- Is used if there is two recursive calls made

- If optimality is asked

- Recursion=>Memoization = DP

- Then buttonUp approach

1. 0-1 knapsack(6)

2. Unbounded knapsack(5)

3. Fibonacci(7)

4. LCS(15)

5. LIS(10)

6. Kodane's Algorithms(6)

7. Matrix Chain Multiplication(7)

8. DP on trees(4)

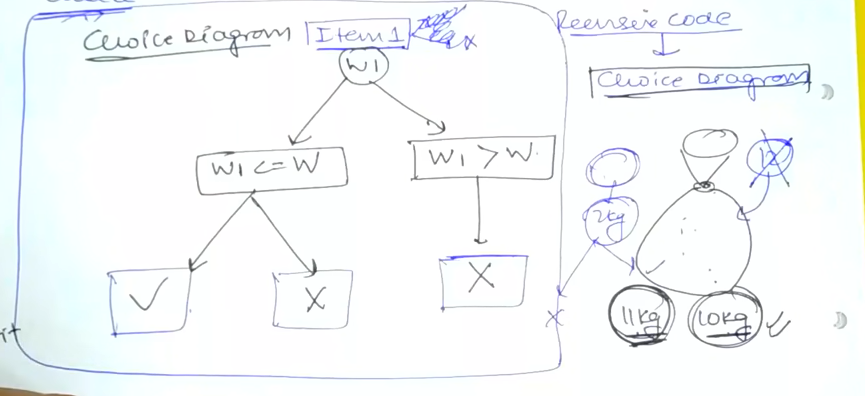
9. DP on Grid(14)

10. Others (5)

Identification - if we have choice

- if asked for optimal

Solution : Recursive ---> Memoization ---> BottomUp/TopDown

Choice Diagram: 

Look at return type, inputs :

Base Condition=> smallest valid input

Choice diagram

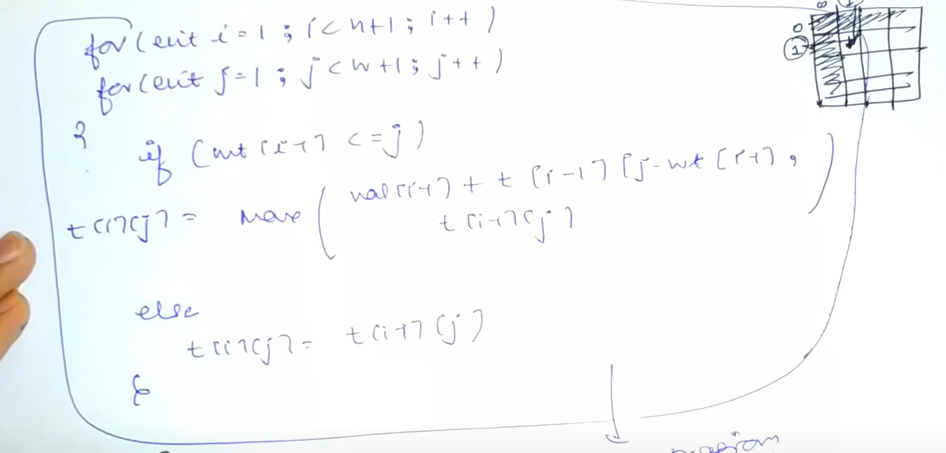
Now initialize matrix or dictionary. Size of matrix is size of changing variable.

Dictionary keys = changing variable

For Knapsack top-down:

1. Initialize =>base condition of base condition changes to initialization
2. Diagram

   Description automatically generated
3. Diagram

   Description automatically generated
4. 
5. i. Subset sum
6. Equal sum partition
7. Count of subset sum
8. Minimum subset sum
9. Target Sum
10. Number of subset

How to identify?

Unbounded Knapsack: allows multiple occurrences of same item

1. Route cutting
2. CoinChange
3. Coinchange 2
4. Maximum ribbon cut