Q3

2. Given a list of n integers, $v_1, \ldots, v_n$, the product-sum is the largest sum that can be formed by multiplying adjacent elements in the list. Each element can be matched with at most one of its neighbors.
For example, given the list 1, 2, 3, 1 the product sum is $8 = 1 + (2 \times 3) + 1$, and given the list 2, 2, 1, 3, 2, 1, 2, 2, 1, 2 the product sum is $19 = (2 \times 2) + 1 + (3 \times 2) + 1 + (2 \times 2) + 1 + 2$.

1. Compute the product-sum of 1, 4, 3, 2, 3, 4, 2.

2. Give the optimization formula for computing the product-sum of the first j elements.

Ans:-

1, 2, 3, 1

1 x 2 = 2 , 2 x 3 = 6

1 + (2 x 3) + 1

2 2 1 3 2 1 2 2 1 2

4, 2 3 6 2 4 2, 3

(2x2) + 1 + (3x2) + 1 + (2x2) + 1 + 2

* 1, 4, 3, 2, 3, 4, 2

4, 12 6, 12

$1 + (4 \times 3) + 2 + (3 \times 4) + 2$

$= 29$

2.

$$dp[j] = \begin{cases} \max(dp[j-1] + \\ a_j, dp[j-2] + a_{j-1} \times \\ a_j \end{cases}$$

0 1 2 3 4 5

1, 4, 3, 2, 3, 4 $n = 6$

Base Case

$dp[0] = 1$

$dp[1] = \max(a[0], a[1])$

$a[0] \times a[1] = (1+4, 1 \times 4) = 5$

For $j = 2$ to $n-1$

$dp[j] = \max(dp[j-1] + a[i],$
$dp[j-2] + a[j-1] \times a[j])$

$dp[0] = 1 \quad dp[1] = 5.$

$j = 2$

$dp[2] = \max(5+3, 1+4 \times 3)$
$= 13$

$dp[3] = \max(13+2, 5+3 \times 2)$
$= 15$

$dp[4] = \max(15+3, 13+(3 \times 2))$
$= 19$

$dp[5] = \max(19+4, 15+(4\times3))$

$= 27$

1, 4, 3, 2, 3, 4

4 12 6, 12

$1+(4\times3)+2+(4\times3)$

$1+12+2+12=27$ ✓

(2) n types of coin

v(1) = 1 so you can always make change for any amount of money give an algorithm which makes change for an amount C with a few coins as possible.

Ans:-

ex:- amount 10 Coins [1, 2, 5, 10]

coin → # of coins ↑ amount

	0	1	2	3	4	5	6	7	8	9	10
	1	1	2	3	4	5	6	7	8	9	10
1											
2	1	1	1	3,2 2	4,2 2	3	3	4	4	5	5
5	1	1	1	2	2	1	3,2 2	4,2 2	4,3 3	3	2
10	1	1	1	2	2	1	2	2	3	3	2

→ $\begin{cases} \text{amount} < \text{coins} & a[i-1][j-1]. \\ \text{amount} > \text{coins} & \\ \rightarrow \min & \begin{cases} c[i-1][j], \\ c[i][j-\text{coins}(i)] \end{cases} \end{cases}$

(3) A, B, C, D, E

A : 3 x 5

B : 5 x 4

C : 4 x 6

D : 6 x 7

E : 7 x 3

j / i	2 (B)	3 (C)	4 (D)	5 (E)
1 (A)	60/1 A	132/2 B	258/3 C	294/2 B
2 (B)		120/2 B	308/2 B	258/2 B
3 (C)			168/3 C	198/3 C
4 (D)				126/4 D

Ans:- P = 3 5 4 6 7 3
0 1 2 3 4 5

[1,5] = A B C D E $1 \le k < 5$

0 258

m[1,5] = m[1,1] + m[2,5] +

3 * 5 * 3 = 303

60 198

k = 2 = m[1,2] + m[3,5] +

3 * 4 * 3 = 294

132 126

= m[1,3] + m[4,5] +

3 * 6 * 3 = 312

258 0

= m[1,4] + m[5,5] +

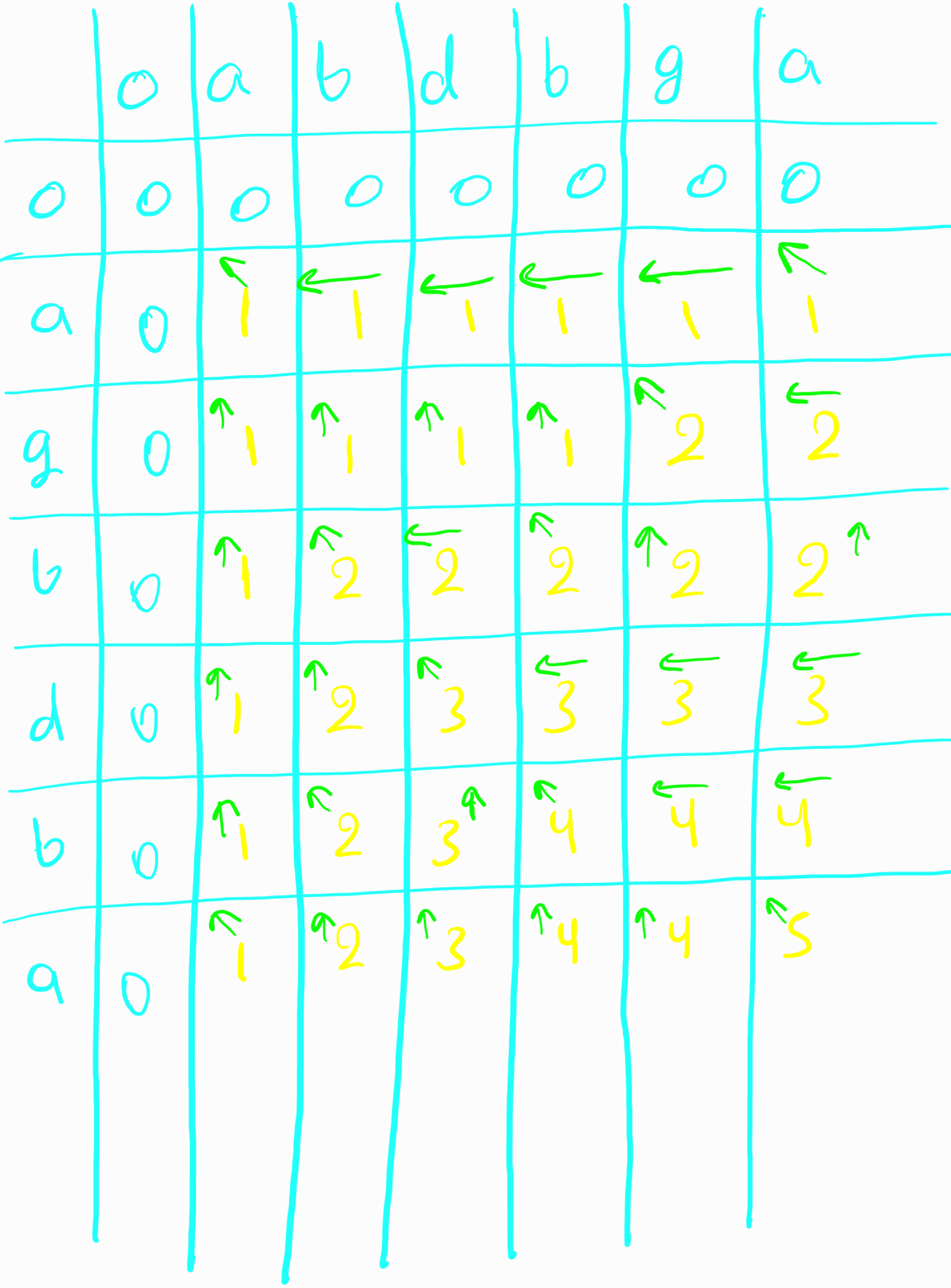
3 * 7 * 3 = 321

(A)((B)(C (D E)))

(4) Palindromic :-

abdbga , agbdba

Ans :-



→ a b d b a

Or :-

L[i][j] = 1 for 1 char

Same = L[i][j] =

L[i+1][j-1] + 2

not = L[i][j] =

max (L[i][j-1] ,

L[i+1][j])

a b d b g a

abdba.

⑥ palindromic

Question Three [20 marks]

A palindrome is a nonempty string over some alphabet that reads the same forward and backward. [1] Design an efficient algorithm that takes a sequence $x[1 \ldots n]$ and returns the Longest Palindromic Subsequence (*LPS*). [2] What is the running time of your algorithm?

(For instance, the sequence ACGTGTCAAAATCG has many palindromic subsequences, including ACGCA and CAAC. On the other hand, the subsequence ACG is not palindromic.)

L[i][j] = 1 for 1 char

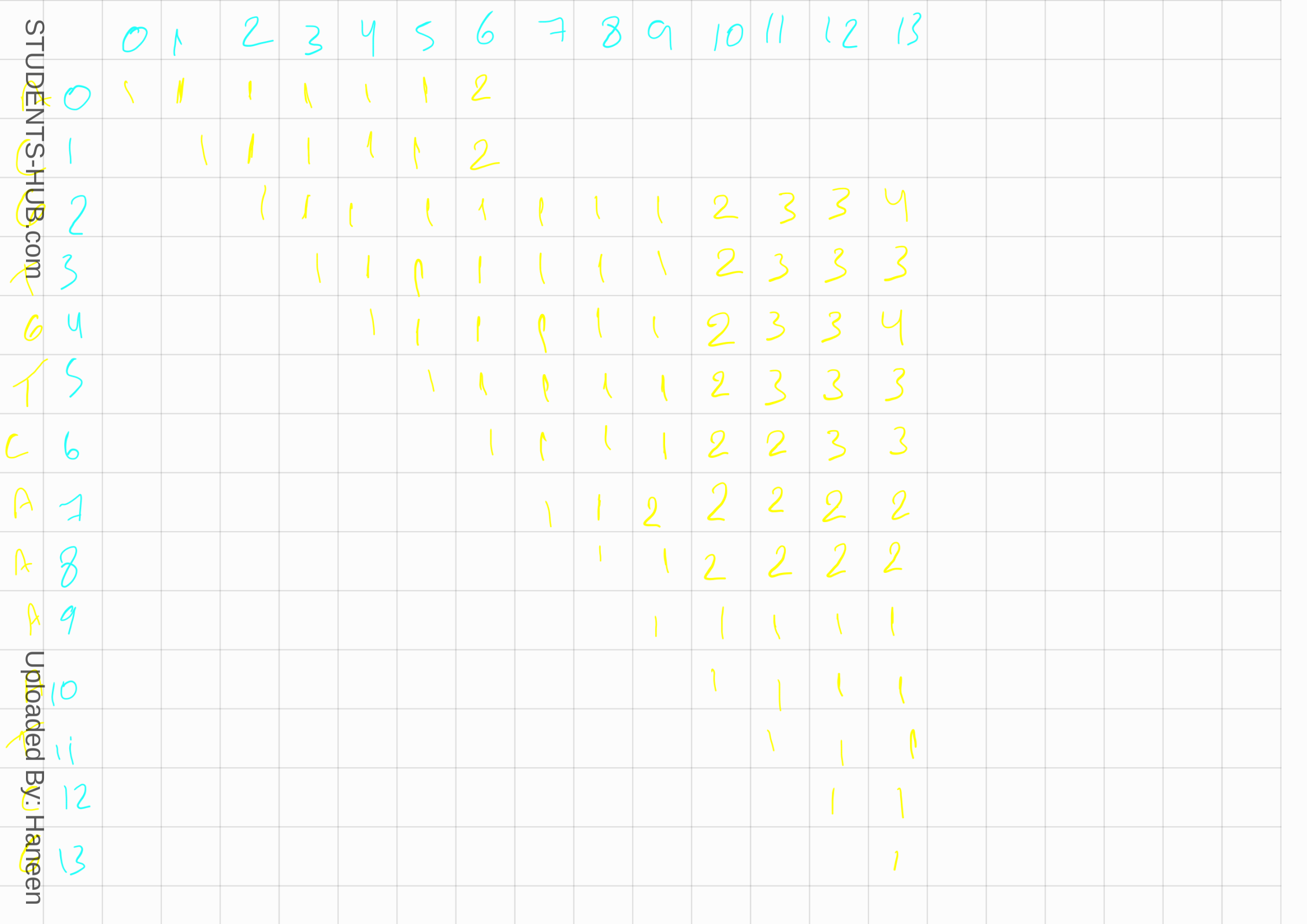
same = L[i][j] =

L[i+1][j-1] + 2

not = L[i][j] =

max(L[i][j-1],

L[i+1][j])



5

Question Three [25 marks]

1. Give an algorithm using dynamic programming to determine how many distinct ways there are to give a cents in change using any coins from among pennies, nickels, dimes, and quarters. For example, there are 6 ways to give 16 cents change:

- A dime, a nickel, and a penny;
- A dime and 6 pennies;
- 3 nickels and a penny;
- 2 nickels and 6 pennies;
- One nickel and 11 Pennies;
- And16 pennies.

2. Demonstrate your solution by showing a step-by-step solution for 12 cents change.

Ans:-

(1, 5, 10, 25)

16 → 6 ways

1- 10, 5, 1

2- 10, 6

3- 5, 5, 5, 1

4- 5, 5, 1, 1, 1, 1, 1, 1

5- 16(1)

6- 5, (11) 1

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4
10	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4	6	6
25	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4	6	6

coins

$dp[i][j]$

$$dp[i][j] = \begin{cases} dp[i-1][j] & coins(j) > i \\ dp[i-1][j] + dp[i][coins(j)-i] & \text{if not} \end{cases}$$