**Resource**

Best Learning Resource →

1. <https://www.youtube.com/playlist?list=PL_z_8CaSLPWekqhdCPmFohncHwz8TY2Go> (**Aditya Verma**) (**Rahim**)
2. <https://www.youtube.com/playlist?list=PLgUwDviBIf0qUlt5H_kiKYaNSqJ81PMMY> (**Maruf**)
3. Recursion bujhar Best Trick → <https://youtu.be/J8mChdOJWOs?si=7JaAWsrWe59FWOkh> (Just Recursion for Beginner First Video ta)
4. Tutorial Link → <https://www.youtube.com/watch?v=oBt53YbR9Kk> (**Shoumik** Bhai)

**DP Step (Aditya Verma)**

DP er 3 ta Step

1. **Recursive Solution marbo**
2. **Recursive Solution** + **Memoization** (**CACHE** kore rakhbo)
3. at last optimize kore, **Top Down Approach**

**Cache = vector<vector<int>> cache**

**OR,**

**Cache = map<int, <vector<int>> cache**

**vector** or **map** akare amra **cache** rakhte pari

\*\*\*Recursive Function ee **2 ta f(n) Function Call** manei DP lagbe… mane **Overlapping Subproblems**… 1 ta f(n) call huile oita Overlapping Nah… mane oita DP nah… oita just Normal Backtracking or Recursion\*\*\*

Recursive Function ee DP lagbe kina taar upay bujhar upay → jodi (2 or Many Term) takhe mane, f(n-1), f(n-2) takhe, tokon ei bujbo… amra Recursive Function ke CACHE kore Solution peye jabo

Recursive bujhar Way:

1. **Choice** → Ask korbe jey 0/1 knapsack ee amra “object” ta ki Nibo naki nibo nah
2. **Optimal** ask korbe

## **Base Recursion Trick**

1. **Base Condition**
2. **Choice Diagram** Banao

For **Base Condition** → **Think of the SMALLEST Valid Input**,

Don’t think Backward like f(n-1), f(n-2)...f(0) → eivabe gele Mathai Base Condition ashbe Nah

**DP Step (Striver)**

\*\*\*Recursive Function ee **2 ta f(n) Function Call** manei DP lagbe… mane **Overlapping Subproblems**… 1 ta f(n) call huile oita Overlapping Nah… mane oita DP nah… oita just Normal Backtracking or Recursion\*\*\*

All **GRID problem er jei Recursion** kortesi sheita Actually → **DFS ei martesi**… but DFS er moto chinta Nah kore nicher Method ee Chinta koro Akash…

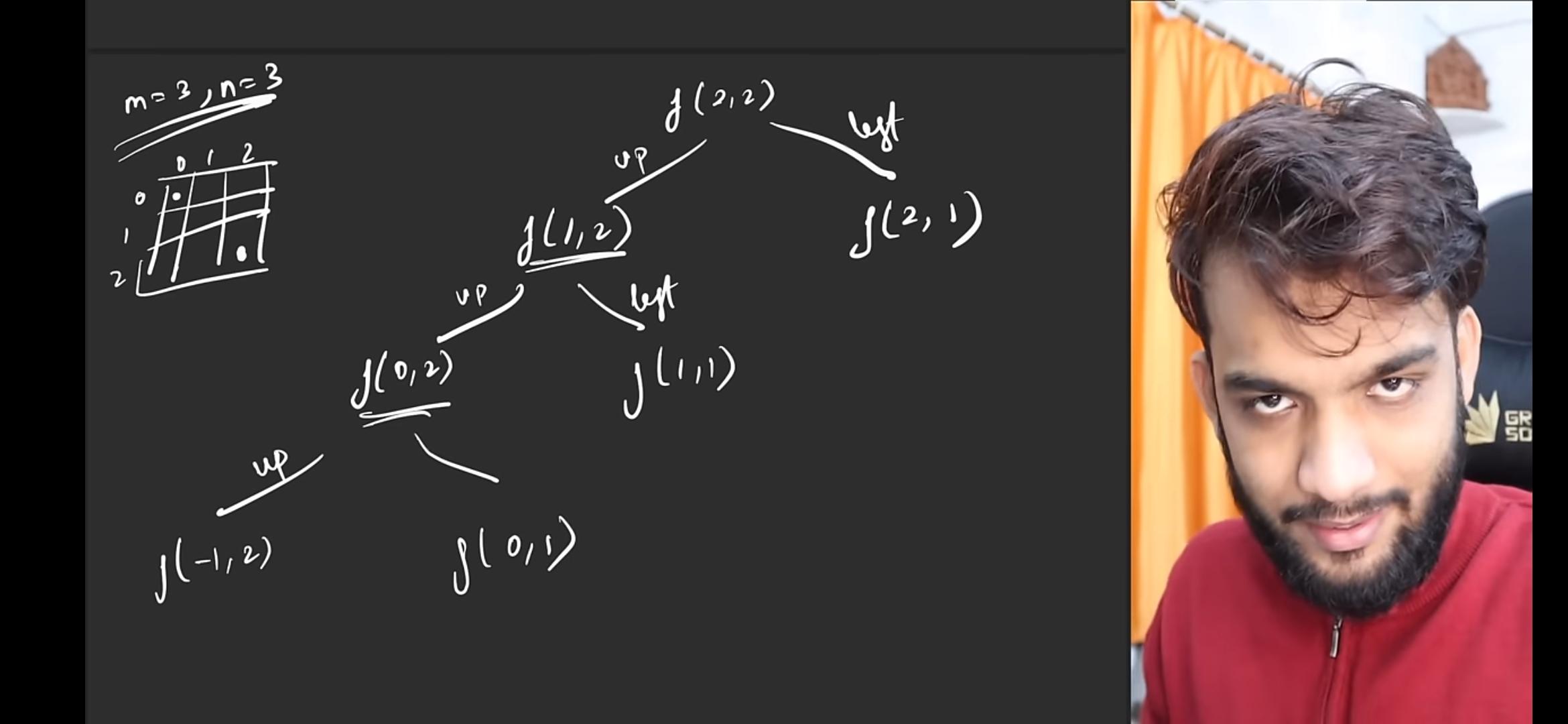
1. Recursion korar Rule hocche first ee… Problem ta ke “**index**” Function akare Create/Chinta korbo…

**“index” means the Parameters which will change (vary) during Traversing**…(like TargetSum ke Minus kore kore komale, sheita 1 ta parameter, abr Grid Traverse korar somoy i, j hocche index etc.)

For Example: f(i, j) → where (i,j) hocche Represent kore GRID er 1 Ghor…

1. Then Base Condition tik korbo → **Think of the SMALLEST Valid Input**
2. Recursion ke Memoization er jonno amra **CHOICE Diagram** Banai dekhbo jey kono Overlapping sub-problem ase naki

for Recursion **Choice Diagram** amra dekhbo how “**f(n)**” er sathe “**f(n-1), f(n-2), f(n-3)...**” **related** thakbe… eita diye arrow kore likbo…



## **When to apply Greedy (See “greedy doc” to have more information)**

We can only do a GREEDY Solution when there is **UNIFORMITY** in the Elements…

Array → 2, 4, 6, 8 → This is Uniform, because the GAP is EQUAL

Array → 2, 7, 8, 10 → This is NOT Uniform

## **Grid Problem**

1. Traverse from FIXED point to Up… usually **from (m-1, n-1) to (0, 0)**
2. Base Case → when to RETURN ZERO, when to RETURN VALUE for ADD
3. **“Take”** or **“Not Take”** → eivabe diye Logic Build Up korbo
4. **Unique Number of Paths** → Just Return 1 at the BASE Case… err majhe f(n) & f(n-1) er moddhe kono “**1+**” **LAGBE NAH**… just directly call f(n-1) from f(n)

## **Subsequence Problem**

### **N.B:**

**Subsequence** → **Contagious** & **Non-Contagious** BOTH

**Subarray** → Only Contagious

**Subset** → Any Possible Combination (including Empty Set) of the Given Sequence

1. **“Take”** or **“Not Take”** → eivabe diye Logic Build Up korbo
2. for “take” → Current Index (n’th) er Value er **PLUS korbo** (n-1th) DP Function er jonno

**array[n] + dp(array[n-1])**

**dp\_solve\_function(n) = array[n] + dp\_solve\_function(array[n-1])**

**Partition Equal Subset Sum** → Given an integer array nums, return true if you can partition the array into two subsets such that the sum of the elements in both subsets is equal or false otherwise:

S1+S2 = TotalSum, and S1=S2 (As equal Subset), so, S1+S1= TotalSum,

So, S1 = TotalSum/2;

So, find a Subset with a SUM = (TotalSum)/2

\*\*\***Divide an array into TWO subset sum**\*\*\*→ ei type er Question ee agge “S1+S2 = TotalSum” eirokom type **EXTRA Calculation** kore then amra then 1 ta “Target Value” pabo sheikan theke “**take/not take**” an Element from (n-1) to 0 concept mere Answer peye jabo

**Memoization to Tabulation**

1. Base case ke convert kore dp[0][0] eirokom Row & Column ee agge INITIALIZE korbo
2. f(i, j) ke amra “**Recursive**” er **OPPOSITE Direction** ee **FOR LOOP** likhbo…
   1. jodi Recursive ee **(n → 0)** **(Bottom Up)** ei Order ee Traverse kori taile FOR LOOP likhbo **(i=0;i<n;i++)**
   2. jodi Recursive ee (0 → n) ei Order ee Traverse kori taile FOR LOOP likhbo (i=n-1;i>=0;i--)
3. then Final dp[][] Return kore dibo

**Learning**

**Greedy is a One Kind of DP**

DP all possible cases & sub-problem gula repetitive… So, ei repetition ta ke **1 ta Common jinish ee anbo**… eitai DP

arr **Greedy hocche Optimal Solution dibe BUT shb Path Traverse korbe Nah**

DP er 3 ta Step

1. RECURSION mere shb Path Apply kore dekhbo
2. Top Down Approach (Boro theke Choto problem ee nibo)
3. Bottom Up Approach (Choto problem gula niye Boro problem ee gora)

DP bujhar 1 ta Bhalo Basic Problem → 1 ta char konar Box er Top Left theke Bottom Right corner ee jabo… eitar Koita UNIQUE UNIQUE Path ase jawar… sheita Ber kore dekhao