CSE-300 Assignment-1
Introduction to LaTeX

An Introduction To Dynamic Programming Algorithms

Waqar Hassan Khan
Student ID : 1505107

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
(BUET)
Dhaka 1000
April 19, 2018

Contents

1 Introduction

In the world of computer programming techniques **Dynamic Programming** a.k.a *DP* perhaps is the most important one used in various fields of computer science. When i came by the name i thought it was something that have difficult mathematical equation like the equation-

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \infty$$

or like the equation $\frac{\partial^2 \psi}{\partial x^2} + \frac{8\pi^2 m}{\hbar^2}(E - V)\psi$

But it is really a different thing.The word was first used by **Bellman**. 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.

- Overlapping Subproblems
- Optimal Substructure

1.1 Overlapping Subproblems

The subproblems of a problem may overlap while solving using **Divide and Conquer**, that is we may end solving the same thing over and over again. In DP we keep track of the solved sub-problems and avoid solving them more than once. Once solved it is stored to use for farther use.

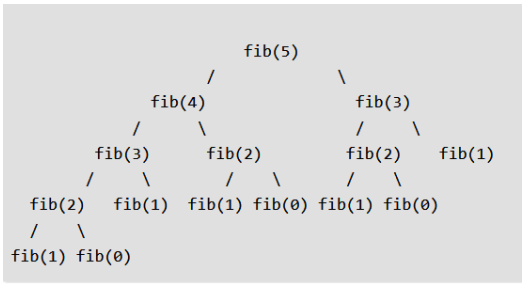

Figure 1: Overlapping Subproblems in determination of Fibonacci series

1.2 Optimal Sub-structure

A given problems has Optimal Substructure Property if optimal solution of the given problem can be obtained by using optimal solutions of its subproblems. For example, the Shortest Path problem has following optimal substructure property: If a node x lies in the shortest path from a source node u to destination node v then the shortest path from u to v is combination of shortest path from u to x and shortest path from x to v. The standard All Pair Shortest Path algorithms like Floyd–Warshall is an example of Dynamic Programming

2 Solving a DP

Steps to solve a DP:

1. Identify if it is a DP problem
2. Decide a state expression with least parameters
3. Formulate state relationship
4. Do tabulation (or add memoization)

Tabulation Bottom Up Dynamic Programming (starting from dp[0] we move up)

Memoization Top Down Dynamic Programming (starting from dp[n] we use recursion here)

3 Some Solved Problems

3.1 Fibonacci Numbers

The Fibonacci series is as follows 1 1 2 3 5 8 13 21... We have to write a program that can give us nth Fibonacci number. We can use both approach, first we show bottom-up then top-down. The codes are written in python-3 (Please see Figure 1)

3.1.1 top-down approach

```
#bottom-up
n=int(input())

fib=[None]*(n+1)

#base case
fib[1]=fib[2]=1
for i in range(3,n+1):
        fib[i]=fib[i-1]+fib[i-2]

print(fib[n])
```

3.1.2 bottom-up approach

```
def fibonacci(n):
        if fib[n]>0:
                return fib[n]

        if n==1 or n==2:
                return 1

        fib[n]=fibonacci(n-1)+fibonacci(n-2)
        return fib[n]

#top-down
n=int(input())
fib=[0]*(n+1)

print(fibonacci(n))
```

3.2 Calculating ${}^nC_r$

Calculating ${}^nC_r$ using dynamic programming is really easy. We have to use the recurrence relation ${}^nC_r = {}^{n-1}C_r + {}^{n-1}C_{r-1}$

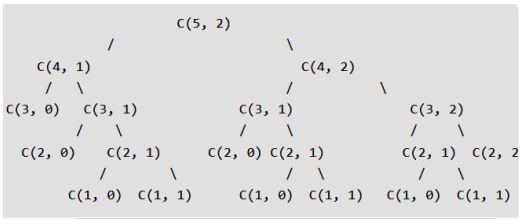

Figure 2: Overlapping Subproblems in determination of Fibonacci series

```
def nCr(n,r):
        if n==0:
                return 0

        if r==0 or r==n:
                return 1

        dp[n][r]=nCr(n-1,r)+nCr(n-1,r-1)
                return dp[n][r]

#top-down
n,r=map(int,input().split())
dp=[[0]*(r+1)]*(n+1)

print(nCr(n,r))
```

4 Conclusion

Dynamic Programming has many classic problems and other problems need intuition and practice to solve.