Q1

ANSWER:

Minimum scalar multiplications = 43

Optimal paranthesisation = $[(M_1 M_2) M_3] M_4$

WORKING:

Input:

M_1	M_2	M_3	M_4
1×5	5×2	2×3	3×9
$m_0 \quad m_1$	$m_1 \quad m_2$	$m_2 \quad m_3$	$m_3 \quad m_4$

$M_i = m_{i-1} \times m_i$

m_0	m_1	m_2	m_3	m_4
1	5	2	3	9

Cases:

$C(i,i) = 0$

$$C(i,j) = \min \left\{ C(i,k) + C(k+1,j) + (m_{i-1} \times m_k \times m_j) \right. \quad \forall k \text{ s.t } i \leq k < j$$

Run-through (done diagonally):

$C(1,1) = C(2,2) = C(3,3) = C(4,4) = 0$

$\forall C(i,j)$ s.t $i > j = X$

$$C(1,2) = \min \left\{ C(1,1) + C(2,2) + (\overset{m_0 \times m_1 \times m_2}{1 \times 5 \times 2}) \right. = 10$$

($C(1,1) = 0$, $C(2,2) = 0$)

$$C(2,3) = \min \left\{ C(2,2) + C(3,3) + (m_1 \times m_2 \times m_3) \right. = 30$$

($C(2,2) = 0$, $C(3,3) = 0$, $m_1 = 5$, $m_2 = 2$, $m_3 = 3$)

$$C(3,4) = \min \left\{ C(3,3) + C(4,4) + (m_2 \times m_3 \times m_4) \right. = 54$$

($C(3,3) = 0$, $C(4,4) = 0$, $m_2 = 2$, $m_3 = 3$, $m_4 = 9$)

	1	2	3	4
1	0			
2	X	0		
3	X	X	0	
4	X	X	X	0

	1	2	3	4
1	0	10		
2	X	0	30	
3	X	X	0	54
4	X	X	X	0

Marks _____ Signature of Instructor _____

$C(1,3) = \min \begin{cases} C(1,1) + C(2,3) + (m_0 \times m_1 \times m_3) = 45 \\ \quad 0 \quad\quad 30 \quad\quad 1 \;\; 5 \;\; 3 \\ C(1,2) + C(3,3) + (m_0 \times m_2 \times m_3) = 16 \checkmark \\ \quad 10 \quad\quad 0 \quad\quad 1 \;\; 2 \;\; 3 \end{cases}$

0	10	96	
X	0	30	121
X	X	0	54
X	X	X	0

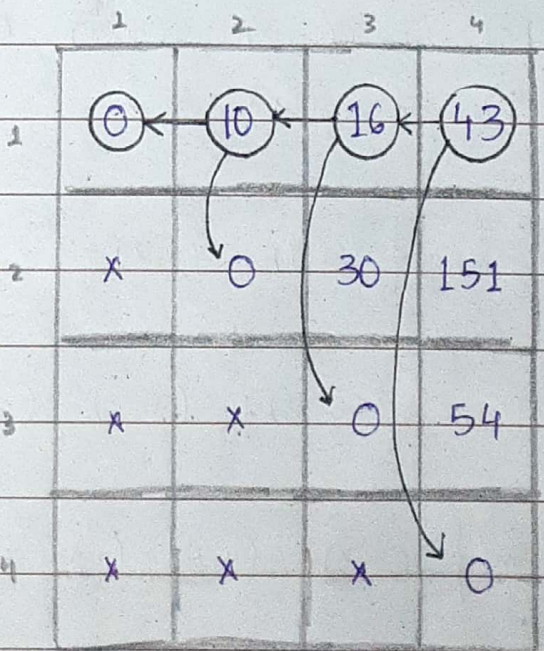
$C(2,4) = \min \begin{cases} C(1,2) + C(3,4) + (m_1 \times m_2 \times m_4) = 154 \\ \quad 10 \quad\quad 54 \quad\quad 5 \;\; 2 \;\; 9 \\ C(1,3) + C(4,4) + (m_1 \times m_3 \times m_4) = 151 \checkmark \\ \quad 16 \quad\quad 0 \quad\quad 5 \;\; 3 \;\; 9 \end{cases}$

0	16	16	
X	0	30	151
X	X	0	54
X	X	X	0

$C(1,4) = \min \begin{cases} C(1,1) + C(2,4) + (m_0 \times m_1 \times m_4) = 196 \\ \quad 0 \quad\quad 151 \quad\quad 1 \;\; 5 \;\; 9 \\ C(1,2) + C(3,4) + (m_0 \times m_2 \times m_4) = 82 \\ \quad 10 \quad\quad 54 \quad\quad 1 \;\; 2 \;\; 9 \\ C(1,3) + C(4,4) + (m_0 \times m_3 \times m_4) = 43 \checkmark \\ \quad 16 \quad\quad 0 \quad\quad 1 \times 3 \times 9 \end{cases}$

0	10	16	43
X	0	30	151
X	X	0	54
X	X	X	0

Backtracking:



ROUGH WORK & PROOF OF CONCEPT : (Shown on next pages)

$M_1 \cdot M_2 \cdot M_3 \cdot M_4$

$1 \times 5 \quad 5 \times 2 \quad 2 \times 3 \quad 3 \times 9$

$m_0 \; m_1 \quad m_1 \; m_2 \quad m_2 \; m_3 \quad m_3 \; m_4$

United Energy Pakistan UEP

Rules:

$$c(i,i) = 0$$

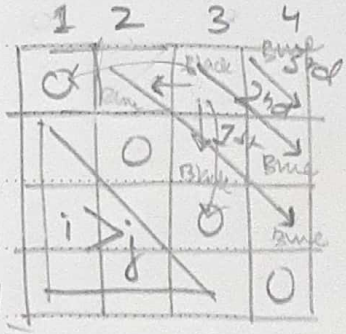
$$c(i,j) = \min_{\forall k \text{ s.t } i \leq k < j} \left\{ c(i,k) + c(k+1,j) + (m_{i-1} \times m_k \times m_j) \right.$$

dimensions:

$$M_i = m_{i-1} \times m_i$$

m_0	m_1	m_2	m_3	m_4
1	5	2	3	9

First diagonal:



$k \neq j$ $\quad c(i,k) \quad c(k+1,j)$

i	j	k	$k+1$	m_{i-1}	m_k	m_j
(1, 2)		1	2	m_0	m_1	m_2

$1 \times 5 \times 2 = 10$

(2,3)		2	3	(2,2)	(3,3)	m_1 m_2 m_3
(3,4)		3	4	(3,3)	(4,4)	m_2 m_3 m_4
(1,3)	MIN 1		2	(1,1)	(2,3)	$m_0 \times m_1 \times m_3$
(1,3)	2		3	(1,2)	(3,3)	$m_0 \times m_2 \times m_3$

m_0	m_1	m_2	m_3	m_4
1	5	2	3	9

UEP

All $i \le k < j$: $k \neq j$

i	j	i-1	k	k+1	$m_{i-1} \times m_k \times m_j$	c(i,k)	c(k+1,j)	x_k
1	2	0	1	2	$m_0 \times m_1 \times m_2$ $1 \times 5 \times 2$ $= 10$	c(1,1) 0	c(2,2) 0	10
2	3	1	2	3	$m_1 \times m_2 \times m_3$ $5 \times 2 \times 3$ $= 30$	c(2,2) 0	c(3,3) 0	30
3	4	2	3	4	$m_2 \times m_3 \times m_4$ $2 \times 3 \times 9$ $= 54$	c(3,3) 0	c(4,4) 0	54
1	3	0	1	2	$m_0 \times m_1 \times m_3$ $1 \times 5 \times 3$ $= 15+30 = 45$	c(1,1) 0	c(2,3) 30	16
1	3	0	2	3	$m_0 \times m_2 \times m_3$ $1 \times 2 \times 3$ $= 6+10 = (16)$ ✓	c(1,2) 10	c(3,3) 0	
2	4	1	2	3	$m_1 \times m_2 \times m_4$ $5 \times 2 \times 9$ $= 90+10+54 = 154$	c(1,2) 10	c(3,4) 54	151
2	4	1	3	4	$m_1 \times m_3 \times m_4$ $5 \times 3 \times 9$ $= 135+16 = (151)$ ✓✓	c(1,3) 16	c(4,4) 0	

i	j	i-1	k	k+1	$m_{i-1} \times m_k \times m_j$	c(i,k)	c(k+1,j)	Minimum
1	4	0	1	2	$m_0 \times m_1 \times m_4$ $1 \times 5 \times 9$ $= 45 + 151$	c(1,1) 0	c(2,4) 151	
1	4	0	2	3	$m_0 \times m_2 \times m_4$ $1 \times 2 \times 9$ $= 18 + 10 + 54 =$	c(1,2) 10	c(3,4) 54	43
1	4	0	3	4	$m_0 \times m_3 \times m_4$ $1 \times 3 \times 9$ $= 27 + 16 =$ (43)	c(1,3) 16	c(4,4) 0	

	1	2	3	4
1	(0)	(10)	(16)	(43)
2	x	(0)	30	151
3	x	x	(0)	54
4	x	x	x	(0)

Answer is 10

Q2 ANSWER: 5

WORKING:

Input: s_1 = INTENTION = a , s_2 = EXECUTION = b

$a_1 \dots a_i \dots a_m$ $\quad$ $b_1 \dots b_j \dots b_n$

Cases:

$ED(0, j) = j$ $\quad$ $|a| = 0$ and only b has letters so add/delete all b letters

$ED(i, 0) = i$ $\quad$ $|b| = 0$ and only a has letters so add/delete all a letters

$$ED(i,j) = \begin{cases} \text{match } (a_i == b_j) \text{ — } ED(i-1, j-1) \\ \text{mismatch } (a_i \neq b_j) \text{ — } \min \begin{cases} 1 + ED(i, j-1) \\ 1 + ED(i-1, j) \\ 1 + ED(i-1, j-1) \end{cases} \end{cases}$$

		O	E	X	E	C	U	T	I	O	N
	0	0	1	2	3	4	5	6	7	8	9
I	1	1	1	2	3	4	5	6	6	7	8
N	2	2	2	3	4	5	6	7	7	7	
T	3	3	3	3	4	5	5	6	7	8	
E	4	3	4	3	4	5	6	7	8	9	
N	5	4	4	4	4	5	6	7	8	8	
T	6	5	5	5	5	5	5	6	7	8	
I	7	6	6	6	6	6	6	5	6	6	
O	8	7	7	7	7	7	7	6	5	6	
N	9	8	8	8	8	8	8	7	6	5	

I	N	T	E	N	-	T	I	O	N	
E	X	-	E	C	U	T	I	O	N	

ROUGH WORK SHOWN FOR PROOF OF CONCEPT: (On next page →)

$ED(0,j) = j$ $\quad a_1 \dots a_i \dots a_m$

$ED(i,0) = i$ $\quad b_1 \dots b_j \dots b_n$

$$ED(i,j) = \begin{cases} ED(i-1, j-1) & a_i == b_j \\ \min \begin{cases} 1 + ED(i, j-1) & \text{insert } b_j \text{ into A} \quad \left[\frac{-}{b_j}\right] \\ 1 + ED(i-1, j) & \text{remove } a_i \text{ from A} \quad \left[\frac{a_i}{-}\right] \\ 1 + ED(i-1, j-1) & \text{replace } a_i \text{ with } b_j \quad \left[\frac{a_i}{b_j}\right] \end{cases} \end{cases}$$

a = INTENTION b = EXECUTION

		O	E	X	E	C	U	T	I	O	N
		0	1	2	3	4	5	6	7	8	9
I	1	1	2	3	4	5	6	6	7	8	
N	2	2	2	3	4	5	6	7	7	7	
T	3	3	3	3	4	5	5	6	7	8	
E	4	3	4	3	4	5	6	7	8	9	
N	5	4	4	4	4	5	6	7	8	8	
T	6	5	5	5	5	5	5	6	7	8	
I	7	6	6	6	6	6	6	5	6	6	
O	8	7	7	7	7	7	7	6	5	6	
N	9	8	8	8	8	8	8	7	6	5	

INT E N - T I O N
E X - E C U T I O N

Full Name:
Student ERP I/D No:
Subject:
Quiz #: Date :
Class/Section:
Teacher's Name:

Q4

Input : $x = \langle x_1, \ldots, x_m \rangle$

$y = \langle y_1, \ldots, y_m \rangle$

Cases :

$SCS(0,j) = j$ (Nothing in x so SCS is all elements of y)

$SCS(i,0) = i$ (Nothing y so SCS is all elements of x)

$$SCS(i,j) = \begin{cases} x_i == y_j & 1 + SCS(i-1, j-1) \\ x_i != y_j & 1 + \min \begin{cases} SCS(i-1, j) & \text{Remove } x_i \text{ check } x_{i-1} = y_j ? \\ SCS(i, j-1) & \text{Remove } y_j \text{ check } x_i = y_{j-1} ? \end{cases} \end{cases}$$

Traverse in row-wise fashion :

i-1, j-1	i-1, j
i, j-1	i, j

Algorithm : SCSFinding (~~int [][] SCS, int m, int n~~) {

```
SCS = int[m][n];
for i = 0 to m
{   for j = 0 to n
    {
        if (i == 0) then SCS[i][j] = j;
        else if (j == 0) then SCS[i][j] = i;
        else if (x[i] == y[j]) then SCS[i][j] = 1 + SCS[i-1][j-1];
        else SCS[i][j] = 1 + min( SCS[i-1][j] , SCS[i][j-1] );
    }
}
```

Marks _____ Signature of Instructor _____

return SCS[m][n];
}

Time complexity : O(m x n)

↳ Two for-loops nested; one runs till m, the other till n where m is length of x and n is length of y.

Space complexity : O(m x n)

↳ We are maintaining a (M+1) x (N+1) matrix for traversal,

WORKING (AS PROOF OF CONCEPT): Shown on next pages →

$x = \langle x_1, \ldots, x_m \rangle$

$y = \langle y_1, \ldots, y_n \rangle$

Time complexity =

Space complexity = $O(m \times n)$

$x = \langle 1, 2, 2, 3 \rangle$

$y = \langle 2, 3, 2 \rangle$

$z = SCS = \langle 1, 2, 2, 3, 2 \rangle$

Cases:

$SCS(0, j) = j$ (a_i = zero length, b_j = j length)

$SCS(i, 0) = i$ (b_j = zero length, a_i = i length)

$$SCS(i,j) = \begin{cases} 1 + SCS(i-1, j-1) & a_i == b_j \\ 1 + \min \begin{cases} SCS(i-1, j) & \text{Remove } a_i, \text{ check } a_{i-1} = b_j ? \\ SCS(i, j-1) & \text{Remove } b_j, \text{ check } a_i = b_{j-1} \end{cases} \end{cases}$$

priority

i-1, j-1	i-1, j
i, j-1	i, j

0	1	2	3	4
A	B	B	C	

a

				3	→ j
		B	C	B	

i	j	a_i	b_i	c(i,j-1)	c(i-1,j)	c(i,j)	1+ whatever
1	1	A	B	c(1,0) = 1	c(0,1) = 1	c(0,0) = 1	2
1	2	A	C	c(1,1) = 2	c(0,2) = 1	(0,1) = 1	2
1	3	A	B	c(1,2) = 2	c(0,3) = 1	c(0,2) = 2	2
2	1	B	B			c(1,0) = 1	2
2	2	B	C	c(2,1) = 2	c(1,2) = 2	c(1,1)	3
2	3	B	B	c(2,2)	c(1,3)	c(1,2) = 2	3
3	1	B	B	c(3,0)	c(2,1)	c(2,0) = 2	3
3	2	B	C	c(3,1) = 3	c(2,2) = 3	c(2,1)	4
3	3	B	B	c(3,2)	c(2,3)	c(2,2) = 3	4
4	1	C	B	c(4,0) = 4	c(3,1) = 3		4
4	2	C	C	c(4,1)	c(3,2)	c(3,1) = 3	4
4	3	C	B	c(4,2) = 4	c(3,3) = 4		5

saying
of being

		B		
	0		C	
0	0	1	2	3
A	1	2	2	2
B	2	2	3	3
B	3	3	4	4
C	4	4	4	5

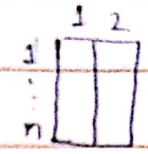
Q6 : Input : $S = \langle s_1, \cdots, s_n \rangle$

Cases :

- j s.t. $1 \leq j < i$ so $s_1 \ldots s_j \ldots s_i \ldots s_n$
- $LIS[i] = 1$ for $\forall i$ and $|LIS| = n$
- $LIS[1] = s_1$ (if only one element makes up sequence, then it is SCS)
- $LIS[i] = \forall j$ s.t. $1 \leq j < i$
  - $s_i \leq s_j$ — $LIS[i]$.
  - $s_i > s_j$ — max { $LIS[i]$, $1 + LIS[j]$. }

Marks _____ Signature of Instructor _____

Algorithm :




```
Find LIS ( s [1 ... n] ) {
    LIS = int [n] [2] ;   // matrix array of n x 2
    for i = 1 to n. // initialize all to 1.
    {   LIS [i][1] = 1 ; }
    LIS [1][1] = s1 ;

    for i = 2 to n
    {
      for j = 1 to i
      {
          if (si > sj)
          {
              if (LIS [i][1] > 1 + LIS[j][1])
              {   LIS[i][1] = LIS [i][1];
                  LIS[i][2] = i;
              }
              else
              {   LIS [i][1] = LIS[j][1] + 1;
                  LIS[i][2] = j ;
              }
          }
          else {
              LIS[i][1] = LIS [i][1];
              LIS [i][2] = [illegible] ;
          }
      }
    }
}
```


Marks _____ Signature of Instructor _____

~~int n = LIS~~ int i = n ;

while (i > 0) // Backtracking to get subsequence printed

{

LIS[i][1] . print ;

i = LIS [i][2] ;

}

}

// end

Time complexity = $O(n \times n)$

↓

Two for loops with one running to n and one running to i. Since max value of i is n so sum of form $\frac{n(n+1)}{2}$ exists which is $O(n^2)$.

Space complexity : $O(n)$

↳ LIS array recorded for traversal which is size $n \times 2$ which is $O(n)$.

Working to show proof of concept:

1	2	3	4	5	6	7	8	9
10	22	9	33	21	50	41	60	80

UEP

j i $j < i$

i = 0 then 0
i = 1 then a[1]
j = then 1.

i	j	LIS[i]	a_i	a_j	LIS[j] + 1	max (LIS[i], LIS[j+1]) or LIS[i]
2	1	LIS[2] 1	22	10	LIS[1] + 1 ✓ 1 + 1 = 2	2
3	1	LIS[3] 1 ✓	9	10		1
3	2	LIS[3] 1 ✓	9	22		1
4	1	LIS[4] 1	33	> 10	LIS[1] + 1 ✓ 1 + 1 = 2	2
4	2	LIS[4] 1	33	> 22	LIS[2] + 1 2 + 1 = 3 ✓	3
4	3	LIS[4] 1	33	> 9	LIS[3] + 1 1 + 1 = 2 ✓	2
5	1	LIS[5] 1	21	> 10	LIS[1] + 1 1 + 1 = 2	2
5	2	" 1	21	22		1
5	3	" 1	21	> 9	LIS[3] + 1 1 + 1 = 2	2
5	4	" 1	21	< 33	L	1
6	1	LIS[6] 1	50	> 10	LIS[1] + 1 1 + 1	2
6	2	1	50	> 22	LIS[2] + 1 2 + 1	3
6	3	1	50	> 9	LIS[3] + 1 1 + 1	2

	10	22	9	33	21	50	41	60	80
①	10	1							
②	22	2							
③	9	1	1						
④	33	2	3	2					
⑤	21	2	1	2	1				
⑥	50	2	3	2	3	3			
⑦	41								
⑧	60								
⑨	80								

i

j

1	1
2	2
3	1
4	3
5	2
6	4

i	j	LIS[i]	a_i	a_j	LIS[j]+1	max or LIS[i]
6	4	8 1	50	> 33	LIS[4]+1 3+1	4
6	5	1	50	> 21	LIS[5]+1 2+1	3

Time = $O(n^2)$

Space = $O(n)$.

Q3 : Input : $S = \langle s_1, \ldots s_i, \ldots, s_j, \ldots, s_n \rangle$

$i \leq j$

Cases :

n.length = 0 so string is null so no palindrome.

n.length = 1 so only one element in sequence so palindrome of size 1.

$PS(i,j) =$

- $j = i$ — 1
- $s_i == s_j$ — match — 2 — $j = i+1$.
  - $2 + PS(i+1, j-1)$ — Include and remove.
- $s_i \neq s_j$ — not match — max —
  - $PS(i, j-1)$ — Remove s_j
  - $PS(i+1, j)$ — Remove s_i

Traverse as diagonal like in Chain Matrix Multiplication.

$i, j-1$	i, j
$i+1, j-1$	$i+1, j$

```
Find LPS ( s[1 .... n] ) {
  if ( n == 0 || n == 1 )    // string is empty or has only one element
      return n ;
  PS = int [n][n] ; // matrix of size nxn.
for   i = 1  to  n
{     PS [i][i] = 1 ;  }  // all initialized to 1 where j = i
int  incrementor = 1 ;
while ( incrementor < n )
{
      for  i = 1  to  n
      {
          j = i + incrementor ;
          if (a_i == a_j)
          {
              if (j == i+1)  then   PS[i][j] = 2 ;
              else                  PS[i][j] = 2 + PS[i+1][j-1];
          }
          else
                PS[i][j] = max( PS[i][j-1] , PS[i-1][j] );
      }
      incrementor++ ;
}
return PS[1][n] ;
} // end.
```

Time complexity : $O(n^2)$

↳ Two loops iterating over nxn matrix total so $O(n^2)$

Space complexity = $O(n^2)$ → cause nxn matrix made to keep track.

WORKING (as proof of concept shown on next pages →):

Marks_____ Signature of Instructor_____

}

}

LONGEST PALINDROMIC SUBSEQUENCE

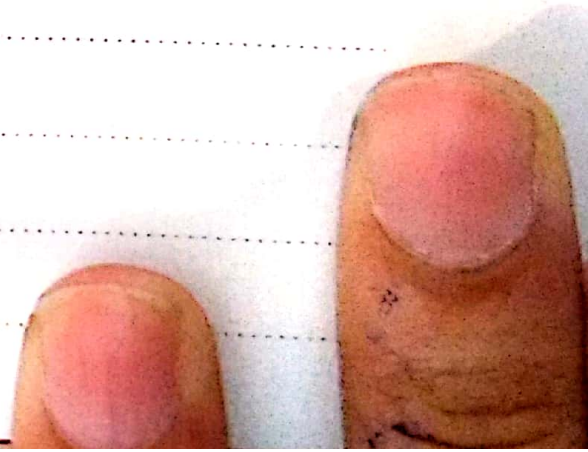
Brute force : $O(2^n)$

No of subproblems : $O(n^2)$

Space complexity : $O(n^2)$

Input : $a_1 \dots a_i \dots a_j \dots a_n$

$i \leq j$



Cases :

$n = 0$ a length is 0 so no palindrome

$n = 1$ so return 1 [only one element so it is palindrome]

$PS(i,i) = 1$

$$PS(i,j) = \begin{cases} a_i == a_j \text{ (same)} \begin{cases} 2 & j = i+1 \\ 2 + PS(i+1, j-1) \end{cases} \\ a_i \neq a_j \text{ (not same)} \rightarrow \max \begin{cases} PS(i, j-1) \\ PS(i-1, j) \end{cases} \end{cases}$$

Bottom-up :

```
if (n.length == 0)   return 0;
if (n.length == 1)   return 1;
for i = 1 to n
{     PS[i,i] = 1;  }
int incrementor = 1;
while (incrementor < n)
{
    for i = 1 to n
    {   j = i + incrementor;
        if (ai == aj)
        {  if (j = i+1)   PS[i][j]=2 ;
           else   PS[i][j] = PS[i+1][j-1] + 2;
        }
        else
        {  PS[i][j] = max( PS[i,j-1] , PS[i-1][j]);}
```

i, j-1	i, j
i+1, j-1	i-1, j

1, 2
2, 3
3, 4
4, 5
5, 6

①	②	③	④	⑤	⑥
1	2	3	4	2	1
1	2	4	3	2	1

(2) |6-2|+1 = 5 ✓

1 2 3 4 5
5 4 3 2 1

(5) 5-5+1 = 1 ✓

0	1	2	4	2	1	3
3	1	2	4	2	1	0

(7-2)+1 = 6

Only insert gives +1.
~~Change~~ & delete does not give +1.

1 2 3 6 4 5 4 3 7 2 1 0
i j

Base cases:
string is of length 0 so no palindrome.
String is of length 1 so a[1] is the palindrome max length.

length 2:

a	a

P=2

or

a	b

P=1

length 3:

a	d	b

2
P=2

a	a	a

2
P=3

a	b	a

1
P=2

a	b	b

1
P=2

a	a	c

2 P=2

a	b	c

1 = 1

length 4:

a	a	b	a

2
P=3

a	a	b	b

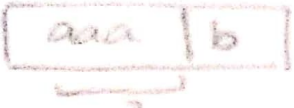
2
P=2

a	a	b	d

2
P=2

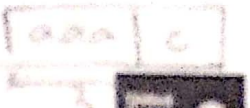
3

P = 4



3

P = 3



3

P = 3

g t 7 6 8 1 2 3 4 3 2 X 9

123421

P(1,6)

12342

P(1,5)

2342

P(2,5)

23421

P(2,6)

1234

P(1,4)

231

P(2,4)

P(3,3)

4

P(1,3)

34

P(2,3)

P(1,2)

P(2,2)

P(3,3)

P(1,1)

P(2,2)

0

9

0 1

9 1

0 1

9 1

1

1

① Part of longest sequence

② Part of current longest

③ Not part of sequence

i		0	1	2	4	2	1	3
	0	1	1	1	1	1	5	5
	1		1	1	1	3	5	5
	2			1	1	3	3	1
	4				1	1	1	1
	2					1	1	1
	1						1	1
	3							1

(Q8):

Input : $A = [a_1, \ldots, a_j, \ldots, a_n]$.

Cases :

Current element a_i forms part of sum $\Rightarrow M_j + a_i$

Current " " does not form " " " $\Rightarrow$ previous $\max M_i$ remains or a_i is negative so we add 0.

So we get :

Max Sum until index $i = MSS_i - 0$

Max Sum overall out of all MSS_i's calculated so far $= MSS = 0$;

$$MSS_i = \begin{cases} MSS_i + a_i \\ 0 \quad \text{if } a_i + MSS_i < 0 \quad (\text{negative}) \end{cases}$$

$$MSS = \max \begin{cases} MSS_i \\ MSS \ (\text{current value}). \end{cases}$$

Algorithm:

```
Find MSS ( a1 .... an ) {
int MSS , MSSi = 0 ;
for i = 1 to n
{   if ( MSSi + ai ≥ 0) then MSSi = MSSi + ai ;
        else MSSi = 0 ;
        MSS = max ( MSS , MSSi ) ;
}
return MSS ; // question says to just give sum.
}
```


Marks _____ Signature of Instructor _____

$l = i \quad \dots \quad k$

Time complexity : $O(n)$

↳ only one for-loop running from 1 to n.

Space complexity : $O(1)$

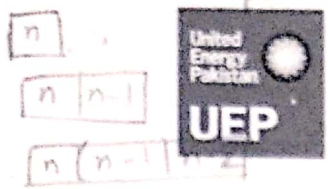
↳ Only 2 constant variables required ; MSS & MSS; so constant space complexity.

Working shown as proof of concept: (Shown on next pages)

Input : $a_1 \dots a_k \dots a_n = a[]$

Time & space complexities

Brute force : $O(n^2)$



$a_1 \dots a_j \dots a_i \dots a_n$

$MSS[1] = a_1$

$MSS[i] = a_{i+1}$ where $1 \leq i \leq n-1$ $MSS_1 \dots MSS_i$

$a_{2 \dots n}$

$MSS[i] < 0 = a_i$

A

1	2	3	4	5	6	7	8
-2	-3	4	-1	-2	1	5	-3

MSS

1	2	3	4	5	6	7	8
-2	-3	1	-1	-2	1	6	2

2 → 6 → 8 → n

$MSS[i]$ — $(a[i] + MSS[i-1]) > 0$ so $a[i] + MSS[i-1]$

$a[i]$.

	0	1	2	3	4	5	6	7
max_end	0	0	4	3	1	2	7	4
max_so_far	0	0	4	4	4	4	7	7

$$M_j = \begin{cases} M_j + a_j \\ 0 \quad \text{if } M_j + a_j < 0 \end{cases}$$

$$MSS = \max \begin{cases} MSS \\ M_j \end{cases}$$

```
{
Mj = 0 , MSS = 0

for i = 1 to n
{
    if ( (Mj + ai) > 0)
        Mj = Mj + ai ;
    else
        Mj = 0 ;

    MSS = max (MSS , Mj) ;
}
return MSS;
}
```

$O(n)$ = Time $O(1)$ = space

(Q7).

Input : m×n grid matrix.

Idea : Calculate column aggregate sum from column i to j $\forall$ j $i \leq j \leq n$. Get maximum subarray sum of each aggregate combination and choose max of these to get max aggregate of entire matrix i.e. max subgrid sum. (Note; question asks only for max sum, not subgrid coordinates).

```
Find MSG ( A [m x n] ) {
    Column array = int [1][m]; // already initialized to zeros.
int MSG = 0;
for  i = 1  to  n    // current column
{    for  j = i  to  n   // column to be added to this column.
     {    for  r = 1  to  m  // rows
          {  Column Array [r] += M [r][j] ; }
          MSG = max ( MSG , MaxSubArraySum ( Column Array ) ) ; // Kada ne takes O(m)
     }
     for k=1 to m Column Array [1 ... m] = [0, 0, ... , 0] // reinitialized to zeros using for loop.
}
return MSG;
}
```

Institute of Business Administration Karachi
and Ideas for Tomorrow

Quiz Paper

Full Name: _____

Student ERP I/D No: _____

Subject: _____

Quiz #: _____ Date : _____

Class/Section: _____

Teacher's Name: _____

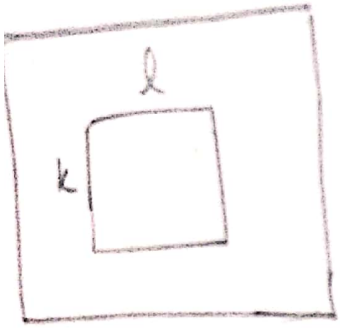
Time complexity: $\max(O(n \times n \times m), O(m \times m \times m))$

- $O(n \times n \times m)$
  - for i to n (no. of columns)
  - for j = 1 to n (runs max of n times)
  - for r = 1 to m or Kadane (both run O(m) time)
- $O(m \times m \times m)$
  - same as previous
  - same as previously stated.
  - Re-initializing column array to 0

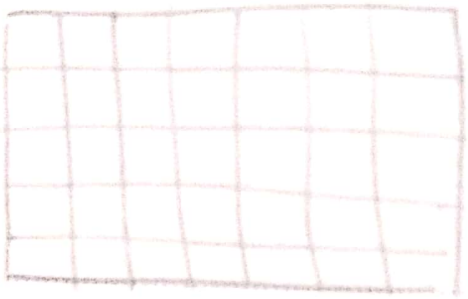
Space complexity: $O(m)$

↳ Column Array has size m and MSG has constant size.

Working: (as proof of concept; shown on next pages →)



Brute force : $O(N^6)$

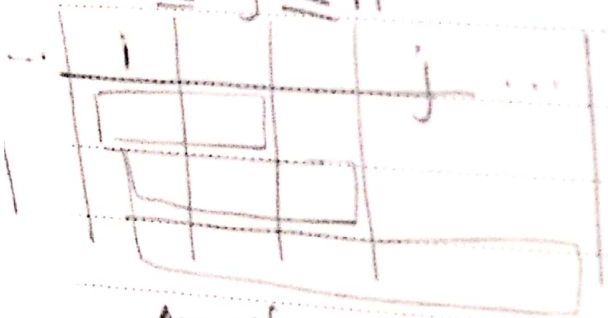


Time = $O(N^3)$

Space = $O(1)$

$l \le i < j \le n$

$l = i \ldots k$



Array of columnsums $[1 \ldots n] = [0, 0, \ldots, 0]$

```
for i = 1 to n
{   // initialize array of columns with column i values.
    for j = i to n
    {
        for r = 1 to m
        { Array of columnsums[r] += M[r][j] }
        //use Kadanes now
        MSG = max(MSG, Kadane(Array of columnsums)); → O(n)
    }
}
```

9	7	8	3	7	34
-6	-2	-1	2	-4	-11
-7	5	5	2	-6	-1
3	2	9	-5	1	20
-1	12	21	12	-2	

-1	11	32	44	42	= (44)
	12	33	45	43	= (45)
		21	33	31	= (45)
			12	10	= (45)
				-2	(45)

20	19	8	(42)
	-1	-12	22
		-11	23
			34

34	23	22	(42)	42
	-11	-12	8	
		-1	19	
			20	

(Q9) Input: $\langle s_1, s_2, \ldots, s_n \rangle$, t

Cases:

n.length == 0 so no subset exists

n.length == 1 : then $s_n == t$ so return True

else $s_n != t$ so return False.

Each s_i either belongs to current valid subset and $t = t - s_i$

or does not belong to subset and $t = t$.

or it belongs to another possible valid subset so both cases possible.

This gives:

$$TSS(i,t) = \begin{cases} \text{True} & t == 0 \\ TSS(i-1, t) & s_i > t \\ TSS(i-1, t - s_i) & \text{keeping } s_i \\ \underline{OR}\ TSS(i-1, t) & \text{not keeping } s_i \end{cases}$$

Algorithm :

```
Find TSS ( s1 .... sn , t ) {
    if  n == 0    return  print ( No subset possible );
    else if  t == 0    return  True and print empty subset;
else {
    TSS = [n+1][t+1];
for i = 0 to n
    TSS [i][0] = True;
for s = 0 to t
    TSS [0][s] = False;

for i = 1 to n
{   for T = 1 to t
    {   if ( si > T )  then  TSS [i-1][T] = TSS[i-1][T];
        else
            TSS[i][T] = TSS[i-1][ T - si ]  ||  TSS [i-1][T];
    }
}
// T-F table made.
```

// (for getting subset).

Algorithm:

```
int i = n , s = t ;        Subset = new Stack ;
TSS = int [n][t];
while ( i ≠ 0  ||  s ≠ 0 )
{
    if (TSS[i][s] == T && TSS[i-1][s-i] == T)
    {   Subset.push (s,) ;
        i = i-1 ;
        s = s - i ;
    }
    else
        i = i - 1 ;
}
return Subset ;
} //end
```

Time complexity : O (n x t)

↳ Traversing nxt TSS matrix ..

Space complexity :

↳ making nxt matrix.

Working (shown as proof of concept on next page →) :

Input : $a_1 \dots a_n$, TargetSum. Find subset

so no subset exists.

n.length == 0

n.length = 1 a_1 == Target Sum < True → a[1]
Else → return no subset exists

$a_1 \dots a_n$

Target Sum, $a_1 \dots a_{n-1}$ $a_1 \dots a_{n-1}$, Target Sum $- a_{n-1}$

Target Sum, $a_1 \dots a_{n-2}$ Target Sum $- a_{n-1}$, $a_1 \dots a_{n-2}$

	1	3	9		
	2	4	7	4	9
	3	5	3	5	5
T	4	2	3	TSS(i, target)	

	1	2	3	4
a	3	4	5	2

TSS(i, target) =
- True — Target = 0
- TSS(i-1, target) $a_i >$ tar
- TSS(i-1, target - sum) or TSS(i-1, target)

	0	1	2	3	4
	0	3	4	5	2
0	T	T	T	T	T
$i \geq a_{i-1}$	F				
2	F				
3	F				
4	F				
5	F				
6	F				
7	F				
8	F				
9	F				

i

S

i \	0	1	2	3	4	5	6	7	8	9
0	T	F	F	F	F	F	F	F	F	F
3	T	F	F	T	F	F	F	F	F	F
4	T	F	F	T	F	F	F	T	F	F
5	T	F	F	T	T	T	F	T	T	T
2	T	F	T	T	T	T	T	T	T	T

Diagonal = include $3+4+2 = 9$

Up = not include $4+5 = 9$

O (sum * n) = time & space both.

→ only traverse if there exists true at end.

```
i = n , S = target
while ( i != 0 && S != 0 )
{
    if TSS[i][S] == T && TSS[i-1][S-i] == T
    {
        stack.push(a[i]).
        i = i-1 , S = S-i
    }
    else
    { i = i-1. }
}
return stack.
```