6]

my summary DP

① accumulative = 1-d style dp

- Fibonaccie
- longest common seq
- only one array [subarray]

Type 1

→ F(0) & F(1) ⟶ = Bottom up

F(n), F(n-1) ⟶ F(0) Top Button

② Target + array = Bottomup = 2d array

Type 2

not subarray

- min To get The Target
- is it possible to get The Target
- How many ways

③ array + array + Target

0/1 knapsack

1	3	4	5	weight
1	4	9	7	val

Table weight = 7

Bottomup = 2d array

Type 3

④ 2 array / 2 string ⇒ ① ---- + subsequance [palindrome ---

Type 4

same arr diagonal

2 diff array row by row

subarr ~~can~~ subseq ~~will~~ ② To convert S1 To S2

5) Decision Tree + memoization

recursion + memoization

Types

Top-bottom

1]

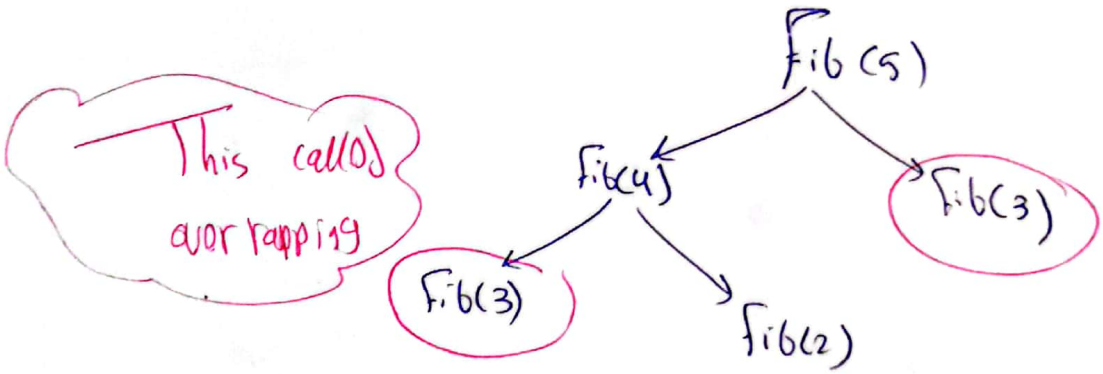
Dynamic programming

It could solve problem o(n) instead of (2^n)

= optimation Technique

*Dynamic programming → optimal sub structure → sub problem of main problem

→ overlapping sub problems



* 2 Types of problem : ① combinatoric problems

② optimation "

*combinatory : How many? - How many ways To make echange

- " " " To Traverse graph

- " " steps To get from A To B

- count The total number of solution

-optimation problems: -what is the min number of steps needed to get from A to B?

- what is the min coins to get sum?
- " " max profit

• DP is Technique to solve combinatory and optimization problem that is optimal solution depends on optimal solution to overlapping sub problems.

Frame Work of DP
5 = steps

1- Define objective function
f(i) is distinct ways ...
2- get base cases ⇒ f(0), f(1) ..
3- find the relation = العلاقة pattern بين
4- order of computation ⇒ [botten-up]
5- location of the answer

(ex) 0 1 2 3 4
1+2+3+4+5+.... what is the summation

f(4) = 5 + f(3)
f(5) = f(4) + 6

} نوع من الـ memoization

$f(n) = f(n-1) + n$

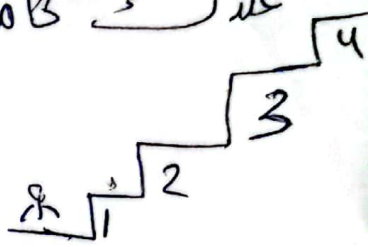
الحد يعتمد على اللي

DP: breaking down the problem into simpler sub problem

3]

عندك سلم و بتطلع سلمة واحدة او سلمتين مع بعض
عندك كام طريقة ؟

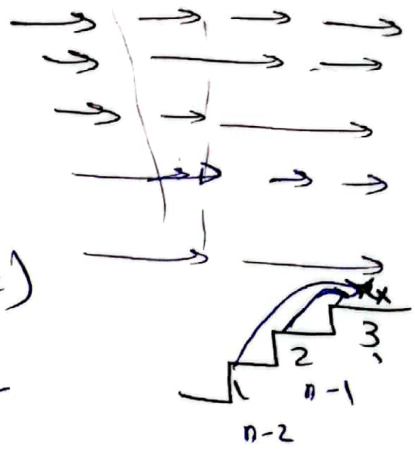
Type 1



دايماً افترضها سلم واحد
وبعدها سلمتين
و // ٣ وطلع العلاقة

$$f(1) = 1$$
$$f(2) = 2$$
$$f(3) = 3$$
$$f(4) = 5$$

$$f(n) = f(n-1) + f(n-2)$$



عشان نروح لـ ٣ ⇐ عندك اختيارين فقط [2 و 1]

* $f(n+1) = f(n) + 1$ ⇒ Future values

#تحدي يوم

Type1

عشان نحل DP عندي اسلوبين للحل

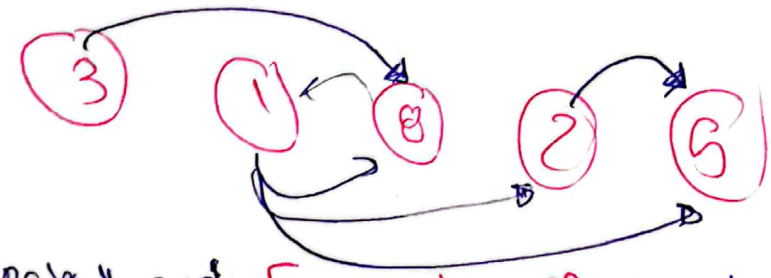
1- memoization
2- Tabulation

Q] Find The longest increasing subsequence (LIS)

3,1,8,2,5

= 3 elements

1) Visualize



انت كده هتقف عند node وتحسبي ليها F(i) عدد الـ nodes اللي جاية ليها

وهتروح اللي قبليها سؤالين

1- هل اللي انا فيها؟!
2- ايه الحاجة اللي جاية ليا؟!

F(0) = 1
F(1) = 1
F(2) = 2

$$L[A] = 1 + \max\{ L[k] \Big|_{k=0}^{n-1} \cdot \text{if } A[k] < A[n] \}$$

Code

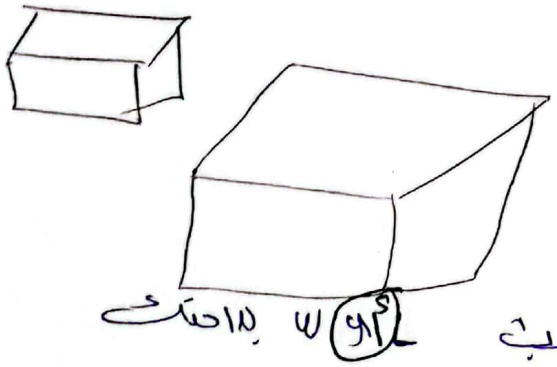
2

BoxStacking ⇒ Box has [Length, width, height]

Type 1

What is The heighest of The tallest possible stack??

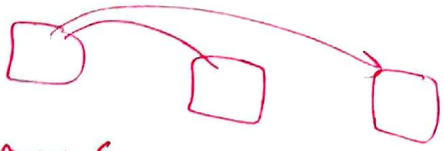
Sol



لو حابب لازم تعمل Sort من كبير لصغير بالمساحة و(الارتفاع)

State machine

لازم { 1- هل الـ Box ممكن أحطه فوقها
2- أنا الطول الي وصلت



Height[n] = Box[z] + max { Height[k] | k=0 to n-1 }

Box Height[z]

if only { width_k < width_n && length_k < length_n }

Code

4]

Find number of {0|0|0} given n

type

n = 3 ⇒ output = 6

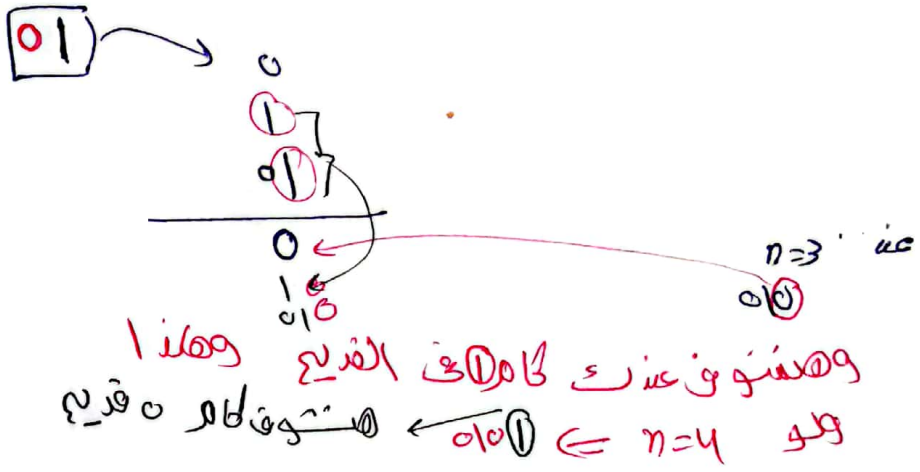
{0|0|0}

0
1
0
01
10
010

6

n = 1 ⇒ A = 1

n = 2 ⇒ A = 3



code:

57]

Q] Yaseira given array of cost ⇒ min cost To reach The end of Stair case: known That you can Jump (1) OR (2) steps. you can start from Index 0 (OR) Index (1)

Type 3

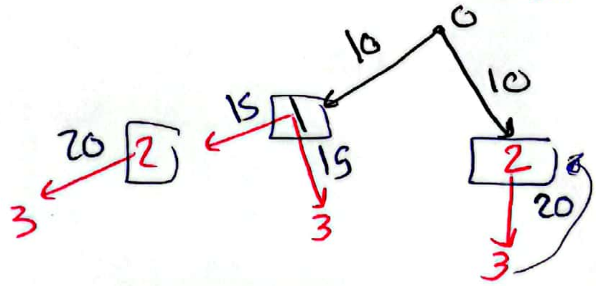
[10, 15, 20] (indices 0, 1, 2, 3)

Sol — sol is (15)

الفكرة كلها انت مش عارف انت عايز توصل لـ index رقم ٣ اعتبرت في هتبدأ من رقم 0 ولا رقم 1

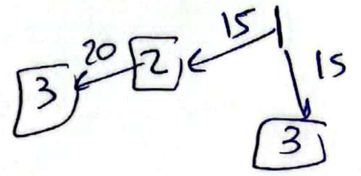
Sol ① ⇒ recursion + memoization

→ start at 0
→ " at 1 } → min



الـ Base هيرجع zero

الى عليها (هنكت a[i]) sum



⟸ هنعمل call تانى منفصل

هاخد min من ① و ⓪

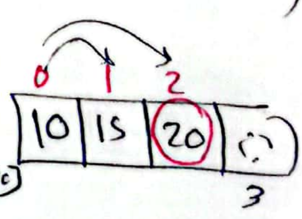
code = [ ]

Sol2 : dp ⇒ انت مش عارف البداية بس عارف النهاية فخلاص البدأ من النهاية لحد ورا

} ans = min(dp[0], dp[1])

dp[2] = 20 → عشان يروح لـ ٣ ⇐ دى تكلفة

dp[1] = min(15+0, 15+20)



dp[0] = a[0] + min(dp[1] + dp[2])

code = [ ]

$dp[i] = a[i] + \min(dp[i+1], dp[i+2])$

Type 1

91. Decode Ways

Solved

Medium | Topics | Companies

A message containing letters from `A-Z` can be **encoded** into numbers using the following mapping:

```
'A' -> "1"
'B' -> "2"
...
'Z' -> "26"
```

To **decode** an encoded message, all the digits must be grouped then mapped back into letters using the reverse of the mapping above (there may be multiple ways). For example, `"11106"` can be mapped into:

- `"AAJF"` with the grouping `(1 1 10 6)`
- `"KJF"` with the grouping `(11 10 6)`

Note that the grouping `(1 11 06)` is invalid because `"06"` cannot be mapped into `'F'` since `"6"` is different from `"06"`.

Given a string `s` containing only digits, return *the **number** of ways to **decode** it*.

The test cases are generated so that the answer fits in a **32-bit** integer.

Example 1:

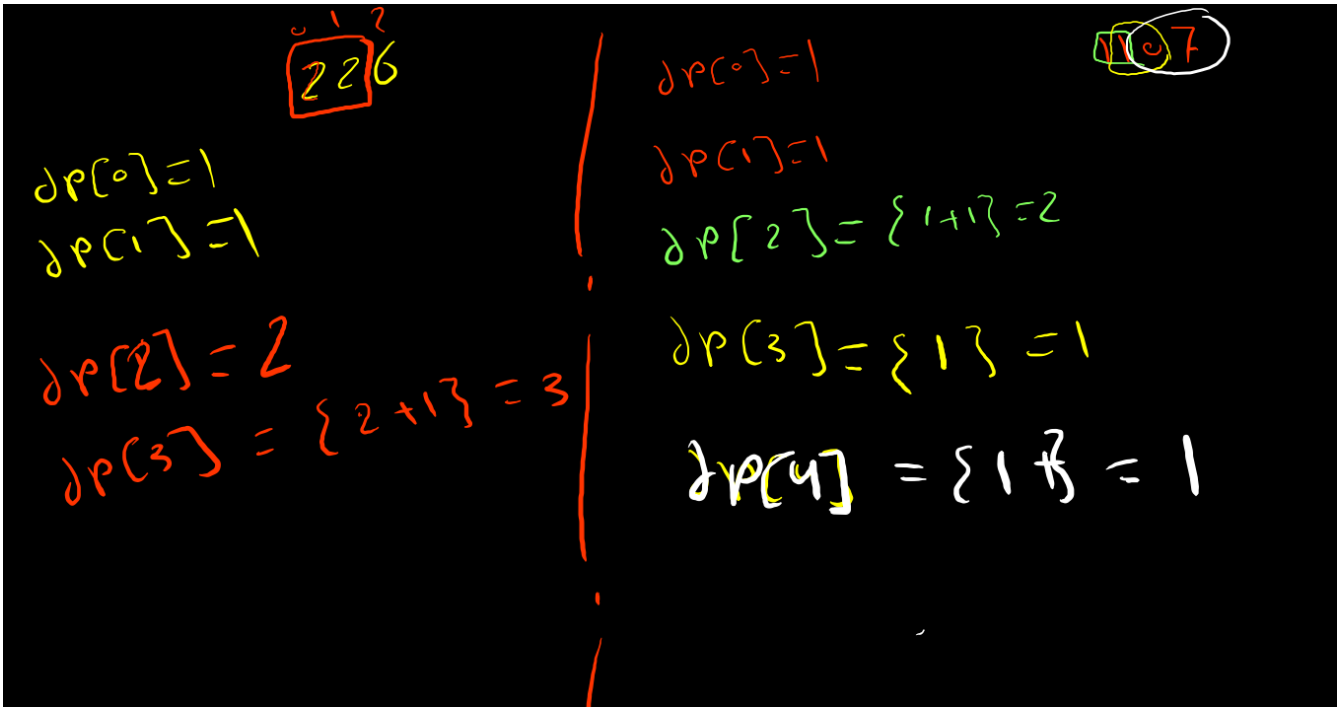
```
Input: s = "12"
Output: 2
Explanation: "12" could be decoded as "AB" (1 2) or "L" (12).
```

Example 2:

```
Input: s = "226"
Output: 3
Explanation: "226" could be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).
```

Example 3:

```
Input: s = "06"
Output: 0
Explanation: "06" cannot be mapped to "F" because of the leading zero ("6" is different from "06").
```



Code

Type 1

Stylus Color Line Eraser Backgrounds Undo Redo Pages Previous Next Erase Board Web Documents Show Desktop OpenBoard

413. Arithmetic Slices

Medium Topics Companies

An integer array is called arithmetic if it consists of **at least three elements** and if the difference between any two consecutive elements is the same.

- For example, `[1,3,5,7,9]`, `[7,7,7,7]`, and `[3,-1,-5,-9]` are arithmetic sequences.

Given an integer array `nums`, return *the number of arithmetic* ***subarrays*** *of* `nums`.

A **subarray** is a contiguous subsequence of the array.

Example 1:

```
Input: nums = [1,2,3,4]
Output: 3
Explanation: We have 3 arithmetic slices in nums: [1, 2, 3], [2, 3, 4] and
[1,2,3,4] itself.
```

Example 2:

```
Input: nums = [1]
Output: 0
```

```
int numberOfArithmeticSlices(vector<int>& arr) {
  int n = arr.size();

  int answer = 0;
  vector<int> dp(n, 0);


  for (int i = 2; i < n; ++i) {
    if (arr[i] - arr[i - 1] == arr[i - 1] - arr[i - 2]) {
      dp[i] += 1 + dp[i - 1];
      answer += dp[i];
    }
  }
  return answer;
}
```

1,2,3,4

0 | 0 | 0 | 0

at n = 2 ⇒ 1,2,3 ⇒ DP[2] = 1

at n = 3 ⇒ 2,3,4 ⇒ DP[3] 1+

ans = sum DP

Type 2 [array + Target]

- المسائل دى بتتحل 2d array / 1d array

* 1d array ⇒ كل شوية هتلاقى هل ممكن على الى واقف عليه؟ لحد ما توصل لـ Target

حلول متغيرك فى المستقبل

متطلع العلاقة من مين

1,2,3

عندك نوعين

1

هتقف فى

① at DP[2] → 1,2

at DP[5] → 1,2,3,4,5

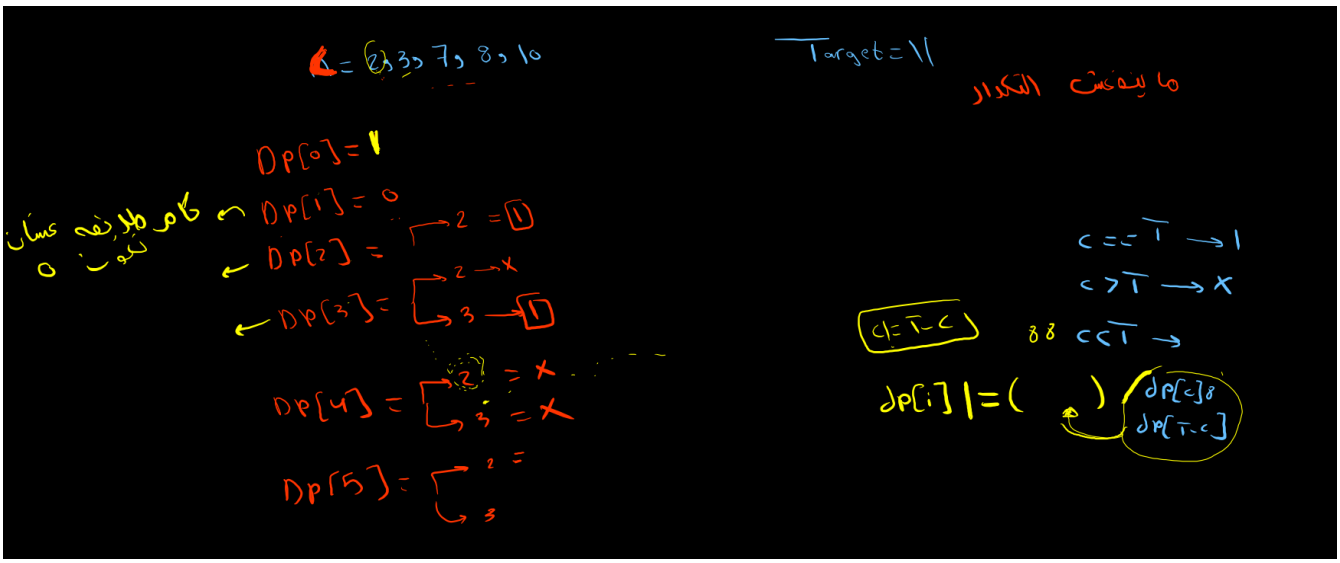
② ... → منفسيه على DP ما وبعدها C2

DP[0] =

DP[1] =

DP[2] =

DP[3] =



Code

322. Coin Change

Medium | Topics | Companies

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return *the fewest number of coins that you need to make up that amount*. If that amount of money cannot be made up by any combination of the coins, return `-1`.

You may assume that you have an infinite number of each kind of coin.

Example 1:

> **Input:** coins = [1,2,5], amount = 11
> **Output:** 3
> **Explanation:** 11 = 5 + 5 + 1

Example 2:

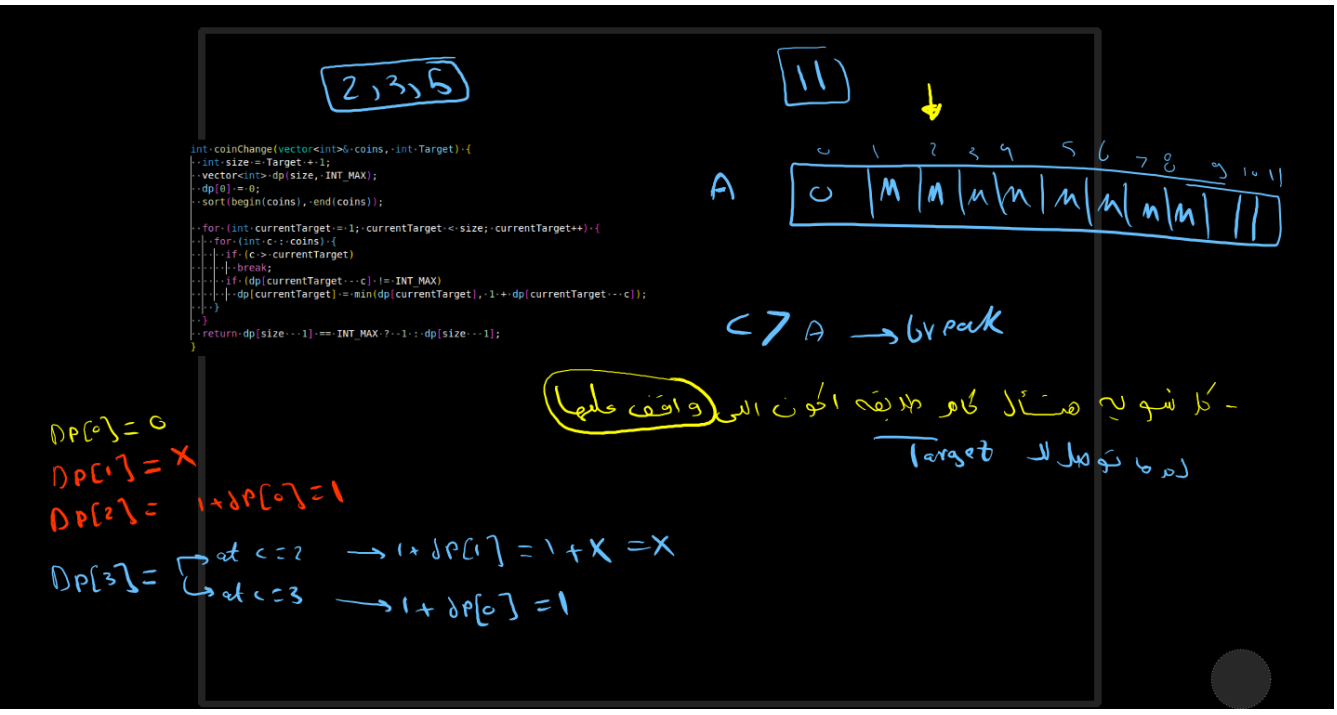
> **Input:** coins = [2], amount = 3
> **Output:** -1

Example 3:

> **Input:** coins = [1], amount = 0
> **Output:** 0

Constraints:

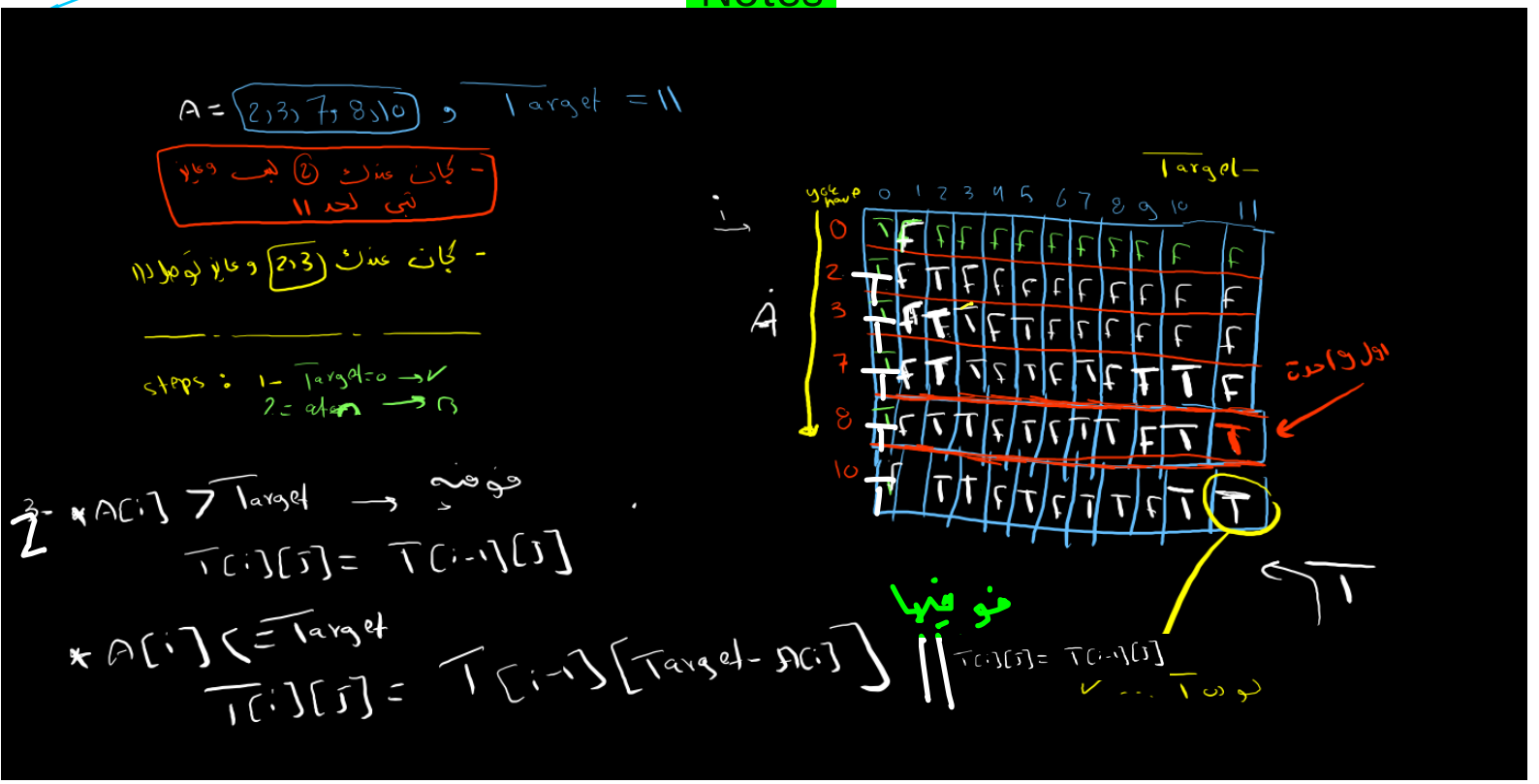
- `1 <= coins.length <= 12`
- `1 <= coins[i] <= 2^31 - 1`
- `0 <= amount <= 10^4`



Code :

Target [array + Target]

Notes



Code

ExtraNotes :

١- انا + اللي معايا = T ← احتمال جديد

٢- ازاى تحدق subset

T فوق T ← اطلع

else ← A ← دوج فوق عندك Target - A =

Type 2

416. Partition Equal Subset Sum

Medium | Topics | Companies

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.*

Example 1:

> **Input:** nums = [1,5,11,5]
> **Output:** true
> **Explanation:** The array can be partitioned as [1, 5, 5] and [11].

Example 2:

> **Input:** nums = [1,2,3,5]
> **Output:** false
> **Explanation:** The array cannot be partitioned into equal sum subsets.

-- 2 parition ==> sum/2
target = sum/2
then : array+target .. if true ..then tmaam

7]

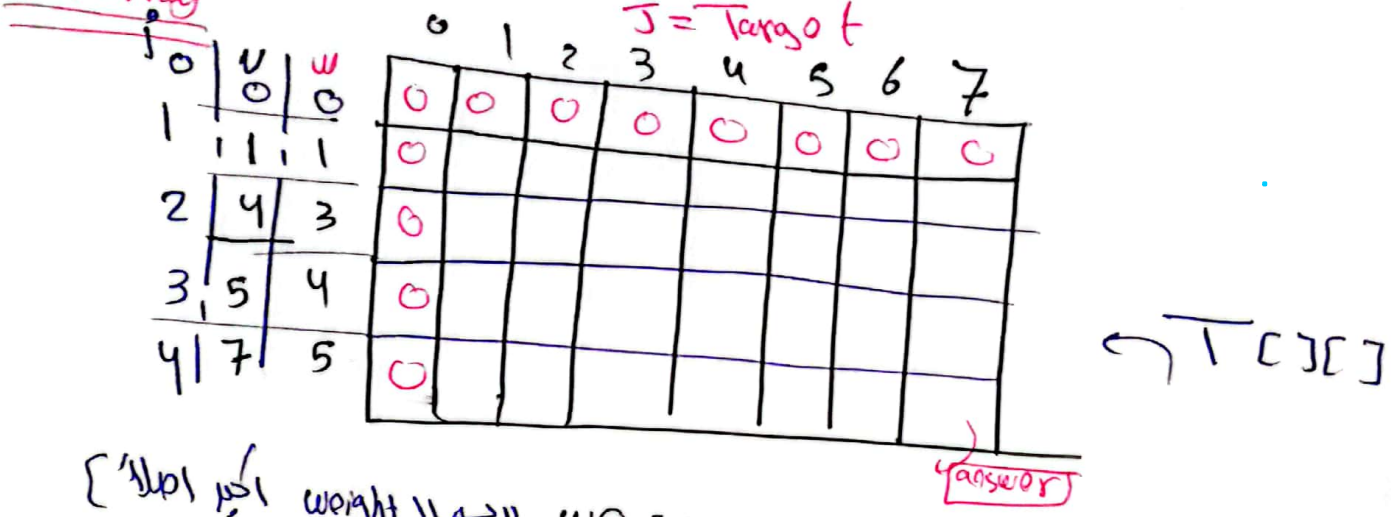
Q] knapsack 0/1 , weight = {1, 3, 4, 5}

Value = {1, 4, 5, 7}

what is The max value of weight = 7 ?

Type 3

2-D array



T[ ][ ]

$w[i-1] > J$ "سوال weight اكبر اصلا" ⇐ cell cell هنفضى

$T[i-1][J]$ يعني خد اللى فوقيها

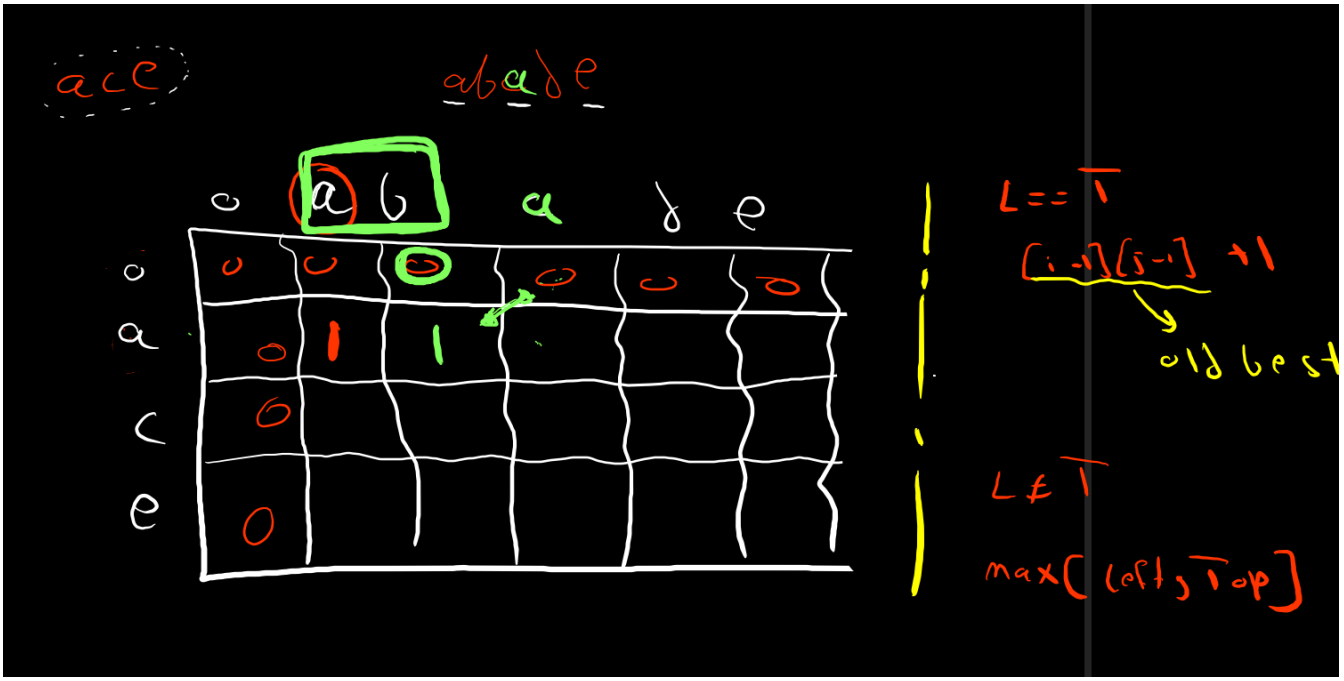
else ⇒ max (v[i-1] + هنعى , فوقيها)

$= \max(v[i-1] + T[i-1][J-w[i-1]] , T[i-1][J])$

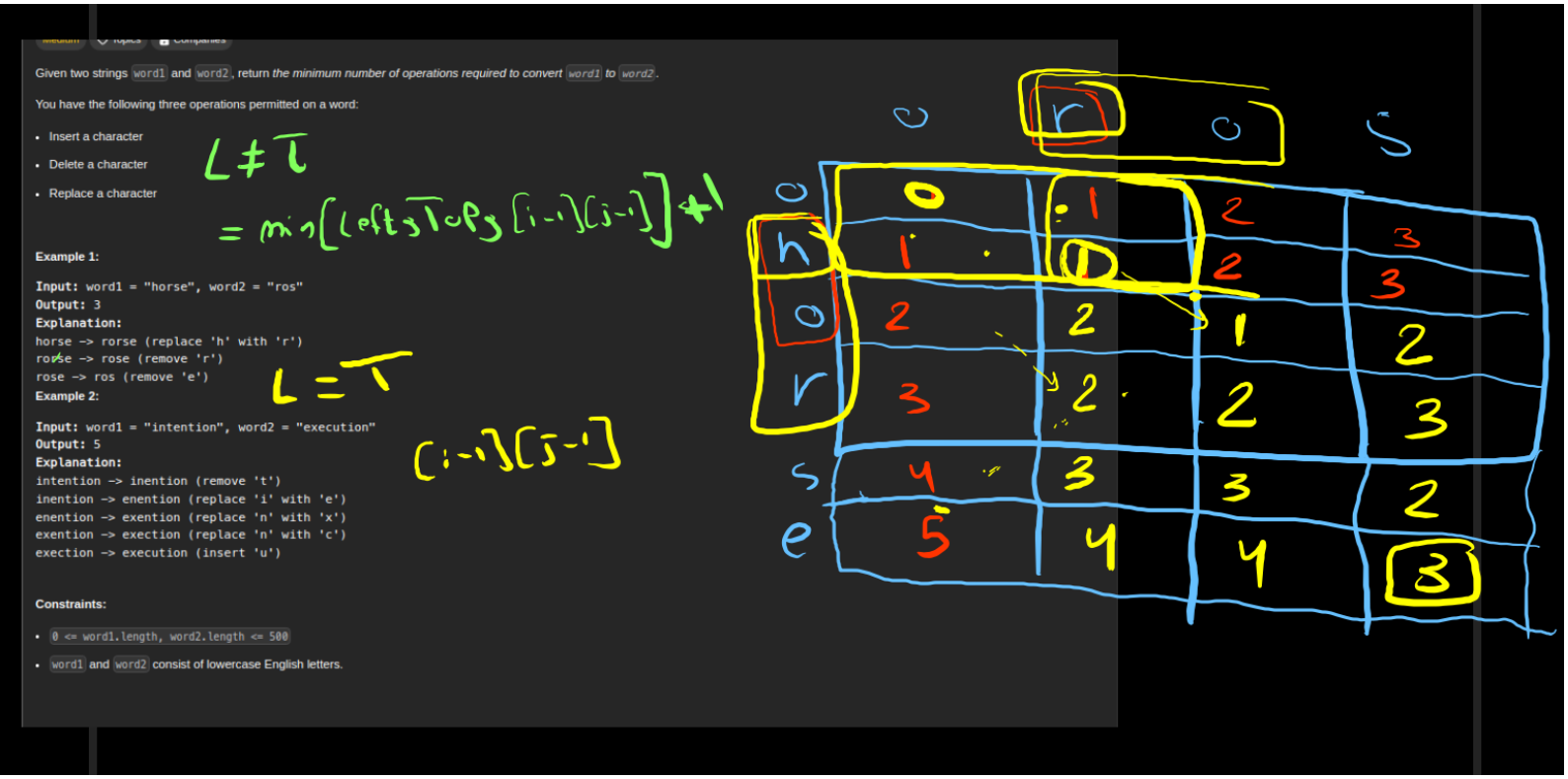
code

Knapsack 0/1 ⇒ ناخد الحبة بالاً وماتهمناش ناخدها اكتر من مرة

Type 4 : abcdef } acbef } find Longest common Subsequence



Code :



Code :

10. Regular Expression Matching

Solved

Hard | Topics | Companies

Given an input string `s` and a pattern `p`, implement regular expression matching with support for `'.'` and `'*'` where:

- `'.'` Matches any single character.
- `'*'` Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

Example 1:

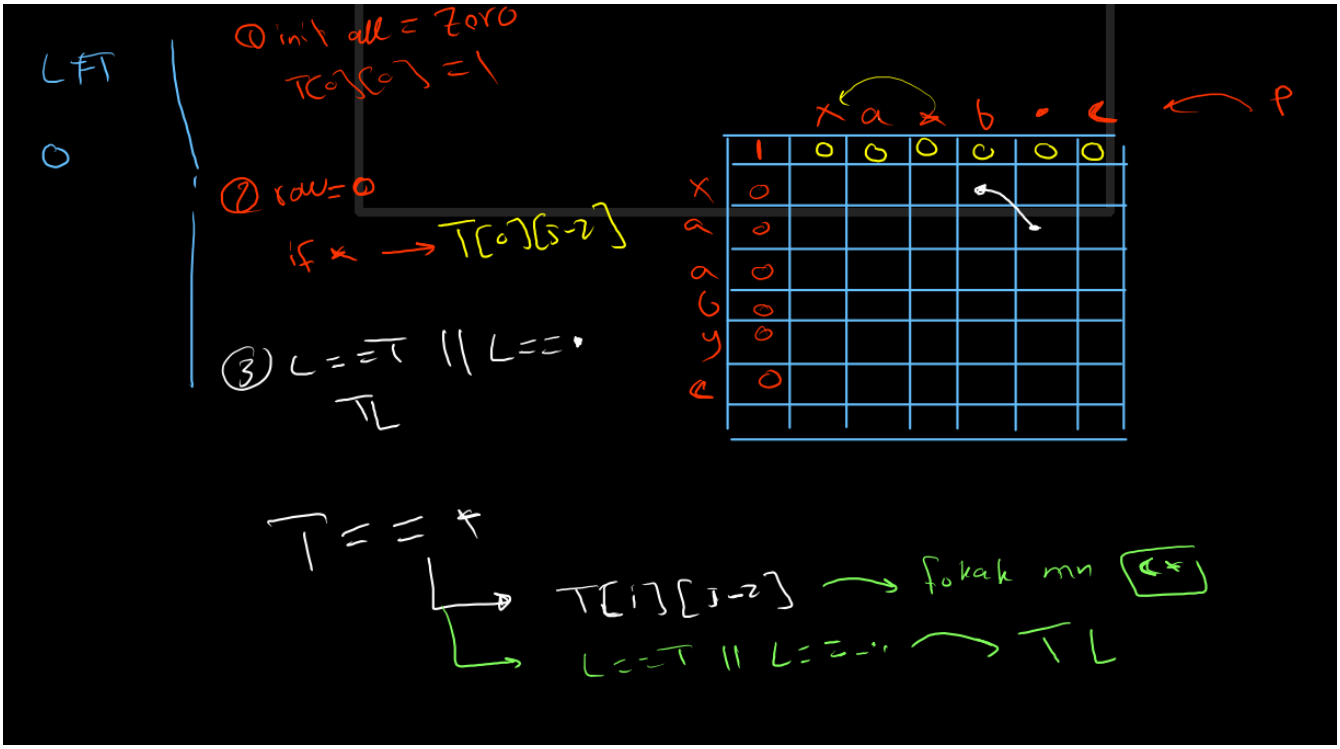
```
Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input: s = "aa", p = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".
```

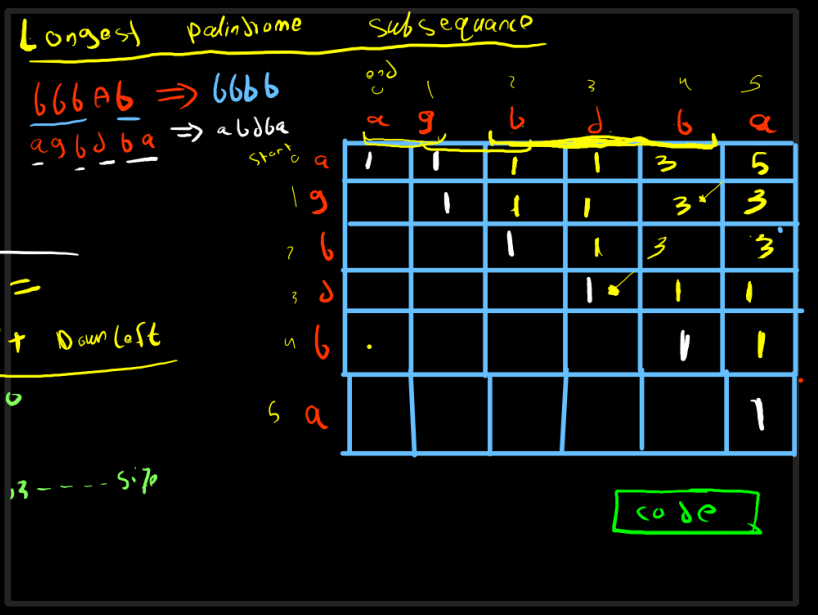
Example 3:

```
Input: s = "ab", p = ".*"
Output: true
Explanation: ".*" means "zero or more (*) of any character (.)".
```



CODE

Type 4


Longest palindrome subsequance
66A6 ⇒ 666
agbdba ⇒ abdba
L=1
≠
=
2+ Down left
① initial = 0
②
③
code