Dynamic Programming

– Saurabh Verma

Agenda

- What is DP?
- Memoizarion & Tabulation with Example
- 0/1 Knapsack Problem
- Subset Sum Problem
- Equal Sum Partition Problem
- Count of Subsets with Sum equal to X
- Unbounded Knapsack Problem
- Coin Change Problem 1
- Coin Change Problem 2
- Rod Cutting Problem
- Maximum Ribbon Cut Problem
- Longest Common Subsequences
- Longet Common Substring
- Minimum Insertion or Deletion to change A into B
- Minimum Insertion or Deletion to make Palindrome
- Longest Plindromic Subseuence

What is DP?

- Dynamic Programming (DP) is an algorithmic technique for solving an optimization problem by breaking it down into simpler subproblems and utilizing the fact that the optimal solution to the overall problem depends upon the optimal solution to its subproblems.

What is dp? (contd..)

- Dynamic Programming is mainly an optimization over plain recursion. Wherever we see a recursive solution that has repeated calls for same inputs, we can optimize it using Dynamic Programming.
- The idea is to simply store the results of subproblems, so that we do not have to re-compute them when needed later. This simple optimization reduces time complexities from exponential to polynomial.

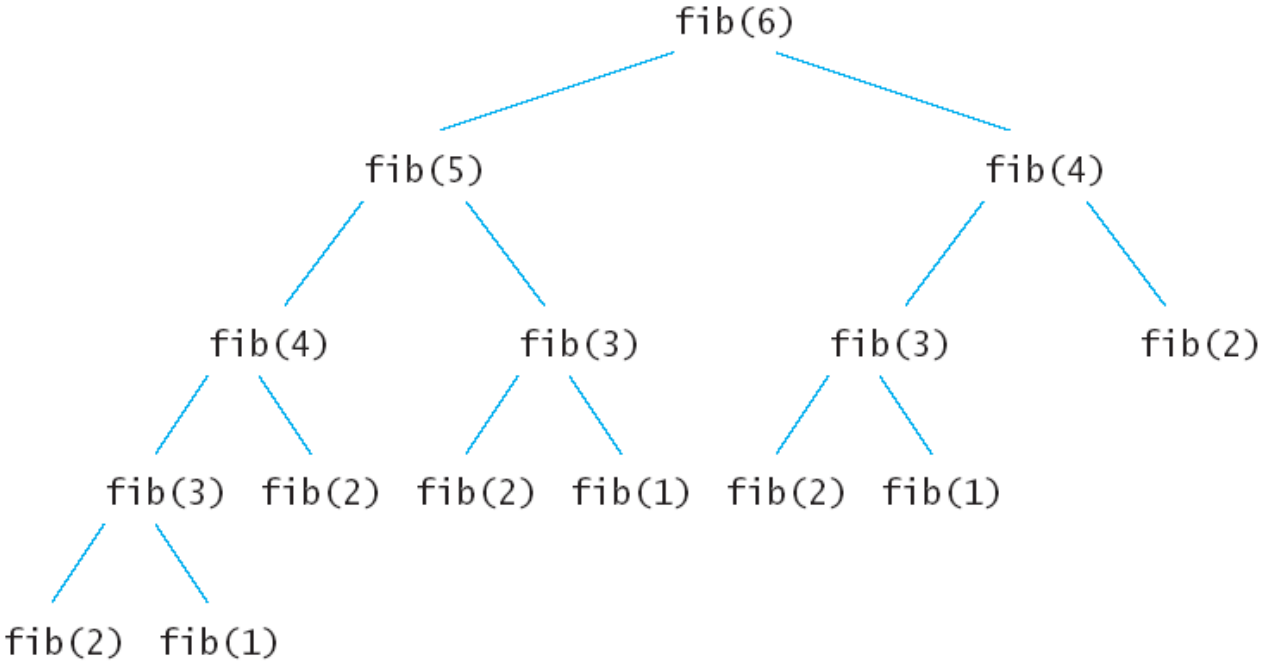
Example: N^{th} Fibonacci Number

- 0, 1, 1, 2, 3, 5, 8, 13, 21,..

```
int fib(int n){
    if(n<2)
        return n;
    return fib(n-1) + fib(n-2);
}
```

Time Complexity : $O(2^n)$

Example: Nth Fibonacci Number (Contd..)



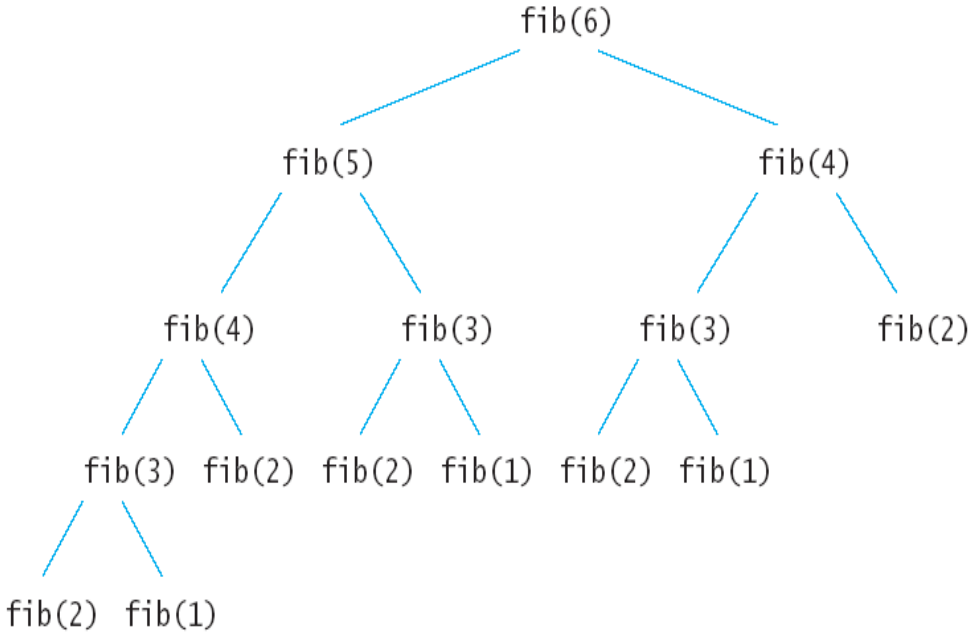
Example: Nth Fibonacci Number (Contd..)

```
static int fib(int n){
    if(n<2)
        return n;
    if(a[n] != -1)
        return a[n];
    a[n] = fib(n-1) + fib(n-2);
    return a[n];
}
```

Time Complexity : O(n)

Memoization Technique

Example: Nth Fibonacci Number (Contd..)



0	1	2	3	4	5	
0	1	-1	-1	-1	-1	-

Example: N^{th} Fibonacci Number (Contd..)

```java
static int fibTabulation(int n) {
    if (n==0) return 0;
    int dp[] = new int[n+1];

    //base cases
    dp[0] = 0;
    dp[1] = 1;

    for(int i=2; i<=n; i++)
      dp[i] = dp[i-1] + dp[i-2];

    return dp[n];
  }
```

Time Complexity : O(n)

Tabulation Technique

Memoization V/S Tabulation

```
static int fib(int n){
    if(n<2)
        return n;
    if(a[n] != -1)
        return a[n];
    a[n] = fib(n-1) + fib(n-2);
    return a[n];
}
```

Time Complexity : O(n)

```
static int fibTabulation(int n) {
    if (n==0) return 0;
    int dp[] = new int[n+1];

    //base cases
    dp[0] = 0;
    dp[1] = 1;

    for(int i=2; i<=n; i++)
      dp[i] = dp[i-1] + dp[i-2];

    return dp[n];
  }
```

01 Knapsack Problem

Problem Statement: Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Input: wt[], val[], n, W

Output: Max Profit

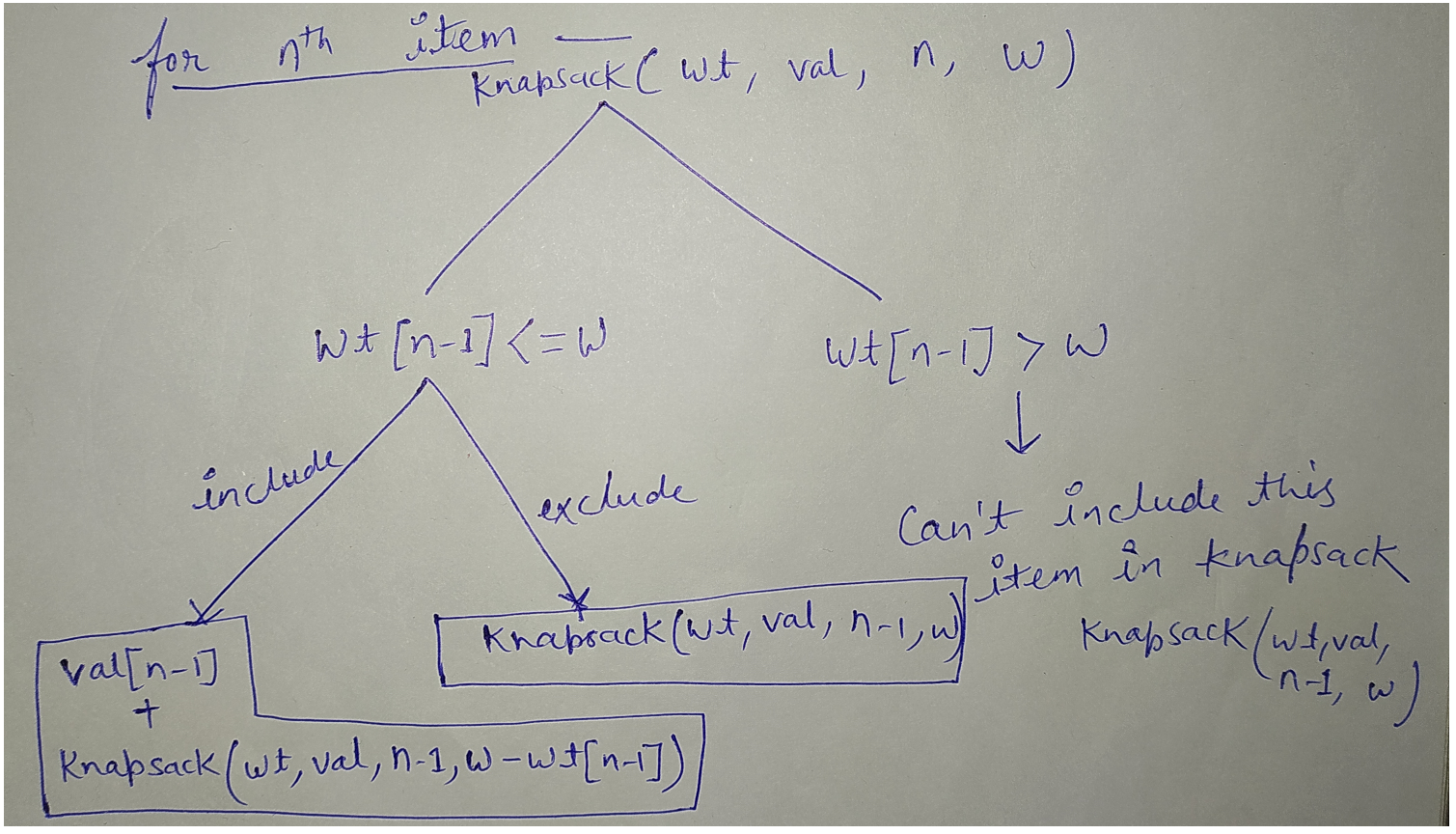
Example

value[] = {60, 100, 120};
weight[] = {10, 20, 30};
W = 50;

Solution: 220

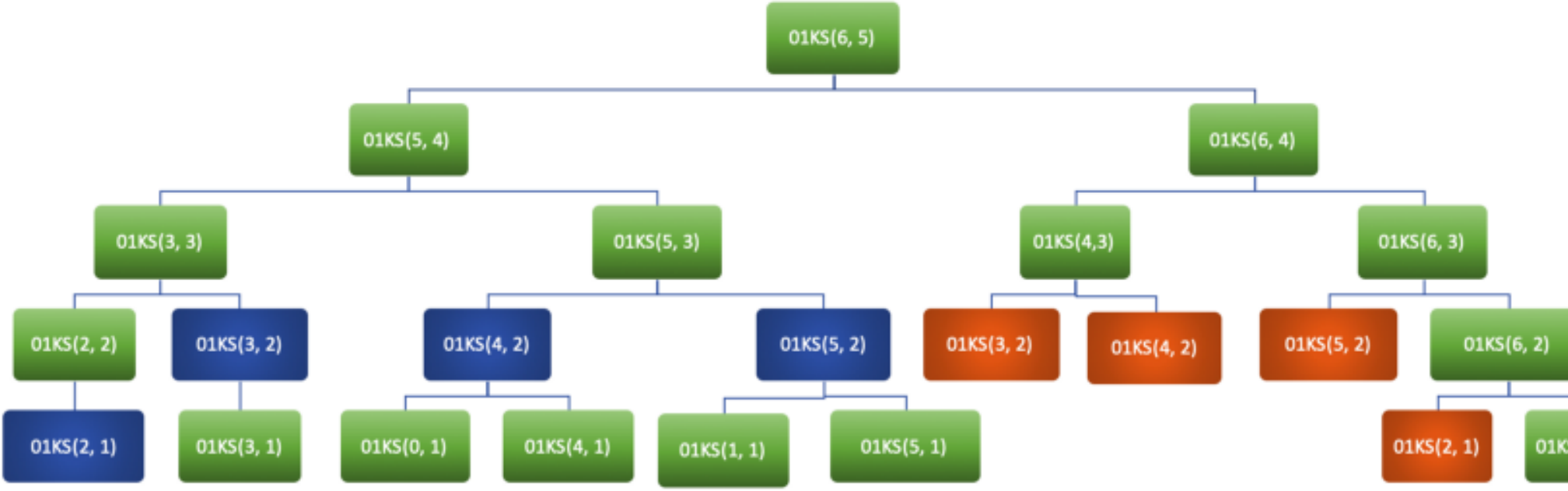
Weight = 10; Value = 60;
Weight = 20; Value = 100;
Weight = 30; Value = 120;
Weight = (20+10); Value = (100+60);
Weight = (30+10); Value = (120+60);
Weight = (30+20); Value = (120+100);
Weight = (30+20+10) > 50

Brute Force Approach(Recursive Sol^n)



Knapsack Capacity = 6

Objects	B1	B2	B3	B4	B5
Profit	25	125	15	68	19
Weight	2	4	1	2	1



2d Matrix of Memoization Analysis

Capacity of knapsack (w) : 10

	0	1	2	3
wt[]	1	3	4	5
val[]	1	4	5	7

	0	1	2	3	4	5	6	7	8	9	10
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	1	1	1	-1	1	1	1	-1	-1	1
2	-1	1	-1	-1	-1	5	5	-1	-1	-1	5
3	-1	-1	-1	-1	-1	6	-1	-1	-1	-1	10
4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	13

Understand Optimal Subproblems

Capacity of knapsack (w) : 10

	0	1	2	3
wt[]	1	3	4	5
val[]	1	4	5	7

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1
2	0	1	1	4	5	5	5	5	5	5	5
3	0	1	1	4	5	6	6	9	10	10	10
4	0	1	1	4	5	7	8	9	11	12	13

Subset Sum Problem

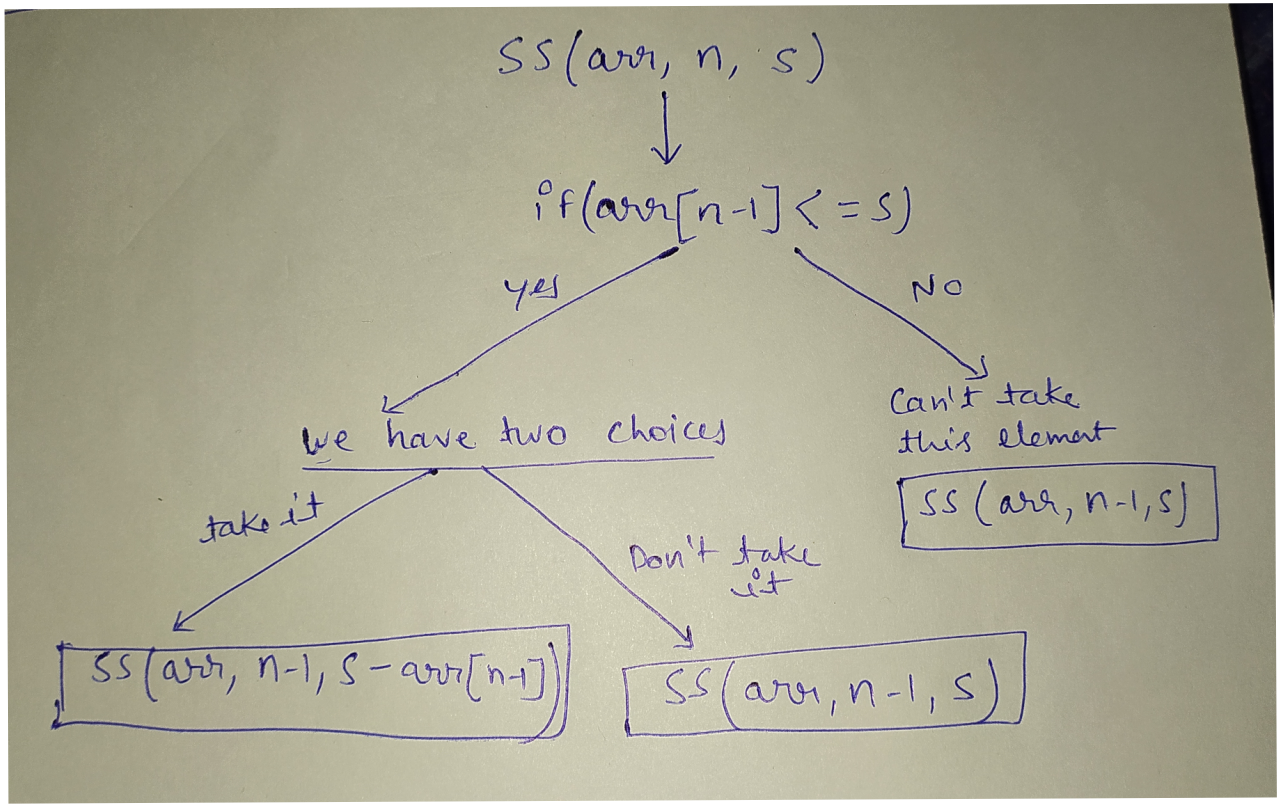
Given a set of non-negative integers, and a value *sum*, determine if there is a subset of the given set with sum equal to given *sum*.

Example:

```
Input: set[] = {3, 34, 4, 12, 5, 2}, sum = 9
Output: True
There is a subset (4, 5) with sum 9.

Input: set[] = {3, 34, 4, 12, 5, 2}, sum = 30
Output: False
There is no subset that add up to 30.
```

Choice Diagram



THANK YOU FOR JOINING!

See you tommorow in Part - 2

For any kind of Query, Feedback or Suggestions, you can contact me here-

Gmail- svworld01@gmail.com

WhatsApp- 7905846682

Linkedin - @svworld01

Instagram- @svworld01

Github - @svworld01

Facebook - @svworld01