03/06/2023



	Dyanamic Programming
	It says that leavn from the bast. It also says
	that if we have calculated a particular thing,
	then store it so that we don't have to calculate
	it again and again.
	Phases in DP approach
	Top-down approach = Recursion + memorgation
2)	Bottom-up approach = Tabulation method
3)	Space optimization.
	There are 4 phases. Oth phase is apply the
	recursion a mon ulgar es deson em enem of
- 1	When to apply DP?
1)	Overlapping subproblems - Calculating same
	problem again and again.
2)	Optimal substructure - Optimal solution of bigger
	problem can be calculated using optimal solution.
	of smaller problems.
	The state of the s
	Fibonacci number
	fib(n) = fib(n-1) + fib(n-2) → Recursive Relation
	Fib(1) = 1 ? Base case
	$fib(0) = 0$ $\int calculating$
	again
	JIB(S)
	fib(4) $(fib(3))$
	318(4)
	calculated (fib(3)) (fib(2)) (fib(2)) fib(1)
	Calculated 2
	(fib(2)) fib(1) fib(1) fib(0) fib(1) fib(0)
	fib(1) fib(0)
2	fib(1) fib(0)

	OLEPS	MARIA	mortalist		
1)	Forget about DP	and	simply	write	the
	recursive code.	to tu	In sword	914 F	

$$if(n = = 1 | 1| n = = 0)$$

return nj int ans = fib(n-1) + fib(n-2); return ans;

3

- Row we need to apply memoization.

 Memoization means storing the answer.

 For doing memoization, we need to follow

 3 steps
- (i) We need to create the db array. Here we are going from n to & hence we need to create n+1 size array.

vector <int>dp(n+1,-1);

5 initialize with

(ii) Now where we are calculating ans, replace the ans with dp[n].

dp[n] = fib(n-1) + fib(n-2); retwn dp[n];

(111) Now just after the base case, check whether value is already calculated

Scarined with cam

	or not.			
	if (dp[n] =-1)			
return db[n];				
,	delawadat dudtamalandelmasagat : [a]al / ,			
	Dry run			
	1 2 3 5 8			
	dp-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -			
	-			
	7 f (6, db) fib(4, db)			
	((5, dp) (7, ap)			
	fib(3)dp)			
	7 f (4, dp)			
	fib(2,dp)			
1 in	7f(3,dp) = ±			
	and 1 / de la serie de la lange de la lang			
	7 f(2,db) \ f(,db)			
	1 (1 / 1) Photo and a second in the second			
	f(1,dp) f(0,dp)			
	<u></u>			
Mote	19t is called top-down as we are going from			
	n to o.			
	/> (a fai) swazgo madoku taki .			
	0 1 (Base case)			
	Code : (1 d'un de rant à raine)			
	2 (11)			
The same	int top Down Solve (int no vector < int> & dp) {			
	// Base case			
	if(n = = 1 1 n = = 0)			
	return no			
	// Step 3: Check if already calculated			

Scarnieu with Carn

```
if (dp(n) != -1)
                                                                       return de [n]
                              // Step 2: Replace and with dp[n]
                             dp[n] = topDownSolve (n-1,dp) + topDownSolve
                                                                                                                                                                                            (n-2,db);
                         return db[n];
                   int fib (int n) {
                                  //Step 1: Create db avuay
                                  vector (int> dp(n+1,-1);
                                  int ans = top Down Solve ( n,db);
                                 return ans;
 TC = O(n)

SC = O(n+n)

Bottom - up approach recursive stack > dp averay

1) Create dp armay (Not same as top-down always)

2) Observe have as consistent in the constant of t
2) Observe base case in top down solution.
3) Reverse the flow which was in top-down
            approach. (All game lies here)
         Code
            int Bottom Up Solve (int n) {
                      //Step 1: (reate de avray vector <int>de (n+1, -1);
                           //Stepa: Observe base case
                         dp[0]=01
                         if (n = = 0)
                                             return de [O]
                      dp[1]=
                       if (n = = 1)
                                                               return dp[1];
```

Scarnieu with Cam

```
//Step 3: Reverse flow
       for (int i=2; i <=n; i++) {
         dp [i] = dp [i-1] + dp [i-2] i
      return dp [n]
      int fib (int n) {
          int ans = Bottom Up Solve (n) j
          return ans i
                                 TC = 0(n)
                                 SC = O(n)
                                       4 db array
    Dry run
    dp[0]=0;
    dþ[1] = 1)
    i = 2
    dp[2] = dp[0] +dp[1] = 0+1=1
    1 = 3
    db[3] = db[2] +db[1] = 1+1=2
    db(4) = db(3) + db(2) = 2+1=3
    1 = 4
Note- Sometimes like in above question bottom-up
    approach has better space complexity & in
    questions where constraints are restricted, top
    down approach can give evor like TLE or
    memory limit exceeded.
```

Scarineu with Carn

	Space optimization
	We see that ability is dependent on apility
	and db [i-2] only and not on the entire
	array. Hence instead of db away we can
	take variables and then move these variables
	accordingly.
	i (a ini an tii
	Code (n) some quantity and a
	S ZIM A SHARK
	int space Opt Solve (int n) {
	1/ Take variables instead of db array
-1,	int breva = 0;
	int preV = 15
	if(n = = 0)
	retuin prev2;
	1f(n=1)
	retwin previs
	Int curri
	for (int i=2; i <= n; i++) {
	Curr = prev1 + prev2;
	prev2 = prev1j
	2 Previ = cury;
	retwin cwer i
	4
	E=1+×= (5)0/1=+ 18 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
	Dru run
41	Sur and the sure of the sure o
1	O pritalphas upper mother of a land
7	The second second
	preva previ
*	l=2

Scarineu wiiii Gaill

	cwu = 0+1=1
1	preva =
4	prevI = 1
- 41 - 41	0 1 1
	\uparrow \uparrow
	preva previ
*	1 = 3
1	cwr = 1+1=2
8 2	preva = 1
1	prevI = 2
- SA	0 1 1 2
	\mathcal{L}
1	prev previ
*	1° = 4
-	Cury = 2+1=3
	preva = 2
	brev1 = 3
	0 1 2 3 Ans of fib(4).
	OD CINER
	We have to yeturn cour as answer.
W.	C 11 coll changing - ID DP
Note-	I parameter in function call changing - ID DP
	2 parameter in function call changing - 2D DP