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 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)*

```
                         fib(5)
                     /             \
               fib(4)                fib(3)
             /      \                /     \
         fib(3)      fib(2)         fib(2)    fib(1)
        /     \        /    \      /    \
  fib(2)   fib(1)  fib(1) fib(0) fib(1) fib(0)
  /    \
fib(1) fib(0)
```

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;
}

/* 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.youtube.com/watch?v=V5hZoJ6uK-s

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

Vibhor Garg • 7 days ago

In the memoization part, there should be a precondition to check that n is greater than equal to one, only then the loop should run.

• Reply • Share ›

Anil Bhaskar • a month ago

The tabulation method is a simple iterative way of computing Fibonacci, isn't it?

• Reply • Share ›

Navroze • 2 months ago

The answer to the second question is that we can use memoization in order to store the answer of some complex calculation and the tabulated version can be used in case we know an arithmetic progression.

• Reply • Share ›

Jeremy Shi • 2 months ago

There is another O(lgn) solution .

• Reply • Share ›

vaibhav • 3 months ago

how is this code handling negative numbers like if i give n = -40 the output still comes out to be 0 ,why can someone please explain ?

• Reply • Share ›

Manraj Singh vaibhav • 3 months ago

How can you find fibonacci number -40th?

• Reply • Share ›

vaibhav Manraj Singh • 3 months ago

if (n <= 1)

lookup[n] = n;

this means that if i give n =-20 it will return lookup[-20]=-20 right?but my qustion is lookup[-20] aint possible only yet the output on n=-20 is 0

1 • Reply • Share ›

Sriram Ganesh vaibhav • 16 days ago

I understsand ur question. Its not relevant to fibonacci.. still good qn..

• Reply • Share ›

Manraj Singh vaibhav • 2 months ago

It will give Segment fault!

• Reply • Share ›

SHIVAM DIXIT • 7 months ago

Can you please provide a post on space optimised knapsack?? In that we dont have to take such a large 2-d array....Its sometimes not possible to take array of size arr[n+1][capacity+1] for dp...example for such a problem is http://www.spoj.com/problems/L...

• Reply • Share ›

Goky SHIVAM DIXIT • 4 months ago

What we can do is we take array as arr[2][capacity+1].If you notice carefully we always require the results of row just one above the current row.So instead of using memory for all n items.We only consider the result of current and prev row only.So after computing current row,we will copy the content of current row to prev. row.We will continue to work this way till N items.If you have any question,please feel free to ask :)

1 • Reply • Share ›

Sivakumar K R SHIVAM DIXIT • 6 months ago

```
int knapSack1(int W, int *wt, int *val, int n)
{
int dp[W + 1];
memset(dp, -1e9, sizeof dp);
for (int i = 0; i < n; i++)
{
for (int j = W; j >= 0; j--)
{
int k = j + wt[i];
if (k <= W)
{
dp[k] = max(dp[k], dp[j] + val[i]);
}
}
cout<<'\n';
}
cout << '\n';
/*for (int j = 0; j <= W; j++)
cout << dp[j] << " ";*/
return dp[W];
}
```

• Reply • Share ›

lucky • 7 months ago

lookup must be of long type

1 • Reply • Share ›

arsh • 8 months ago

First two links are broken

http://www.cs.uiuc.edu/class/f...
http://web.iiit.ac.in/~avidull...

• Reply • Share ›

GeeksforGeeks Mod arsh • 8 months ago

Thanks for pointing this out. Looks like the pdf files have been removed from servers. We have updated the post.

• Reply • Share ›

quantized • 9 months ago

First two links in References are broken.

2 • Reply • Share ›

dg • a year ago

Please give comparison between top down approach(memorization) and bottom up approach. As in memorization(top down) we can avoid computation of some sub-problems(in some cases) on the other hand in bottom up approach we need to find solution of every sub problem. And how should we choose between Memoization and Tabulation?

1 • Reply • Share ›

ankit • a year ago

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

• Reply • Share ›

Anand Barnwal ankit • 4 months ago

Memoization(top down) usually becomes more complicated to implement but it has some advantages in problems, mainly those which you do not need to compute all the values for the whole matrix to reach the answer like: LCS.

Tabulation(bottom up) is easy to implement but it may compute unnecessary values sometimes. It also has benefits of less overhead.

1 • Reply • Share ›

rihansh Anand Barnwal • 4 months ago

Agree:D

• Reply • Share ›

Gaurav Kapoor • a year ago

With Normal recursion OUTPUT for n=30 : time taken to calculate the fibonacci of 30 is 11 microseconds and value is 832040

With Memorization OUTPUT for n=30 : time taken to calculate the fibonacci of 30 is 1647 microseconds and value is 832040

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

11 • Reply • Share ›

aa1992 Gaurav Kapoor • 7 months ago

i think because a lot of unnecessary computation is done in tabulation which is not needed hence increasing the space complexity.

• Reply • Share ›

coder Gaurav Kapoor • 10 months ago

because in recursion there are so many steps involved in calling the function like update stack register and return the value to calling function

• Reply • Share ›

Guest coder • 9 months ago

Gaurav was asking why Recursion is taking LESS time. Your answer is for why Recursion is taking MORE time.

4 • Reply • Share ›

avidullu • a year ago

@GeeksforGeeks Can you please remove the link to web.iiit.ac.in/avidullu That page has been removed.

2 • Reply • Share ›

The Teacher • a year ago

first Two links are giving 404 ERROR !!!!

1 • Reply • Share ›

anjali • 2 years ago

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

• Reply • Share ›

myth17 • 2 years ago

2 of the Reference links are hitting 404.

5 • Reply • Share ›

Ramprasad Meena • 2 years 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;
}
```

1 • Reply • Share ›

Mitch Mitchell • 2 years ago

Thanks Guys! This is an awesome tutorial!

2 • Reply • Share ›

Gurpriya Singh • 2 years ago

thank you sir!

• Reply • Share ›

anon • 2 years ago

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

• Reply • Share ›

geekguy anon • 2 years ago

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

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

1 • Reply • Share ›

Ejike Usulor • 2 years ago

This program is the bomb.

Reply • Share ›

mafia • 2 years 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;}

Reply • Share ›

shiwakant.bharti mafia • 2 years ago

@mafia
1.> Please consider the series as 0 1 1 2 3 as rightly said by @Himani.
2.> If you just the write the code to handle n==0 then it will break at n=1 as it will try to calculate it from n=0 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 iterate and finally lead to segmentation fault/stackoverflow or something.

Reply • Share ›

Himani mafia • 2 years 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];
}*/

1 • Reply • Share ›

Pulkit • 2 years ago

Awesome Post..!!

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

• Reply • Share ›

Neol • 3 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 table but in tabulation we have to fill all entries . Correct me if I am wrong.

• Reply • Share ›

jain.ankitk Neol • 2 years ago

Its just in memoization stack size would be O(n) as the function will recurse till base case before starting to fill the solution for subproblems.

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

1 • Reply • Share ›

shrikant • 4 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 ?

1 • Reply • Share ›

Praveen shrikant • 4 years ago

I think its better to write
lookup[n] = fib(n-2) + fib(n-1)

than

lookup[n] = fib(n-1) + fib(n-2)
nice comment..

• Reply • Share ›

Praveen Praveen • 4 years ago

srry for above...

It doesn't matter about the order ...

since we each time look up for the table.

2 • Reply • Share ›

Anand shrikant • 4 years ago

http://anandtechblog.blogspot....

• Reply • Share ›

Anand • 4 years ago

http://anandtechblog.blogspot.com/2011/01/adobe-question-cuboidal-boxes.html

• Reply • Share ›

Anand • 4 years ago

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

• Reply • Share ›

shubh • 4 years ago

You have written Memoization as Top Down, but standard books say that Dynamic Programming is Bottom Up. Please explain.

• Reply • Share ›

GeeksforGeeks shubh • 4 years ago

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

• Reply • Share ›

Anand • 4 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.html

• Reply • Share ›

satya • 4 years ago

@geeksfoegeeks..Great Post & Gud Work Keep Posting All Standerd DP Problems Please Keep it Up

Satya

• Reply • Share ›

Load more comments

-
-
-
-
    - [Interview Experiences]
    - [Advanced Data Structures]
    - [Dynamic Programming]
    - [Greedy Algorithms]
    - [Backtracking]
    - [Pattern Searching]
    - [Divide & Conquer]
    - [Mathematical Algorithms]
    - [Recursion]
    - [Geometric Algorithms]
-

- **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
- Follow @GeeksforGeeks

- ## Recent Comments

  - lt_k

    i need help for coding this function in java...

    Java Programming Language · 1 hour ago
  - Piyush

    What is the purpose of else if (recStack[*i])...

    Detect Cycle in a Directed Graph · 1 hour ago
  - Andy Toh

    My compile-time solution, which agrees with the...

    Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago
  - lucy

    because we first fill zero in first col and...

    Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago
  - lucy

    @GeeksforGeeks i don't n know what is this long...

    Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 2 hours ago
  - manish

    Because TAN is not a subsequence of RANT. ANT...

    Given two strings, find if first string is a subsequence of second · 2 hours ago