C, C++, DSA in depth

Dynamic programming

Saurabh Shukla (MySirG)

Agenda

① what is dynamic Programming?

Dynamic Programming

Dynamic Programming (DP) is simply an optimization over plain recursion.

Whenever a recursive solution is repeatedly solves a subproblem due to overlapping in subproblems, time complexity can be reduced by storing the results of subproblems. thus it avoids re-computation.

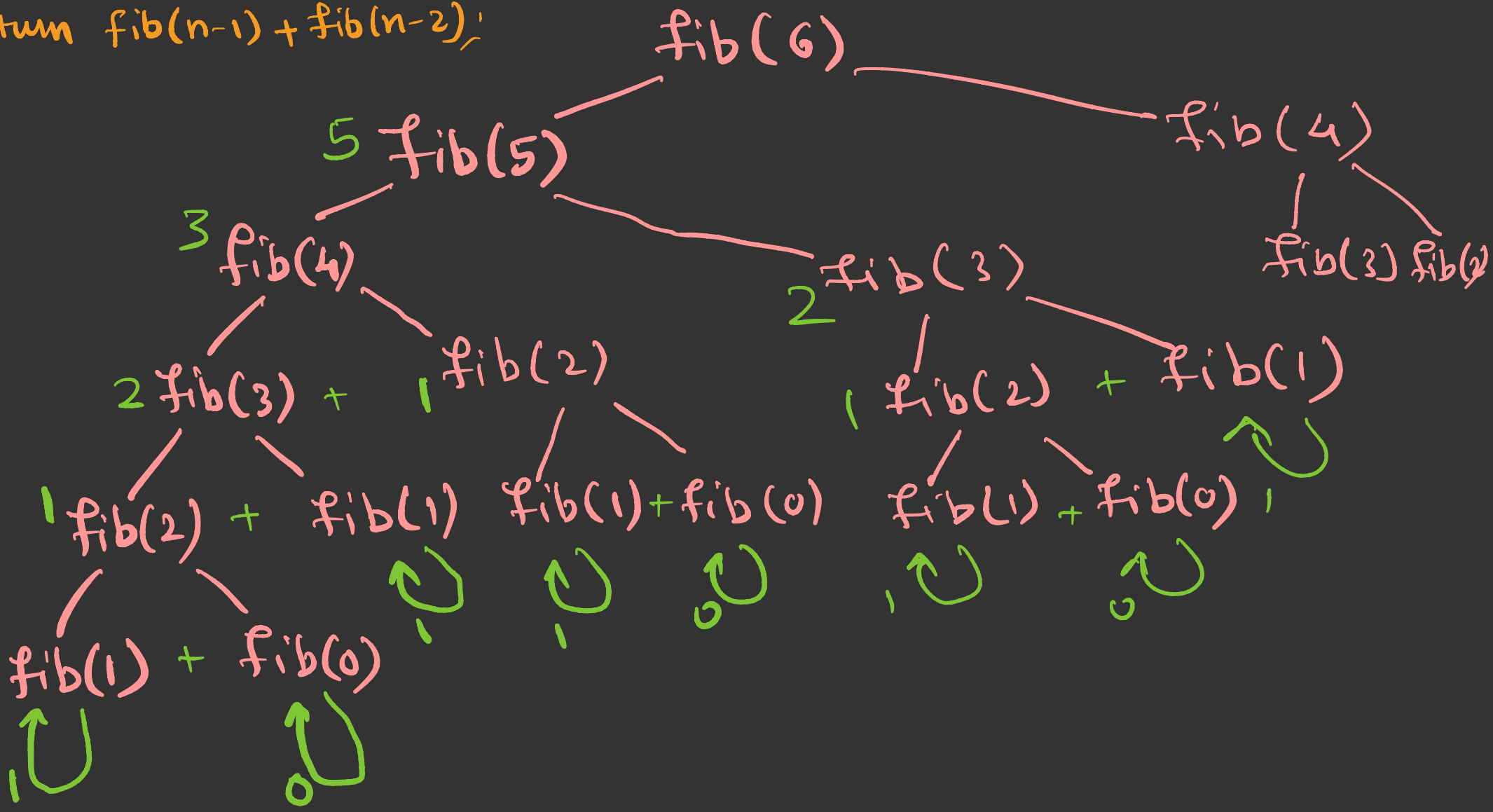
Write a recursive function to find n^{th} term of a fibonacci series.

0 1 1 2 3 5 8 13 21 34 55...

```
int fib(int n)
{
    if (n==0 || n==1)
        return n;
    return fib(n-1) + fib(n-2);

}
```

Overlapping Sub problems

```
if (n==0 || n==1)
    return n;
return fib(n-1) + fib(n-2);
```



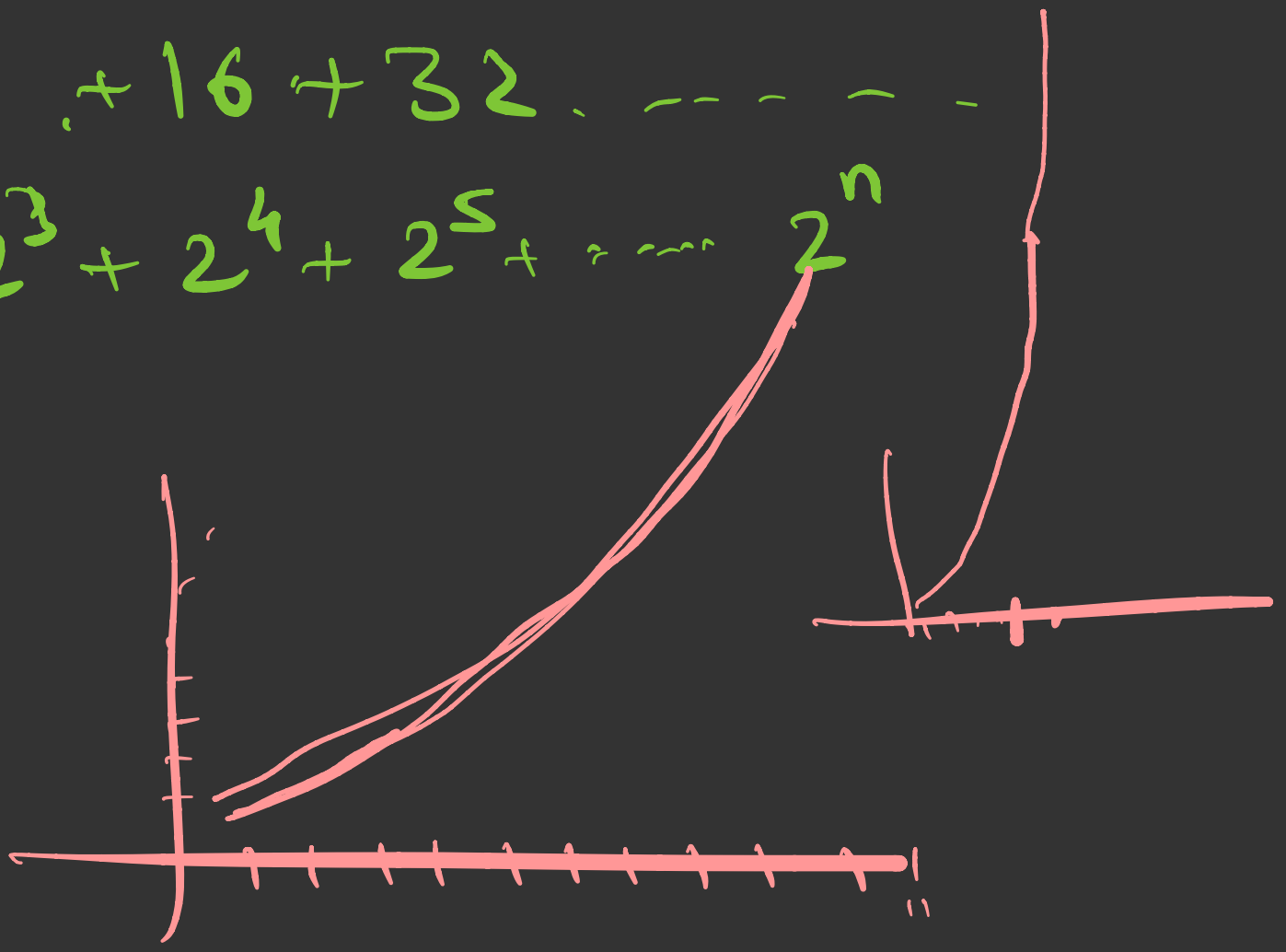
Exponential time complexity

$2 + 4 + 8 + 16 + 32 \ldots\ldots$

$f(n) = 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + \ldots\ldots 2^n$

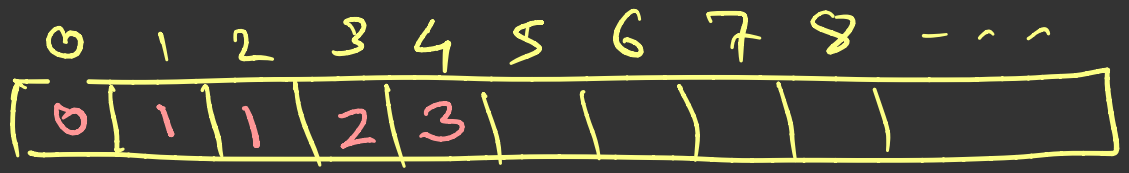
$T = O(2^n)$

n	2^n
1	2
2	4
5	32
10	1024



Solution using dp

```
int dp[20];
for(i=0; i<=19; i++)
    dp[i]=-1;
```



n=6

fib(5) + fib(4)

3 fib(4) + fib(3)

2 fib(3) + fib(2)

1 fib(2) + fib(1)

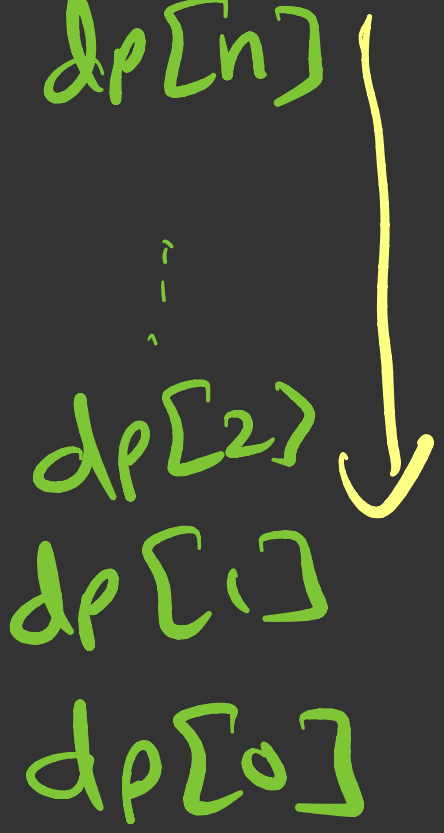
1 fib(1) + fib(0) 0

```
int fib(int n)
{
  if( dp[n] != -1)
      return dp[n];

  if(n==0 || n==1)
     dp[n] = n;
  else
    dp[n]= fib(n-1) + fib(n-2);

  return dp[n];

}
```

DP solution means

Recursion + Memoization

Top down approach



Memoization

It is an optimization technique that makes applications more efficient.

It does this by storing computation results in memory and retrieving that same information next time needed instead of recomputing.

Top down approach follows the memoization technique.

Any state can be computed by recent state results, this goes recussively till the base state if required.

Tabulation

Tabulation method is a bottom up approach

We start our transition from our base state and follow our state transition relation to reach our destination state, we call it Bottom up approach.

Solution through tabulation

```
int dp[size];
dp[0] = 0;
dp[1] = 1;
```

{

```
for(i=2; i<=n; i++)
    dp[i] = dp[i-1] + dp[i-2];
```

}