

Dynamic Programming | Set 1 (Overlapping Subproblems Property)

Dynamic Programming is an algorithmic paradigm that solves a given complex problem by breaking it into subproblems and stores the results of subproblems to avoid computing the same results again. Following are the two main properties of a problem that suggest that the given problem can be solved using Dynamic programming.

- 1) Overlapping Subproblems
- 2) Optimal Substructure

1) Overlapping Subproblems:

Like Divide and Conquer, Dynamic Programming combines solutions to sub-problems. Dynamic Programming is mainly used when solutions of same subproblems are needed again and again. In dynamic programming, computed solutions to subproblems are stored in a table so that these don't have to be recomputed. So Dynamic Programming is not useful when there are no common (overlapping) subproblems because there is no point storing the solutions if they are not needed again. For example, [Binary Search](#) doesn't have common subproblems. If we take example of following recursive program for Fibonacci Numbers, there are many subproblems which are solved again and again.

```
/* simple recursive program for Fibonacci numbers */
int fib(int n)
{
    if ( n <= 1 )
        return n;
    return fib(n-1) + fib(n-2);
}
```

Recursion tree for execution of *fib(5)*

Google™ Custom Search



GeeksforGeeks



53,524 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

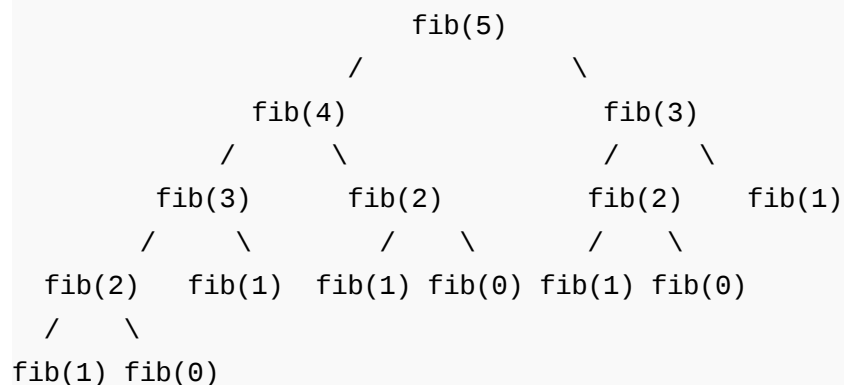
[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)



We can see that the function `f(3)` is being called 2 times. If we would have stored the value of `f(3)`, then instead of computing it again, we would have reused the old stored value. There are following two different ways to store the values so that these values can be reused.

a) *Memoization (Top Down):*

b) *Tabulation (Bottom Up):*

a) *Memoization (Top Down):* The memoized program for a problem is similar to the recursive version with a small modification that it looks into a lookup table before computing solutions. We initialize a lookup array with all initial values as NIL. Whenever we need solution to a subproblem, we first look into the lookup table. If the precomputed value is there then we return that value, otherwise we calculate the value and put the result in lookup table so that it can be reused later.

Following is the memoized version for nth Fibonacci Number.

```

/* Memoized version for nth Fibonacci number */
#include<stdio.h>
#define NIL -1
#define MAX 100

int lookup[MAX];

/* Function to initialize NIL values in lookup table */
void _initialize()
{
    int i;
    for (i = 0; i < MAX; i++)
        lookup[i] = NIL;
}

```



Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

```

}

/* function for nth Fibonacci number */
int fib(int n)
{
    if(lookup[n] == NIL)
    {
        if ( n <= 1 )
            lookup[n] = n;
        else
            lookup[n] = fib(n-1) + fib(n-2);
    }

    return lookup[n];
}

int main ()
{
    int n = 40;
    _initialize();
    printf("Fibonacci number is %d ", fib(n));
    getchar();
    return 0;
}

```

b) Tabulation (Bottom Up): The tabulated program for a given problem builds a table in bottom up fashion and returns the last entry from table.

```

/* tabulated version */
#include<stdio.h>
int fib(int n)
{
    int f[n+1];
    int i;
    f[0] = 0;    f[1] = 1;
    for (i = 2; i <= n; i++)
        f[i] = f[i-1] + f[i-2];

    return f[n];
}

int main ()
{
    int n = 9;
    printf("Fibonacci number is %d ", fib(n));
    getchar();
    return 0;
}

```



}

Both tabulated and Memoized store the solutions of subproblems. In Memoized version, table is filled on demand while in tabulated version, starting from the first entry, all entries are filled one by one. Unlike the tabulated version, all entries of the lookup table are not necessarily filled in memoized version. For example, memoized solution of [LCS problem](#) doesn't necessarily fill all entries.

To see the optimization achieved by memoized and tabulated versions over the basic recursive version, see the time taken by following runs for 40th Fibonacci number.

[Simple recursive program](#)

[Memoized version](#)

[tabulated version](#)

Also see method 2 of [Ugly Number post](#) for one more simple example where we have overlapping subproblems and we store the results of subproblems.

We will be covering Optimal Substructure Property and some more example problems in future posts on Dynamic Programming.

Try following questions as an exercise of this post.

- 1) Write a memoized version for LCS problem. Note that the tabular version is given in the CLRS book.
- 2) How would you choose between Memoization and Tabulation?

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

References:

<http://www.cs.uiuc.edu/class/fa08/cs573/lectures/05-dynprog.pdf>

<http://web.iit.ac.in/~avidullu/pdfs/dynprg/Dynamic%20Programming%20Lesson.pdf>

<http://www.youtube.com/watch?v=V5hZoJ6uK-s>

Recent Comments

Abhi You live US or India?

[Google \(Mountain View\) interview](#) · 1 minute ago

[Aman](#) Hi, Why arent we checking for conditions...

[Write a C program to Delete a Tree](#) · 40 minutes ago

[kzs](#) please provide solution for the problem...

[Backtracking | Set 2 \(Rat in a Maze\)](#) · 44 minutes ago

[Sanjay Agarwal](#) bool

[tree::Root_to_leaf_path_given_sum\(tree...](#)


[Root to leaf path sum equal to a given number](#) · 1 hour ago

[GOPI GOPINATH @admin](#) Highlight this sentence "We can easily...

[Count trailing zeroes in factorial of a number](#) · 1 hour ago

[newCoder3006](#) If the array contains negative numbers also. We...

[Find subarray with given sum](#) · 1 hour ago

AdChoices 

Free Downloadable Videos

 worldslastchance.com


You Must Own The Most Viewed
End-Time Bible Prophecy Videos
Now



[▶ Computer Science](#)

[▶ Fibonacci](#)


[▶ Algorithm](#)

AdChoices 

[▶ Recursion](#)

[▶ C++ Source Code](#)

[▶ Function](#)

AdChoices 

[▶ Function One](#)

[▶ C++](#)

[▶ Complex Numbers](#)

Related Topics:

- [Backtracking | Set 8 \(Solving Cryptarithmic Puzzles\)](#)
- [Tail Recursion](#)
- [Find if two rectangles overlap](#)
- [Analysis of Algorithm | Set 4 \(Solving Recurrences\)](#)
- [Print all possible paths from top left to bottom right of a mXn matrix](#)
- [Generate all unique partitions of an integer](#)
- [Russian Peasant Multiplication](#)
- [Closest Pair of Points | O\(nlogn\) Implementation](#)



10



2



4

Writing code in comment? Please use ideone.com and share the link here.

33 Comments [GeeksforGeeks](#)

Sort by Newest ▼



Join the discussion



with the recursion...



ankit · a month ago

@GFG

which approach is better bottom-up or top-down?

^ | v ·



Gaurav Kapoor · 4 months ago

With Normal recursion OUTPUT for n=30 : time taken to calculate the fibonacci is 832040

With Memorization OUTPUT for n=30 : time taken to calculate the fibonacci of 832040

Why its taking less time in Recursion ?? It should be the reverse way

8 ^ | v ·



avidullu · 4 months ago

@GeeksforGeeks Can you please remove the link to web.iiit.ac.in/avidullu Tha

1 ^ | v ·



The Teacher · 5 months ago

first Two links are giving 404 ERROR !!!!

1 ^ | v ·



anjali · 7 months ago

please give examples for top-down and bottom-up?

^ | v ·



myth17 · 8 months ago

2 of the Reference links are hitting 404.

5 ^ | v ·



Ramprasad Meena · 9 months ago

#include

```
int unsigned long fc(int n)
{
int i;int unsigned long previousFib = 0, currentFib = 1,newFib;
if( n == 0)
return 0;
else for(i=1;i<n;i++) // loop is skipped if n=1
{
newFib= previousFib + currentFib;
previousFib = currentFib;
currentFib = newFib;

}
return currentFib ;
}
```

```
int main()
{
int val;
printf("\nEnter number you want to generate fb :");
scanf("%d",&val);
printf("new bottum to down :%lu",fc(val));
return 0;
}
```

^ | v ·



Mitch Mitchell · a year ago

Thanks Guys! This is an awesome tutorial!

^ | v ·



Gurpriya Singh · a year ago

thank you sir!

^ | v ·



anon · a year ago

can any DP problem be solved by both top-down and bottom-up approach??

^ | v ·



geekguy → anon · 11 months ago

Almost every problem can be solved by both top-down and bottom-up :

```
/* Paste your code here (You may delete these lines if not write your code here) */
```

^ | v ·



Ejike Usulor · a year ago

This program is the bomb.

^ | v ·



mafia · a year ago

```
/* /* function for nth Fibonacci number */
int fib(int n)
{
    if(lookup[n] == NIL)
    {
        if ( n <= 1 )
            lookup[n] = n; //if(n==0) then lookup[n]=n; wrong??
        else
            lookup[n] = fib(n-1) + fib(n-2);
    }

    return lookup[n];
}
```



```
}*/
```

here if (n==0){lookup[n]==1;}

^ | v .



shiwakant.bharti → mafia · 9 months ago

@mafia

1.> Please consider the series as 0 1 1 2 3 as rightly said by @Himani
2.> If you just write the code to handle n==0 then it will break at n=' and n=-1. This will end up in negative index of lookup. In Java you might get ArrayIndexOutOfBoundsException, in C/C++ (depends on versions) it might cause segmentation fault/stackoverflow or something.

^ | v .



Himani → mafia · a year ago

/* /* function for nth Fibonacci number */

int fib(int n)

{

if(lookup[n] == NIL)

{

if (n <= 1)

lookup[n] = n; /*right if series goes on as 0 1 1 2 3 5 .. */

else

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

}

return lookup[n];

*/

^ | v .



Pulkit · 2 years ago

Awesome Post..!!

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | v ·



Neol · 2 years ago

Which one is more efficient to store solutions of subproblems ?

In my view, Memoization will be efficient as we don't have to fill complete lookup all entries . Correct me if I am wrong.

^ | v ·



jain.ankit → Neol · 10 months ago

Its just in memoization stack size would be $O(n)$ as the function will re-fill the solution for subproblems.

```
/* Paste your code here (You may delete these lines if not wri
```

1 ^ | v ·



shrikant · 3 years ago

will changing the fib code from

`lookup[n] = fib(n-1) + fib(n-2)`

to

`lookup[n] = fib(n-2) + fib(n-1)`

not make it better ?

^ | v ·



Praveen → shrikant · 3 years ago

I think its better to write

`lookup[n] = fib(n-2) + fib(n-1)`

than

$\text{lookup}[n] = \text{fib}(n-1) + \text{fib}(n-2)$

nice comment..

^ | v .



Praveen → Praveen · 3 years ago

srory for above...

It doesn't matter about the order ...

since we each time look up for the table.

^ | v .



Anand → shrikant · 3 years ago

<http://anandtechblog.blogspot....>

^ | v .



Anand · 3 years ago

<http://anandtechblog.blogspot.com/2011/01/adobe-question-cuboidal-boxes.ht>

^ | v .



Anand · 3 years ago

<http://anandtechblog.blogspot.com/2011/05/arra-y-problem.html>

^ | v .



shubh · 3 years ago

You have written Memoization as Top Down, but standard books say that Dyn
Please explain.

^ | v .



GeeksforGeeks → shubh · 3 years ago



@shubh: CLRS book considers Memoization as a variation of Dynamic Programming while maintaining a top down strategy.

^ | v .



Anand · 3 years ago

Here is blog that has all solved DP problem frequently asked in interviews.

<http://anandtechblog.blogspot.com/2011/01/amazon-question-dynamic-programming>

^ | v .



satya · 3 years ago

@geeksforgeeks..Great Post & Gud Work Keep Posting All Standard DP Problems

Satya

^ | v .



SDiZ · 3 years ago

I just realize how flawed the wikipedia page is.

Traditional source (e.g. NIST DADS) say

Definition: Solve an optimization problem by caching subproblem solutions and recomputing them.

^ | v .



kartik → SDiZ · 3 years ago

@SDiZ: The Wikipedia page talks about general definition of Memoization. Memoization is not limited to algorithms. NIST DADS talks about algorithm specific definition.

<http://www.algorithmist.com/index.php/Memoization>

^ | v .



sid → kartik · 2 years ago

@GeeksForGeeks, Can You post the shortest path algo like be path etc .

```
/* Paste your code here (You may delete these lines if
```

^ | v ·



SDiZ · 3 years ago

.. is an algorithmic paradigm that solves a given complex problem by breaki

This is called "divide and conquer".

Dynamic Programming is about memorization.

^ | v ·



GeeksforGeeks → SDiZ · 3 years ago

@SDiZ: Both Dynamic Programming(DP) and Divide and Conquer(DC) into subproblems. The key difference is: In DP, results of subproblems might be needed later. In DC, results of subproblems are not stored. We store results of subproblems.

^ | v ·



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team