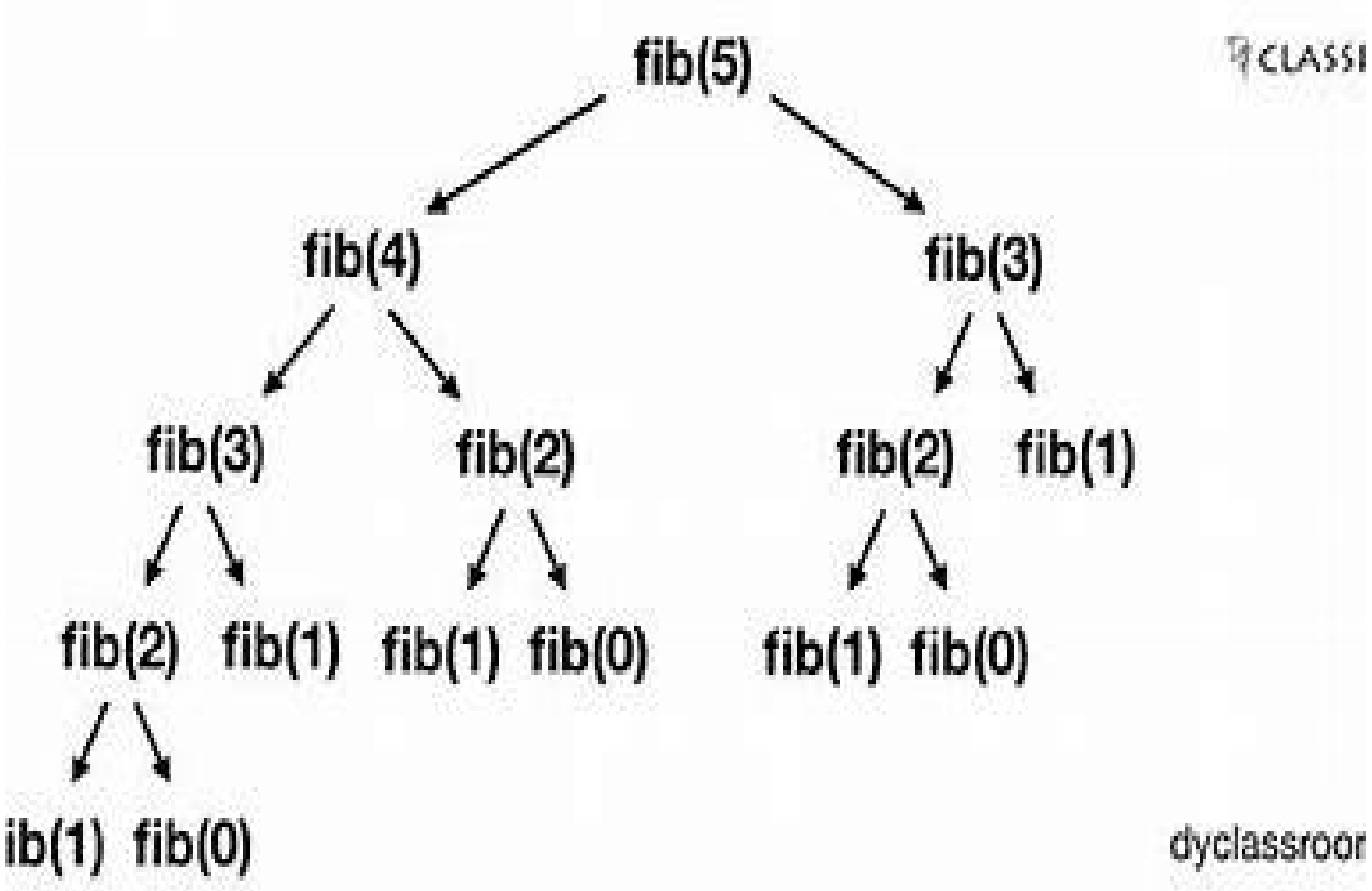
**fib(0)=0**

**fib(1)=1**

**Code(Recursive..)??**

```c
int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n-1) + fib(n-2);
}
```

## Recursion Tree:




Recursion Tree of fibonacci

## Time Complexity of fibonacci??

first layer: fib(5)=fib(4)+fib(3)-->O(1)-->1 unit time
second layer: fib(4),fib(3)-->2*O(1)-->2 units
third layer:
fib(3),fib(2),fib(2),fib(1)-->4*O(1)-->4 units
...
…n layers

1+2+4+8+16...nth term
sum=1*(2^n -1 )/(2-1)==(2^n-1)
O(2^n)-->time complexity..exponential

more exact time complexity:
O((1+sqrt(5))/2)^n)-->O((1.6180)^n)
1.6180-->golden ratio..

## Repetitions..

int dp[n+1]-->store all fibonacci upto n(0..n)
where:

dp[i]=any -ve no
dp[0]=0
dp[1]=1

dp[3]=x
fib(3)??
if(dp[3]>=0)-->yes
 use dp[3]

→**Memoization**..storing all the results we got in the past and then using it for present calculation..

Kya ukhaada isse??

fib(2)-->O(1)
fib(3)-->O(1)
fib(4)-->O(1)
fib(5)-->O(1)
…
fib(n)-->O(1)

n*O(1)-->O(n)-->time complexity-->linear space complexity-->O(n)-->linear..

expo-->linear time complexity..(ye ukhada hmne memoization se)..

## Code using memoisation ??

```
int dp[n+1]
for(int i=0;i<=n;i++)
 dp[i]=-1;

dp[0]=0
dp[1]=1

int fib(int n)
{
 if(n<=1)
  return n;
 if(dp[n]!=-1)//already calc
  return dp[n];
int x=fib(n-1)+fib(n-2);
dp[n]=x;
return x;
}
```

**DP == memoization+recursion**

**O(n)-->space complexity**

**Some general cases where dp is applied--:**

**1.what are the no of ways of doing this..,**
**2.what is the min/max possible value of this..**
**3.Can ram reach to the end of the city?**
**(yes/no type questions)**

**how to be sure about applying dp??**

**1.first think about recursion…**
**2.write recursive relation..**
**3.make recursion tree for any small example..**
**4.check if there is any repetition in the tree..**
**5.apply memoization..**

**Or**

→ breaking into subproblems and then check for repeated subproblems..
(this is called overlapping subproblems)

## Q)Factorial calculation:

factorial of any positive integer is the product of all positive integers less than equal to n..

fact(n)=n*(n-1)*(n-2)...3*2*1

assumption:fact(0)=1..

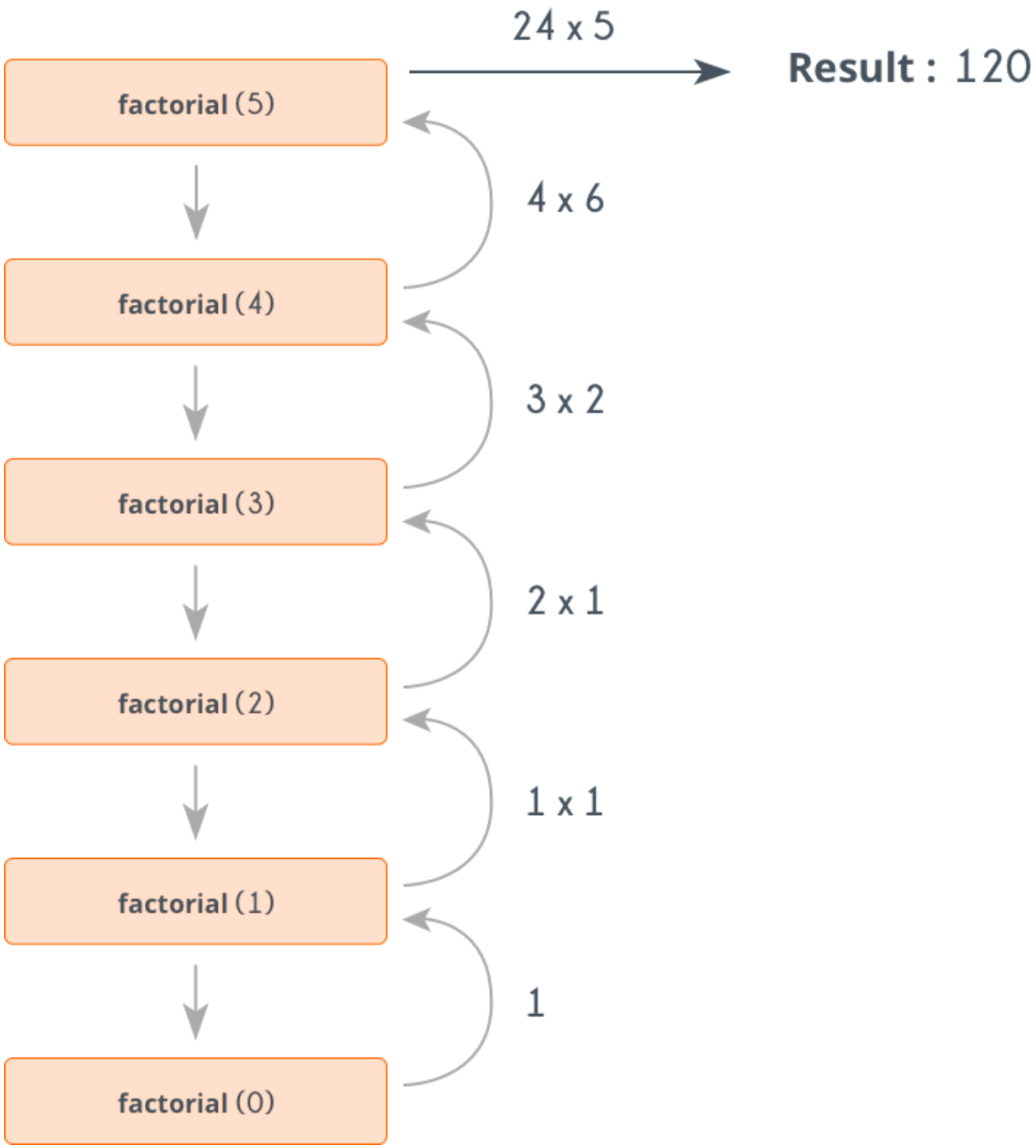
## recursive code??

```
int fact(int n)
{
  if(n==0)
    return 1;//base case
 return n*fact(n-1);
}
```

Is dp required for calculating fact(n) ??
Or Can we use DP to find factorial ?
No, DP can't be applied here, because no overlapping subproblems….. (Important)

**We hope, now the difference between normal recursion and dynamic programming is clear.**

## Recursion tree:




**Q)chintu(a legend from whitehat..)..standing at the ground(0th)..there are n ladders he needs to climb..1 ladder or 2 ladder he can climb..**

**ith ladder→(i+1) or (i+2)**

**In how many ways chintu can reach the nth ladder?**

**ith ladder..**
**2 ways→(i-1)th ladder**
**→ (i-2)th ladder..**
**f(i)-->no of ways of reaching ith ladder..**
**f(i)=f(i-1)**
**(i-2)th→(i-1)th**
**-->ith**

**(i-1)th ladder→(i-2)**
**f(i)=f(i-1)**

**4 ladders..**
**f(0)=1**
**f(1)=1**
**f(2)=f(1)+f(0)=1+1=2**
**f(3)=f(2)+f(1)+1=2+1+1=4**
**f(3)=f(2)+f(1)=2+1==3**
**f(i)=f(i-1)+f(i-2)**

**home--->city 1(5 routes)-->city 2(10 routes..)**
**city2 reach..no of ways??**
**5*10==50 ways**