Tabulation vs Memoizatation

Prerequisite – Dynamic Programming, How to solve Dynamic Programming problems?

There are following two different ways to store the values so that the values of a problem can be reused. Here, will discuss two patterns of solving DP problem:

1. **Tabulation:** Bottom Up
2. **Memoization:** Top Down

Before getting to the definitions of the above two terms consider the below statements:

- **Version 1**: I will study the theory of Dynamic Programming from GeeksforGeeks, then I will practice some problems on classic DP and hence I will master Dynamic Programming.
- **Version 2**: To Master Dynamic Programming, I would have to practice Dynamic problems and to practice problems – Firstly, I would have to study some theory of Dynamic Programming from GeeksforGeeks

Both the above versions say the same thing, just the difference lies in the way of conveying the message and that's exactly what Bottom Up and Top Down DP do. Version 1 can be related to as Bottom Up DP and Version-2 can be related as Top Down Dp.

Tabulation Method – Bottom Up Dynamic Programming

As the name itself suggests starting from the bottom and cumulating answers to the top. Let's discuss in terms of state transition.

Let's describe a state for our DP problem to be dp[x] with dp[0] as base state and dp[n] as our destination state. So, we need to find the value of destination state i.e dp[n].

If we start our transition from our base state i.e dp[0] and follow our state transition relation to reach our destination state dp[n], we call it Bottom Up approach as it is quite clear that we started our transition from the bottom base state and reached the top most desired state.

Now, Why do we call it tabulation method?

To know this let’s first write some code to calculate the factorial of a number using bottom up approach. Once, again as our general procedure to solve a DP we first define a state. In this case, we define a state as dp[x], where dp[x] is to find the factorial of x.

Now, it is quite obvious that dp[x+1] = dp[x] * (x+1)

```
// Tabulated version to find factorial x.
int dp[MAXN];

// base case
int dp[0] = 1;
for (int i = 1; i< =n; i++)
{
    dp[i] = dp[i-1] * i;
}
```

The above code clearly follows the bottom-up approach as it starts its transition from the bottom-most base case dp[0] and reaches its destination state dp[n]. Here, we may notice that the dp table is being populated sequentially and we are directly accessing the calculated states from the table itself and hence, we call it tabulation method.

Memoization Method – Top Down Dynamic Programming

Once, again let’s describe it in terms of state transition. If we need to find the value for some state say dp[n] and instead of starting from the base state that i.e dp[0] we ask our answer from the states that can reach the destination state dp[n] following the state transition relation, then it is the top-down fashion of DP.

Here, we start our journey from the top most destination state and compute its answer by taking in count the values of states that can reach the destination state, till we reach the bottom most base state.

Once again, let’s write the code for the factorial problem in the top-down fashion

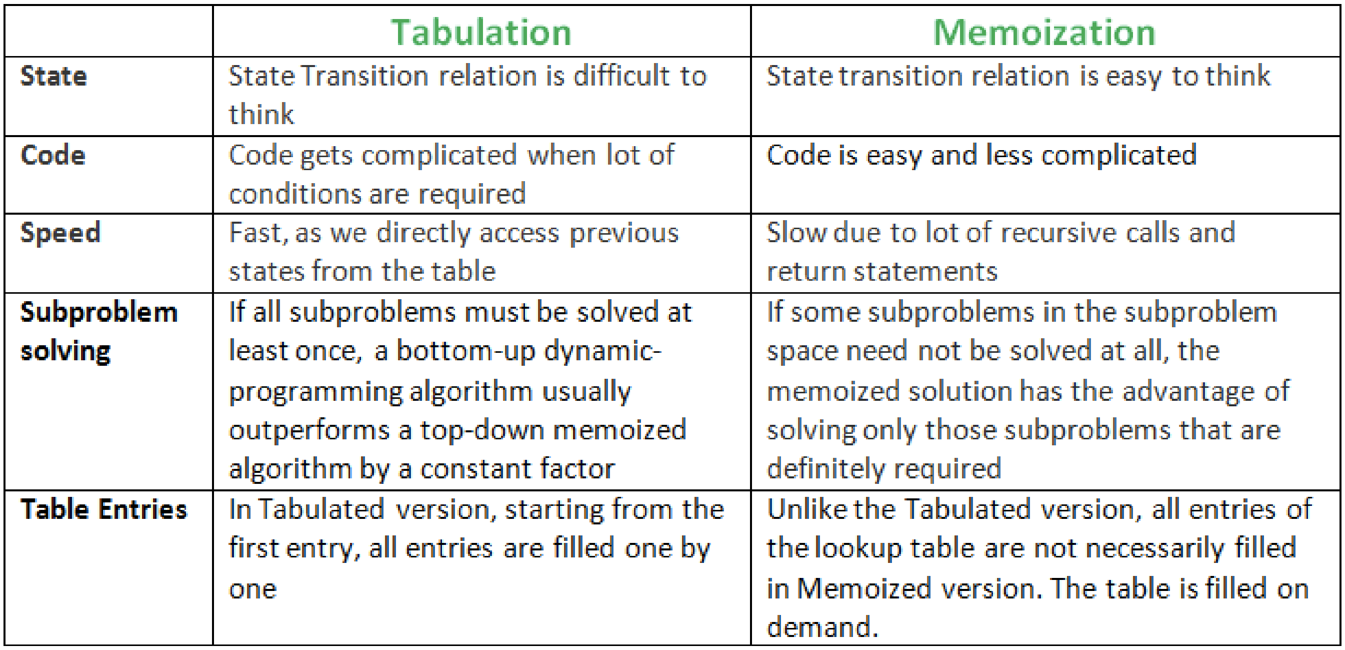
```
// Memoized version to find factorial x.
// To speed up we store the values
// of calculated states

// initialized to -1
int dp[MAXN]

// return fact x!
int solve(int x)
{
    if (x==0)
        return 1;
    if (dp[x]!=-1)
        return dp[x];
    return (dp[x] = x * solve(x-1));
}
```

As we can see we are storing the most recent cache up to a limit so that if next time we got a call from the same state we simply return it from the memory. So, this is why we call it memoization as we are storing the most recent state values.

In this case the memory layout is linear that's why it may seem that the memory is being filled in a sequential manner like the tabulation method, but you may consider any other top down DP having 2D memory layout like Min Cost Path, here the memory is not filled in a sequential manner.



	Tabulation	Memoization
State	State Transition relation is difficult to think	State transition relation is easy to think
Code	Code gets complicated when lot of conditions are required	Code is easy and less complicated
Speed	Fast, as we directly access previous states from the table	Slow due to lot of recursive calls and return statements
Subproblem solving	If all subproblems must be solved at least once, a bottom-up dynamic-programming algorithm usually outperforms a top-down memoized algorithm by a constant factor	If some subproblems in the subproblem space need not be solved at all, the memoized solution has the advantage of solving only those subproblems that are definitely required
Table Entries	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. The table is filled on demand.

This article is contributed by **Nitish Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

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

Practice Tags : Dynamic Programming

Article Tags : Dynamic Programming

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Bitmasking and Dynamic Programming | Set 1 (Count ways to assign unique cap to every person)
How to solve a Dynamic Programming Problem ?
Program for Fibonacci numbers
Tabulation vs Memoizatation
Weird Number
Semiperfect Number
Maximum sum in circular array such that no two elements are adjacent
Number of ways to reach Nth floor by taking at-most K leaps
Number of ways to represent a number as sum of k fibonacci numbers
Minimum number of single digit primes required whose sum is equal to N

(Login to Rate)

2.5 Average Difficulty : **2.5/5.0**
Based on **79** vote(s)

Add to TODO List

Mark as DONE

Basic Easy Medium Hard Expert

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Company-wise
Topic-wise
Contests
Subjective Questions

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos