CS & IT ENGINEERING

Data structure and Programming

Arrays

Lec- 01

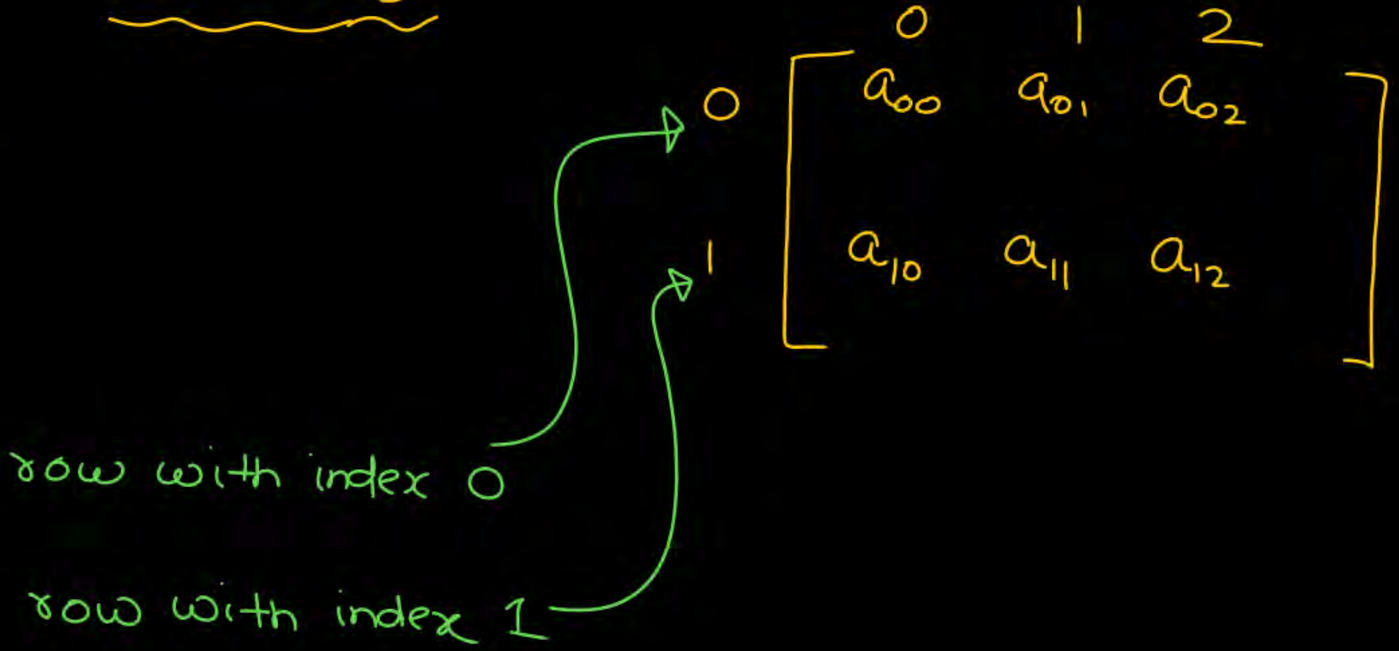
By- Pankaj Sharma sir

Arrays-1

C-concept

2-D array

```
int a[2][3];
```

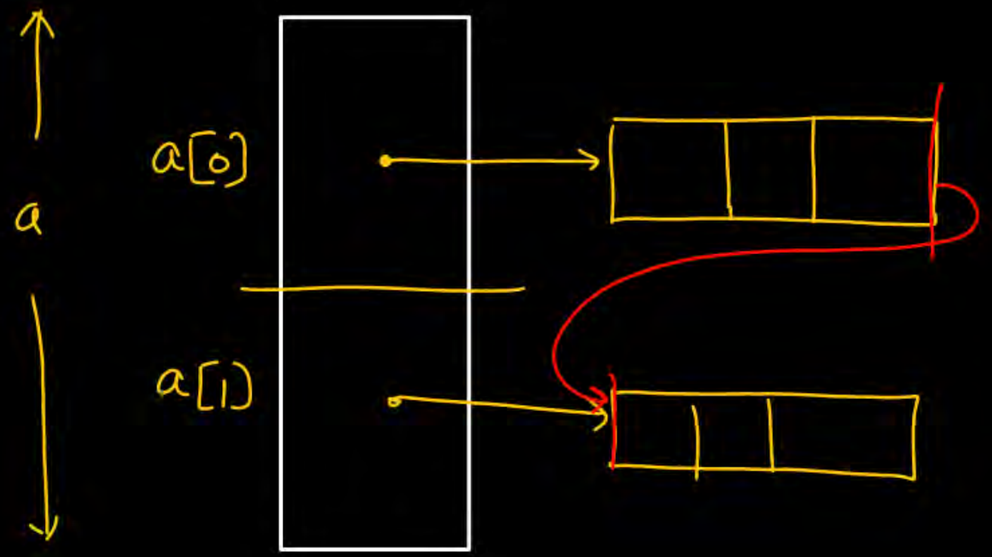


# of elements in row with index 0 = 3

# of elements in row with index 1 = 3

Every row, the no of elements = 3 = no of col.

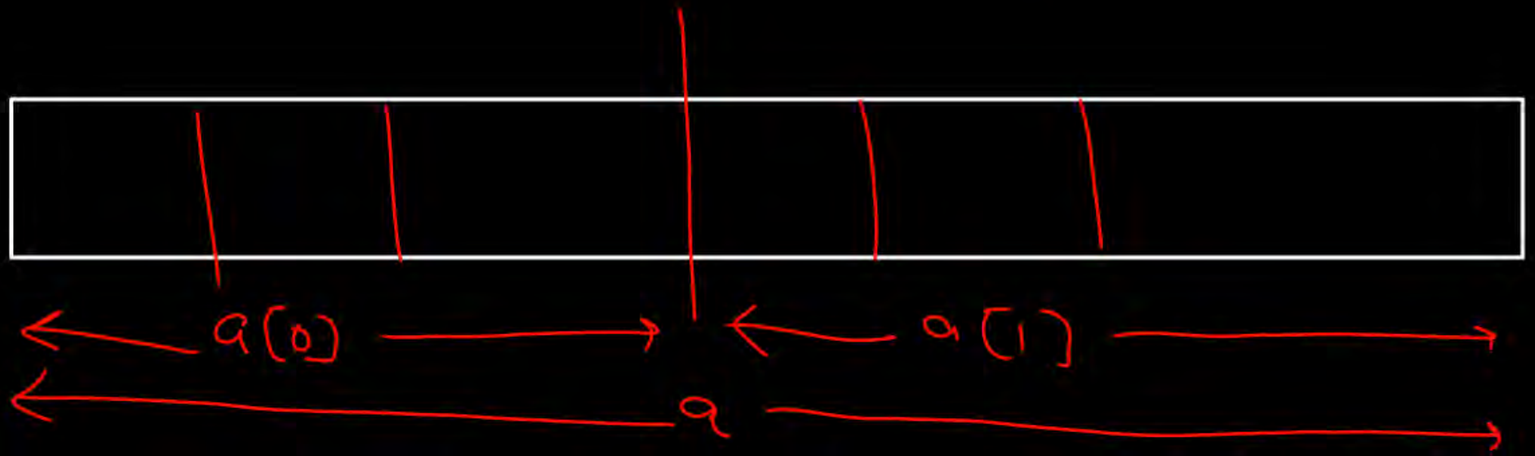
int a[2][3];

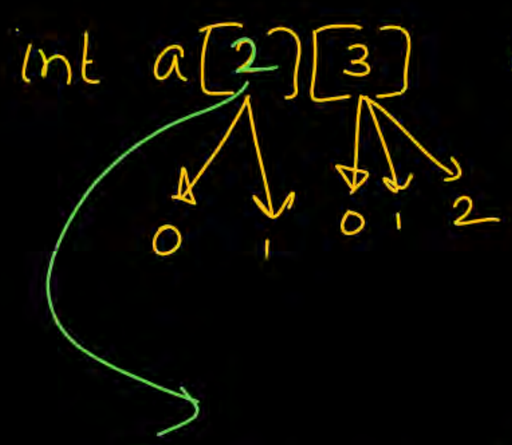


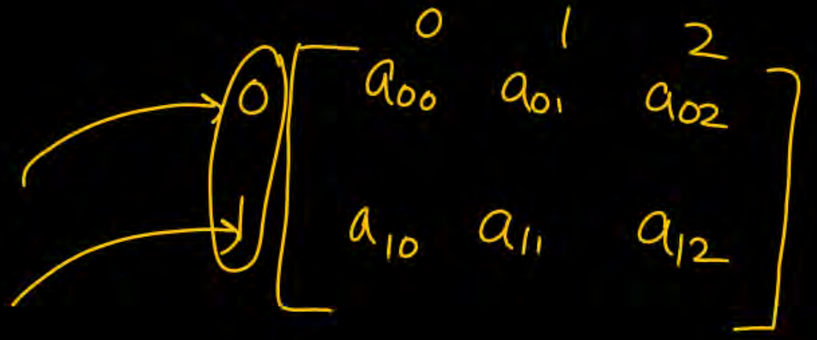
	0	1	2
0	a_{00}	a_{01}	a_{02}
1	a_{10}	a_{11}	a_{12}

row a[0] ⇒ 3

row a[1] ⇒ 3



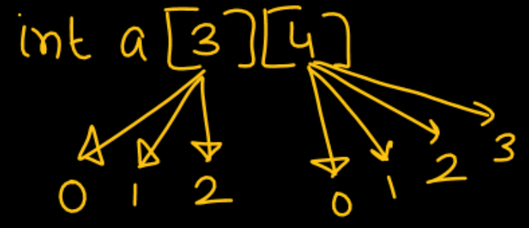




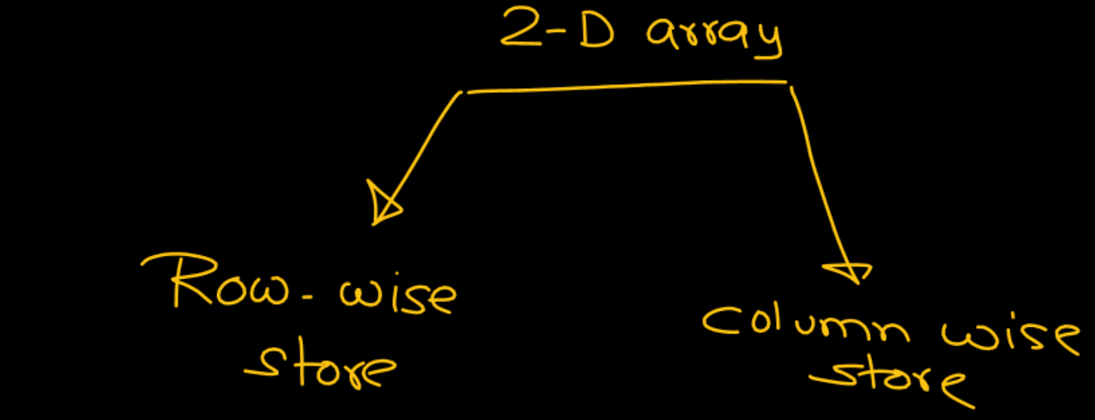
Each index/no. in this dim. represent 3 element

a[0] — 3

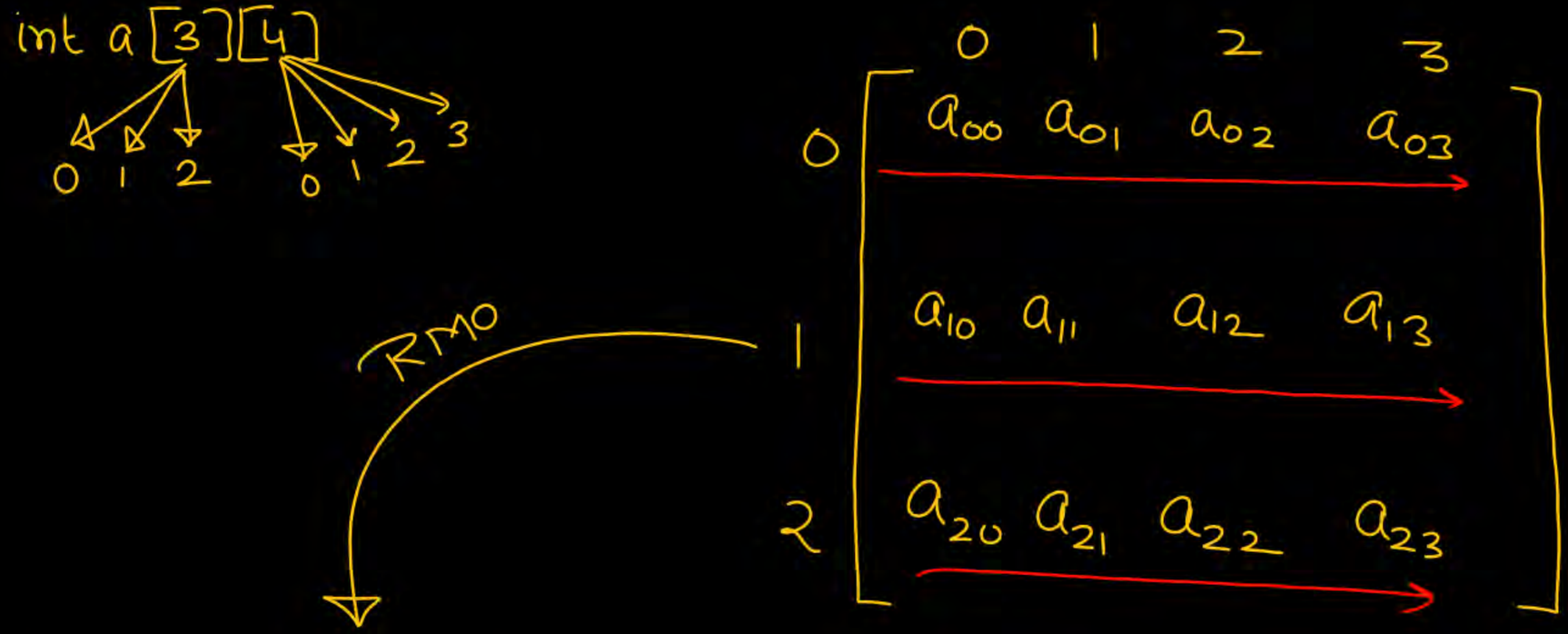
a[1] — 3

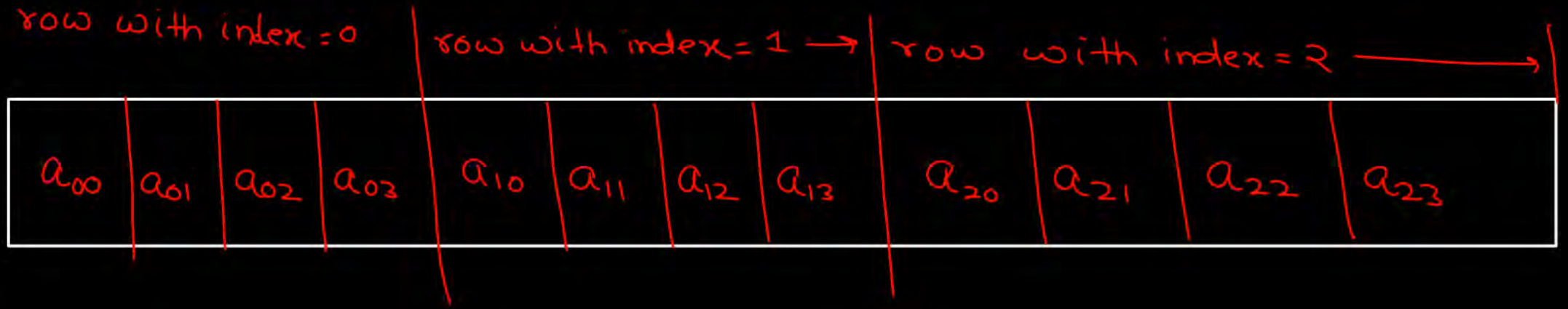


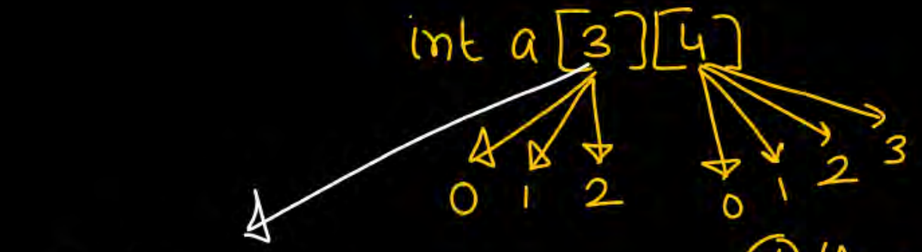
$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{bmatrix} \end{array}$$



Row-major Order Column major order

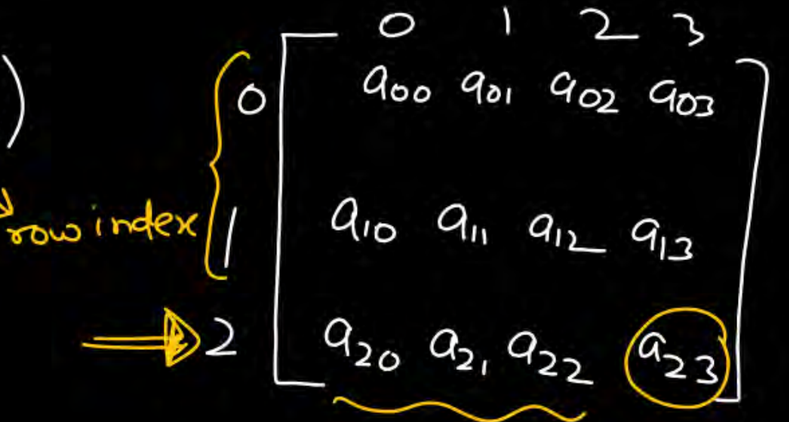

int a[3][4]
0 1 2
0 1 2 3
RMO
0 1 2 3
0 a00 a01 a02 a03
1 a10 a11 a12 a13
2 a20 a21 a22 a23


row with index = 0
row with index = 1 →
row with index = 2 →
a00 a01 a02 a03 a10 a11 a12 a13 a20 a21 a22 a23



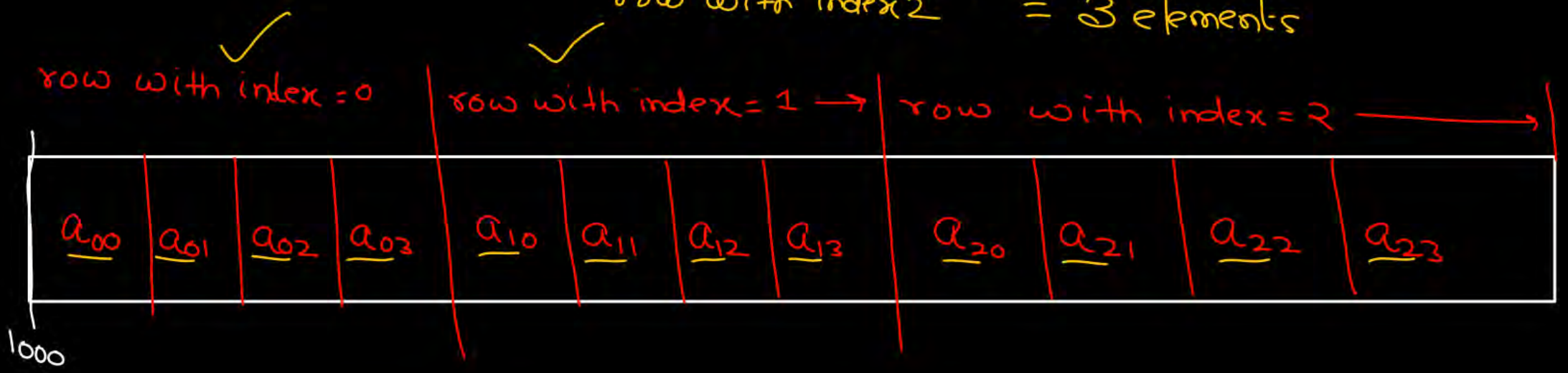
E2 index = 4 element

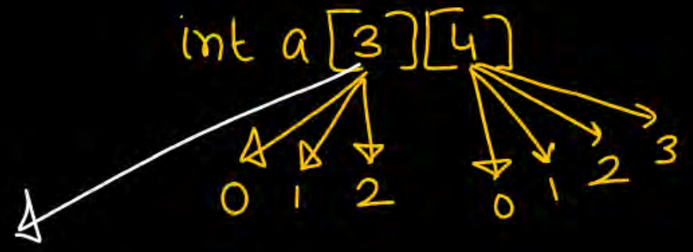
add(a_{23})



① How many rows already filled before row with index 2 = 2 rows

② No. of elements already filled before a_{23} in row with index 2 = 3 elements





add(a_{23}) → row index

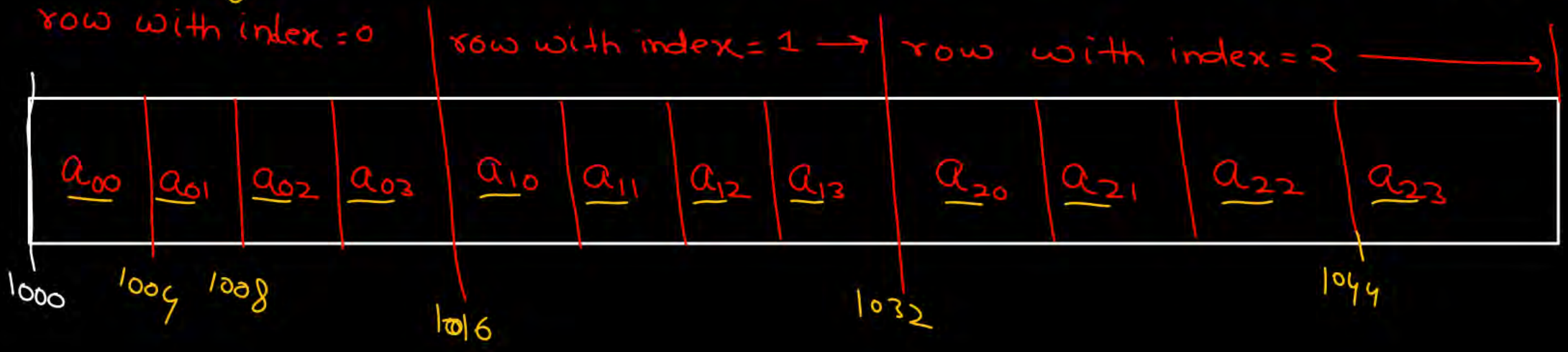
	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
→ 2	a_{20}	a_{21}	a_{22}	a_{23}

2 Rows ⇒ 2×4

3 elements ⇒ + 3

11 elements

No. of elem. already filled before a_{23} = 2×4 + 3 = 11 elements ✓

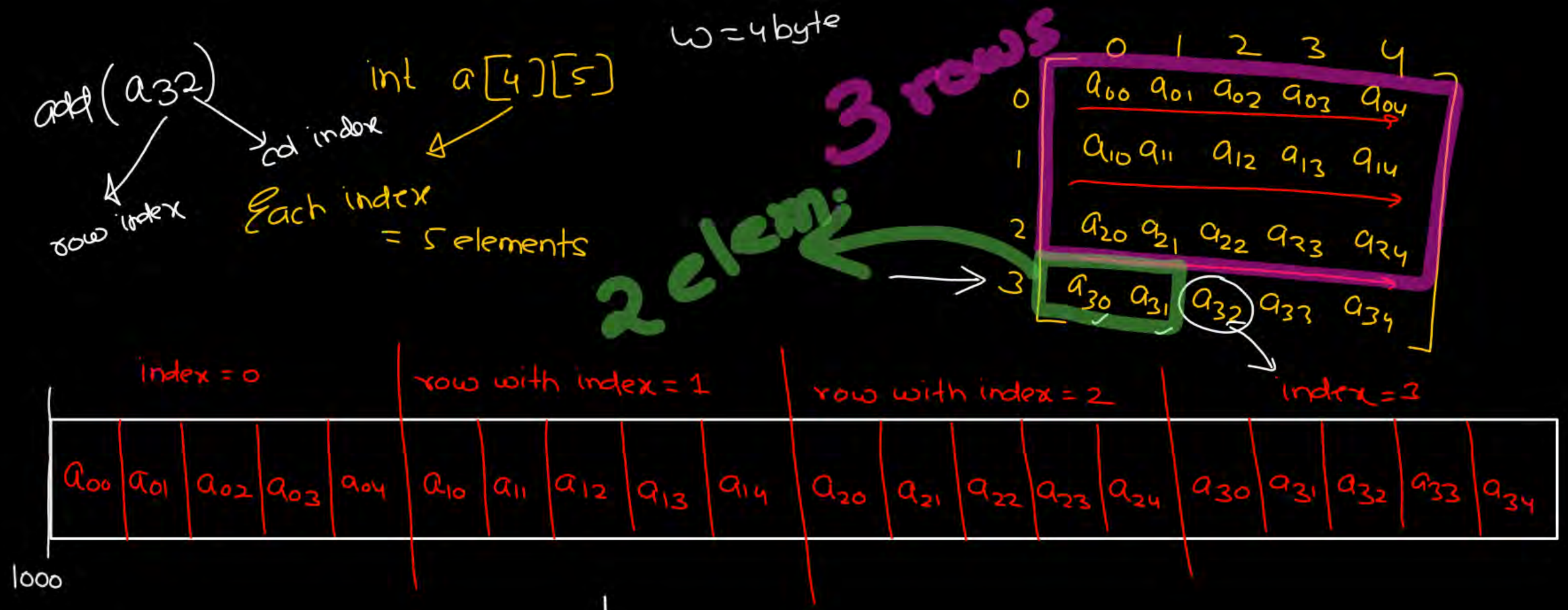


$\omega = 4$ bytes

Memory already filled before $a_{23} = 11 \times 4 = 44$ bytes

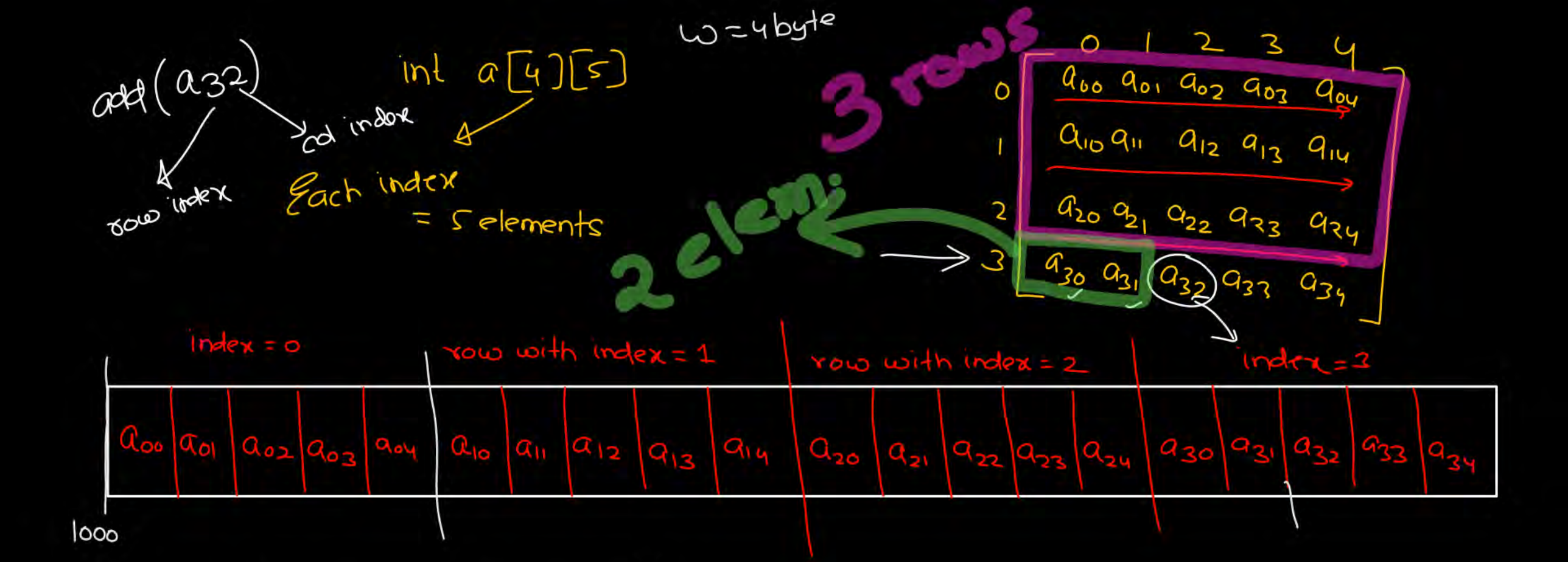
$1000 + 44$

$= 1044$



① How many row already filled befor row with index 3 = row with index 0 to 2
= 2 - 0 + 1 = 3 rows

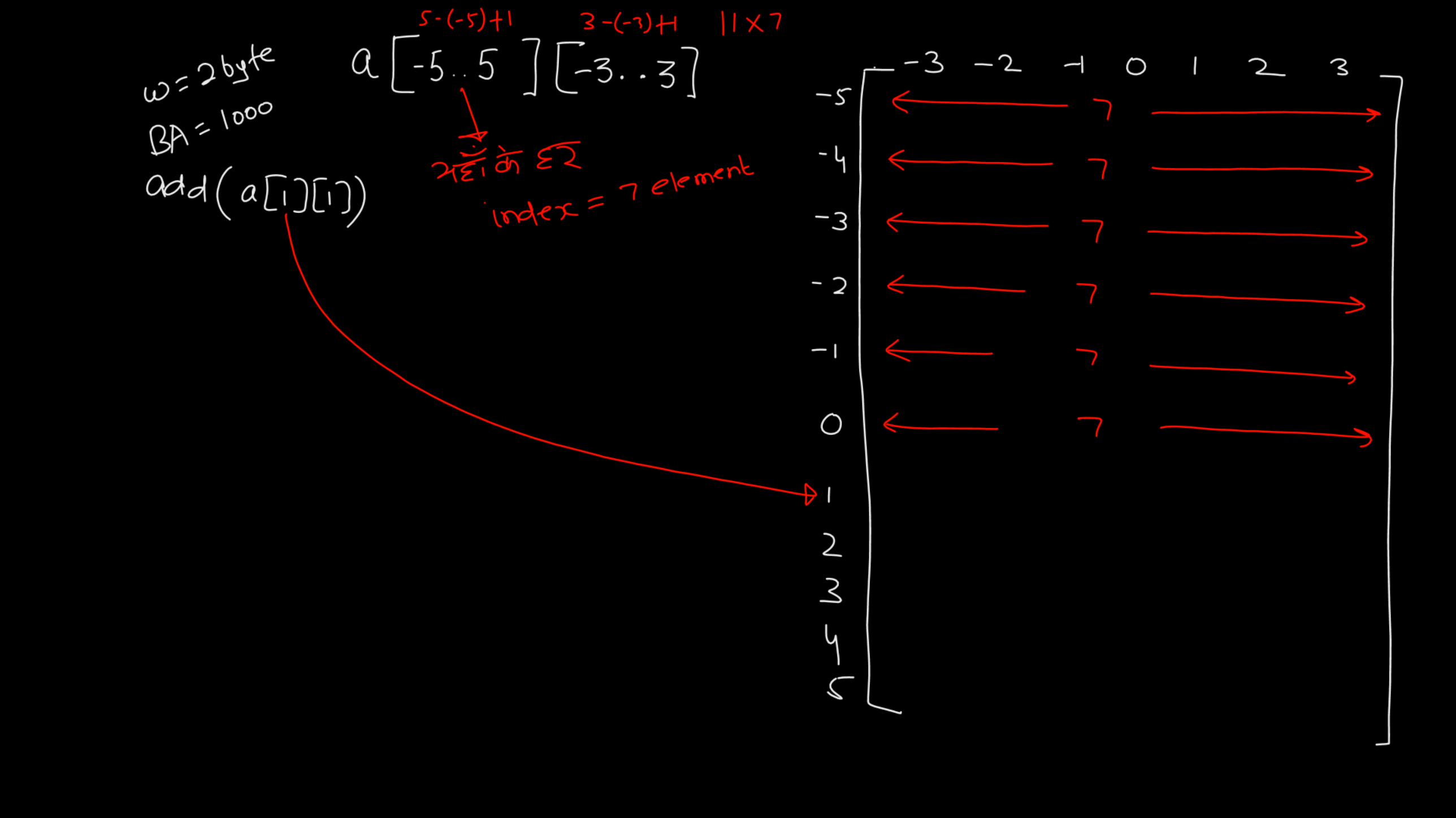
② within row whose index is 3, the no. of elem. already filled befor a_{32} = col with index 0 to 1
= 1 - 0 + 1
= 2 elements



before a_{32} ⇒ 3 rows & 2 ele already filled

⇒ $3 \times 5 + 2 = 17$ elements are already filled

Memory already filled before a_{32} = $17 \times 4 = 68$ bytes


5-(-5)+1
3-(-3)+1
11 X 7
w = 2 byte
BA = 1000
a [-5..5] [-3..3]
add(a[1][1])
यहाँ को ER
index = 7 element
-3 -2 -1 0 1 2 3
-5
-4
-3
-2
-1
0
1
2
3
4
5
7

5-(-5)+1 3-(-3)+1 11×7

w = 2 byte

BA = 1000

add(a[1][1])

a [-5..5] [-3..3]

यहाँ के ER

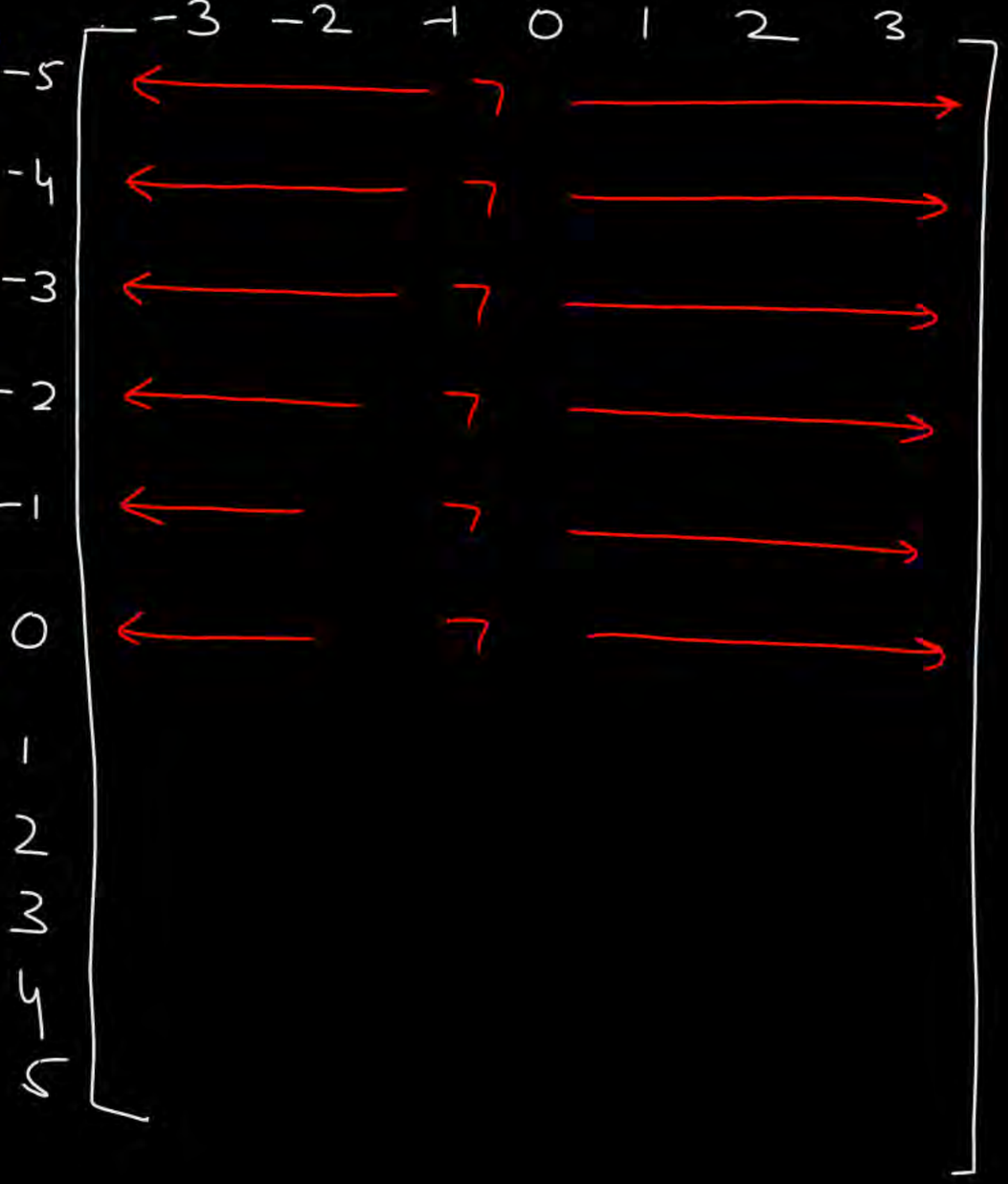
index = 7 element

rows already filled

= -5 to 0

= 0-(-5)+1

= 6 rows



5-(-5)+1 3-(-3)+1 11×7

w = 2 byte

BA = 1000

a[-5..5][-3..3]

add(a[1][1])

यहाँ के ER

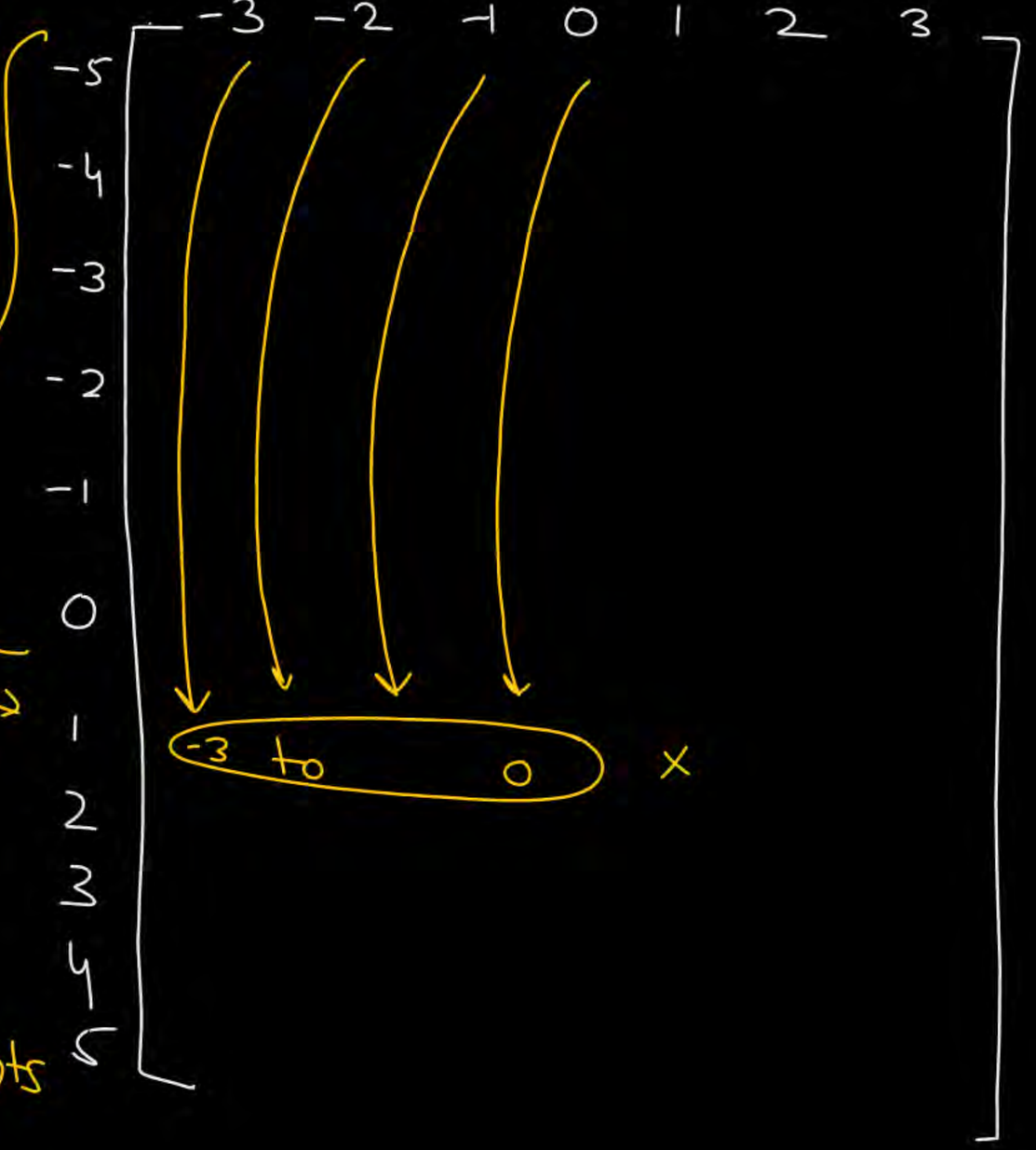
index = 7 element

Within row with index 1

Elements already filled before a[1][1]

= col. with index -3 to 0

0-(-3)+1 = 4 elements



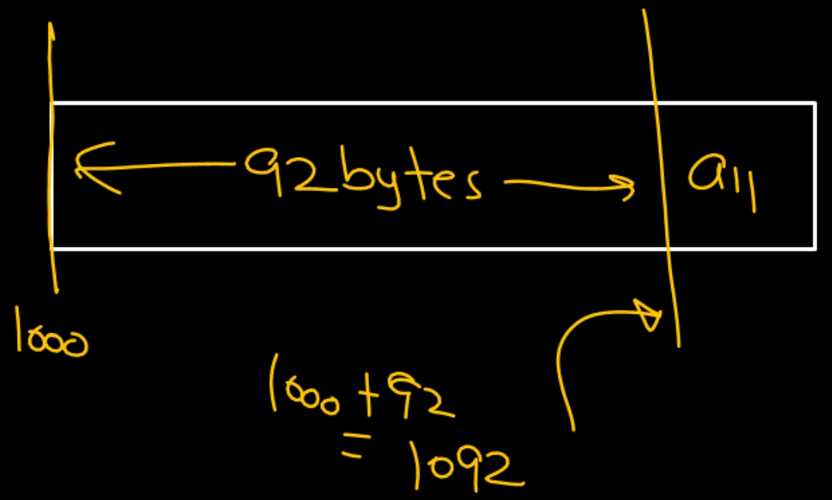
After 6 rows & 4 elements a_{11} is stored

Elem. already filled before a_{11} = $6 \times 7 + 4$

= 46 elements

Memory already filled before a_{11} = 46×2

= 92 bytes



Q RMO

$5-(-5)+1 = 11$ $\quad 7-(-7)+1$

a[-5..5][-7..7]

Each index/No. in this dim ⇒ 15 elem.

w = 4 bytes

BA = 1000

add(a[1][3])

How many index already filled in this dim

= -5 to 0

= 0-(-5)+1

= 6

⇓

6 X 15

-7 to 2

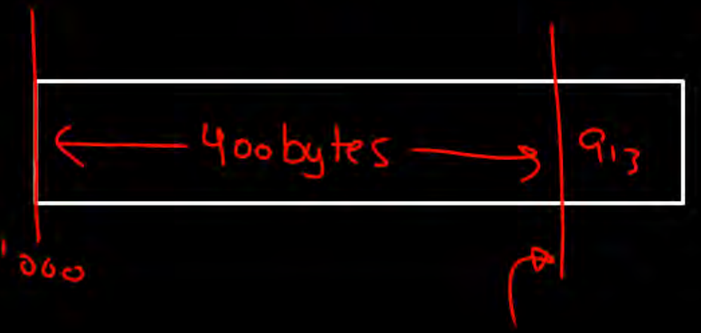
2-(-7)+1

= 10 elem

Total element already filled = 6 X 15 + 10

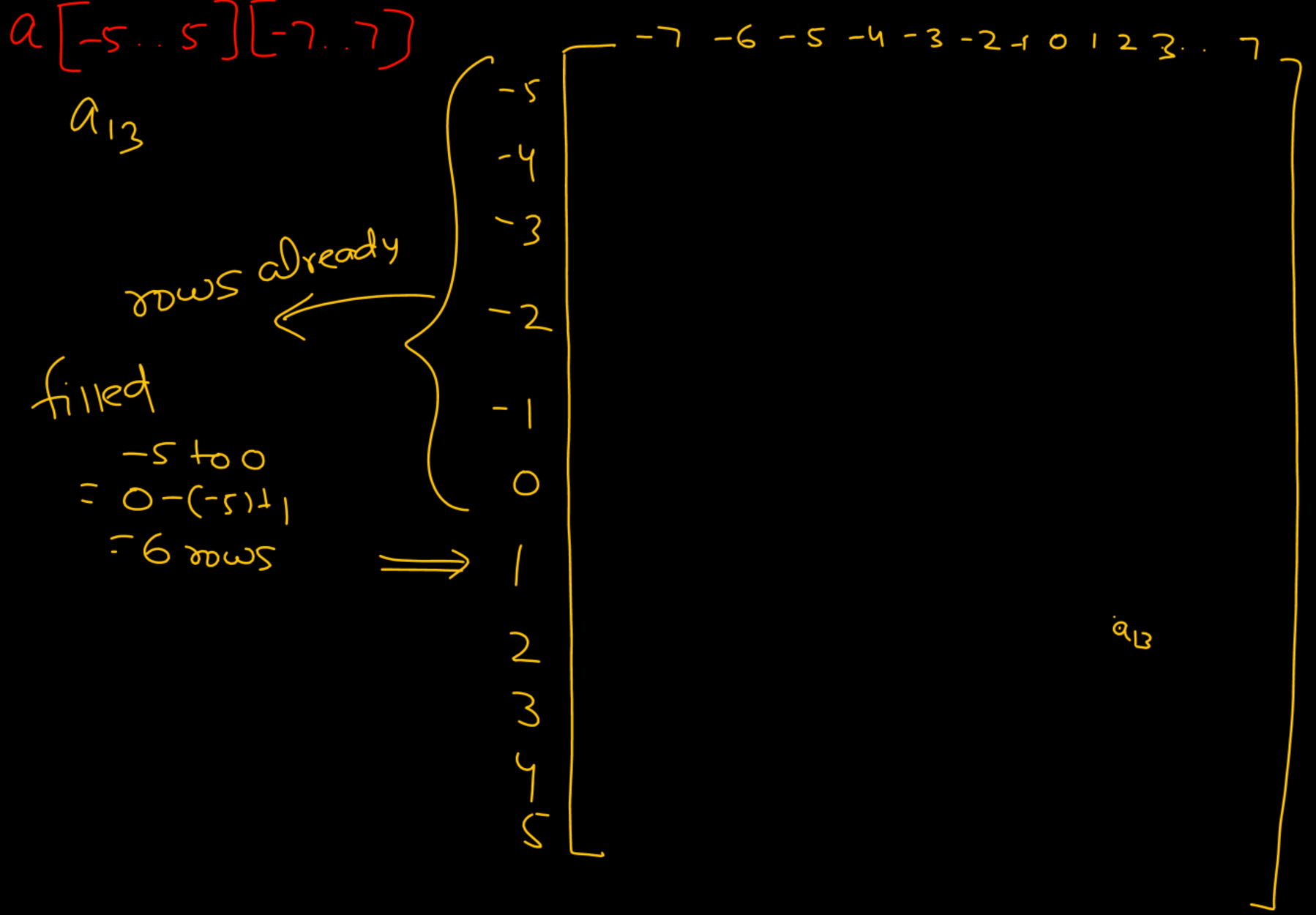
= 100 elem.

Memory already filled = 100 X 4 = 400 byte



1000 + 400

= 1400


a[-5..5][-7..7]
a_{13}
rows already filled
-5 to 0
= 0-(-5)+1
= 6 rows
-7 -6 -5 -4 -3 -2 -1 0 1 2 3 .. 7
-5
-4
-3
-2
-1
0
1
2
3
4
5
a_{13}

a[M][N]

add(a_{ij})

① rows already filled before row with index i = 0 to i-1

= i-1-0+1

= i Rows

⇓

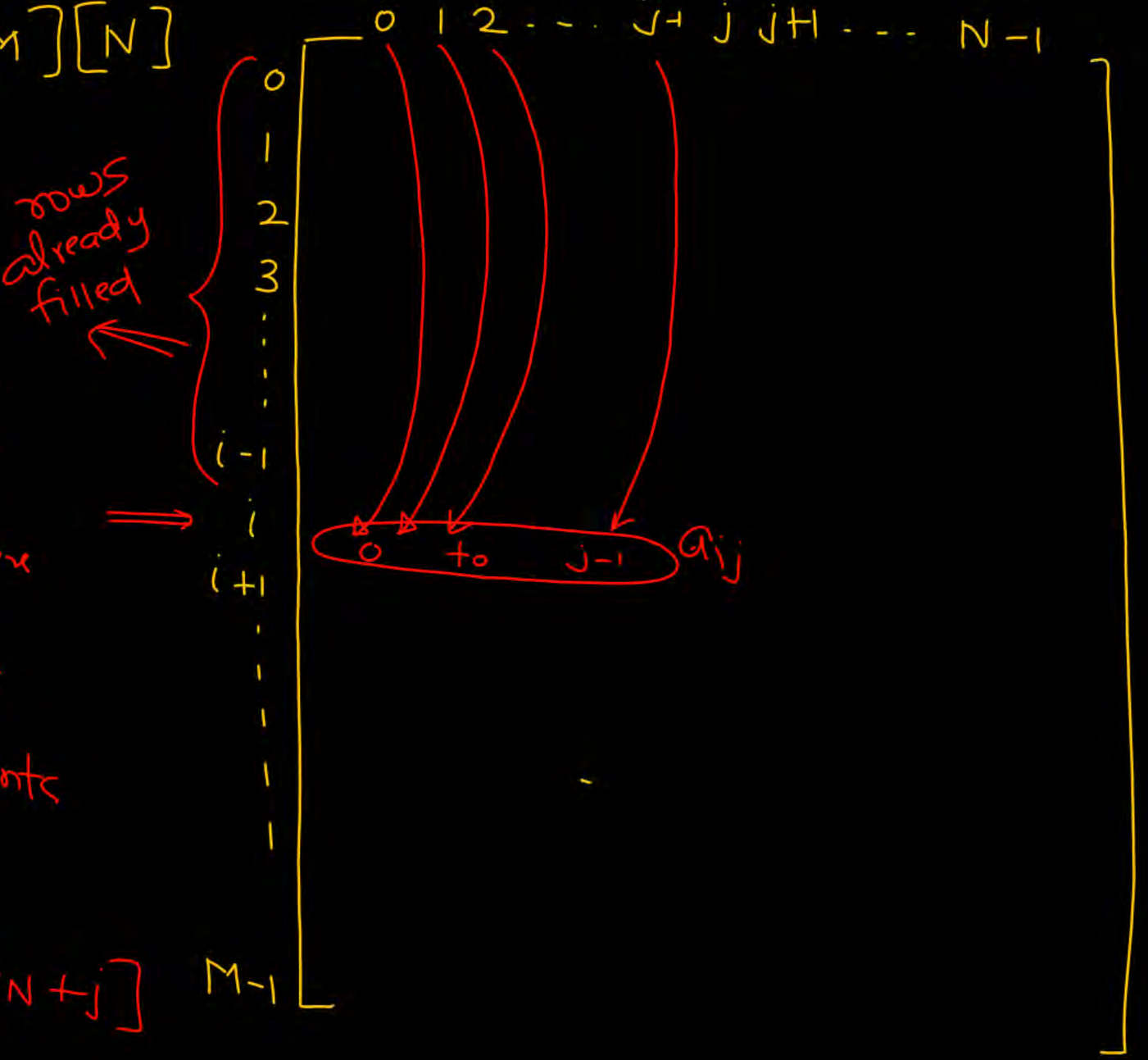
i × N elements

② within ith row, (index) elements already filled before a_{ij} = col with index 0 to j-1

= j-1-0+1

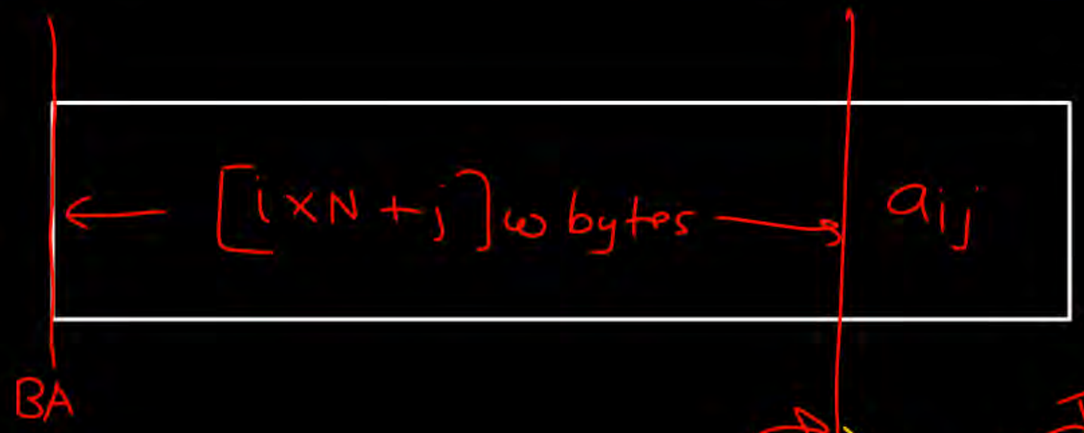
= j elements

Total elem. already filled = [i × N + j]



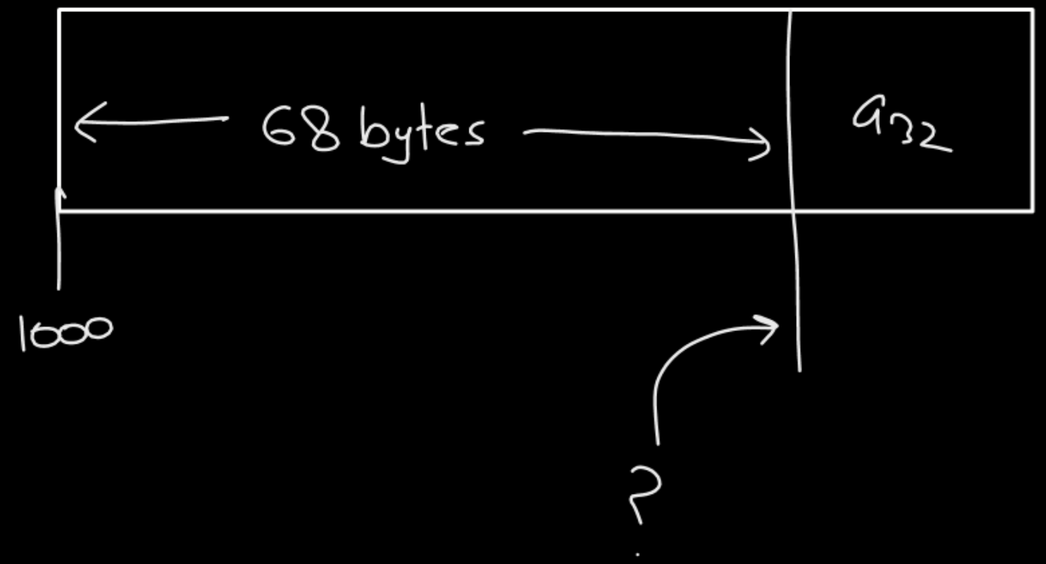
Size of each = ω bytes

Memory already filled = $[i \times N + j] \times \omega$ bytes



$add(a_{ij}) = BA + (i \times N + j) \times \omega$

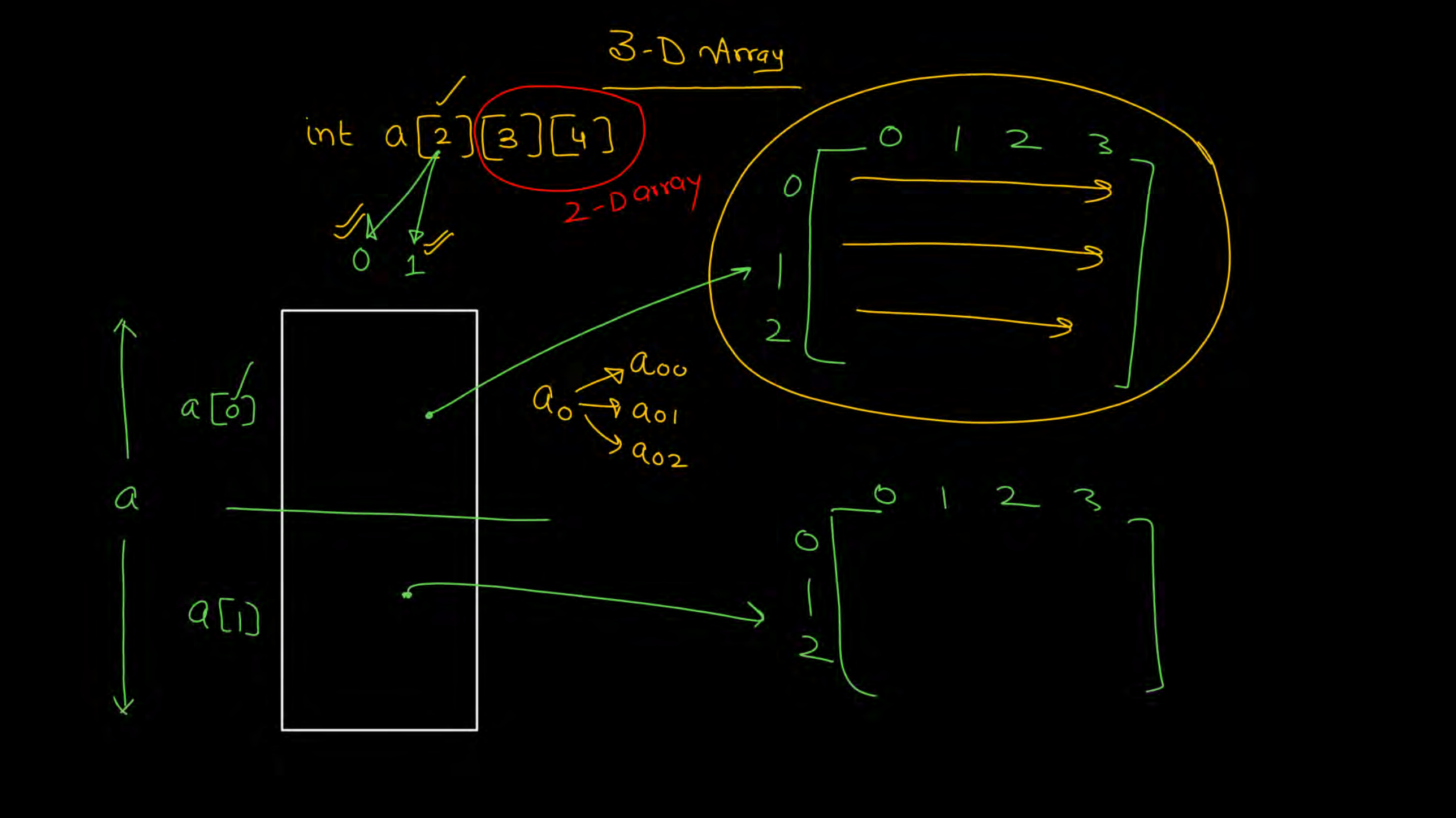
नहीं रहना है

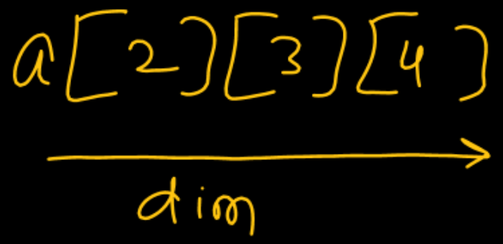


$\Rightarrow 1000 + 68$

$= 1068$

3-D Array




a[2][3][4]
dim

int a[3][4][5]

Every index in this dimension represent = 4 x 5 element

Every index/No. in this dimension represent = 5 elements

Q

RMO

~~INT~~

w = 4 byte

BA = 1000

a [3] [3] [5]

add ($a_{2\ 2\ 3}$)

∈2 index = 3×5

∈2 index = 5

0 to 1

1 - 0 + 1

index already filled = 2

⇓

2×3×5

index filled = 0 to 1

: 1 - 0 + 1

: 2

⇓

2×5

index = 0 to 2

2 - 0 + 1

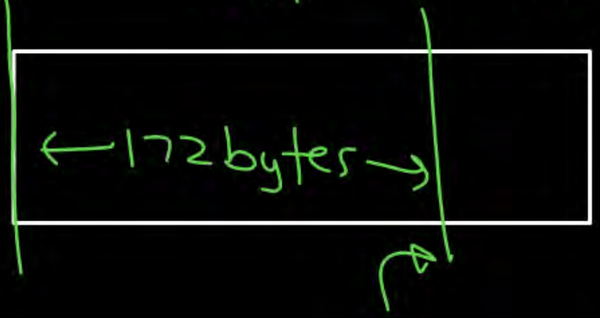
= 3 elements

Total ele = 2×3×5 + 2×5 + 3

= 43 elements

Total memory filled = 43×4 = 172 bytes

1000



1000 + 172

172

w = 2 byte
BA = 1000

$A[-5 \ldots 5][-3 \ldots 3][-5 \ldots 5]$

3-(-3)+1 = 7 ; 5-(-5)+1 = 11

addr(A[0,0,0])

Every index = 7×11

Every index = 11

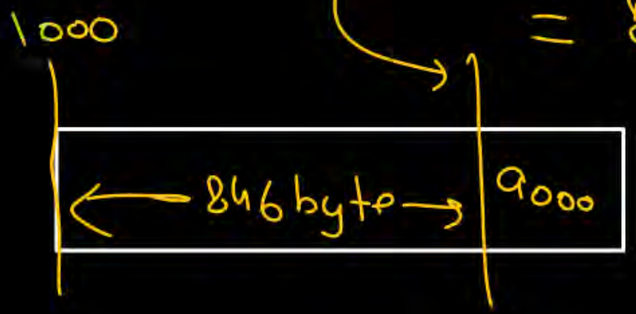
index filled = -5 to -1
= -1-(-5)+1
= 5
⇓
5 × 7 × 11

index filled
= -3 to -1
= -1-(-3)+1
= 3
⇓
3 × 11

inden filled
= -5 to -1
= -1-(-5)+1
= 5

Total elements already filled
= (5 × 7 × 11 + 3 × 11 + 5)
= 423 elements

Memory already filled
= 423 × 2
= 846 bytes



⇒ 1000 + 846
= 1846

w = 2 bytes
BA = 1000

$A[-5..5][-10..10][-3..3][-5..5]$

(21) (7) (11)

add(A_{0213})

21 × 7 × 11 | 7 × 11 | 11

-5 to -1 | -10 to 1 | -3 to 0 | -5 to 2

= -1 - (-5) + 1 = 5

= 1 - (-10) + 1 = 12

0 - (-3) + 1 = 4

= 2 - (-5) + 1 = 8

⇓ ⇓ ⇓

5 × 21 × 7 × 11 elem | 12 × 7 × 11 | 4 × 11

Total elements

= 5 × 21 × 7 × 11
+ 12 × 7 × 11
+ 4 × 11
+ 8

= 105 × 11 × 7 + 88 × 11 + 8

= 9061 elem.

105 × 11 × 7 ⇒ 735 × 11

735 × 11
88 × 11
8

8230
823
9053
+8
9061

823 × 11
+
8

Ex Programmer RMO CMO

Memory = 9061 x 2

=> 18122 bytes

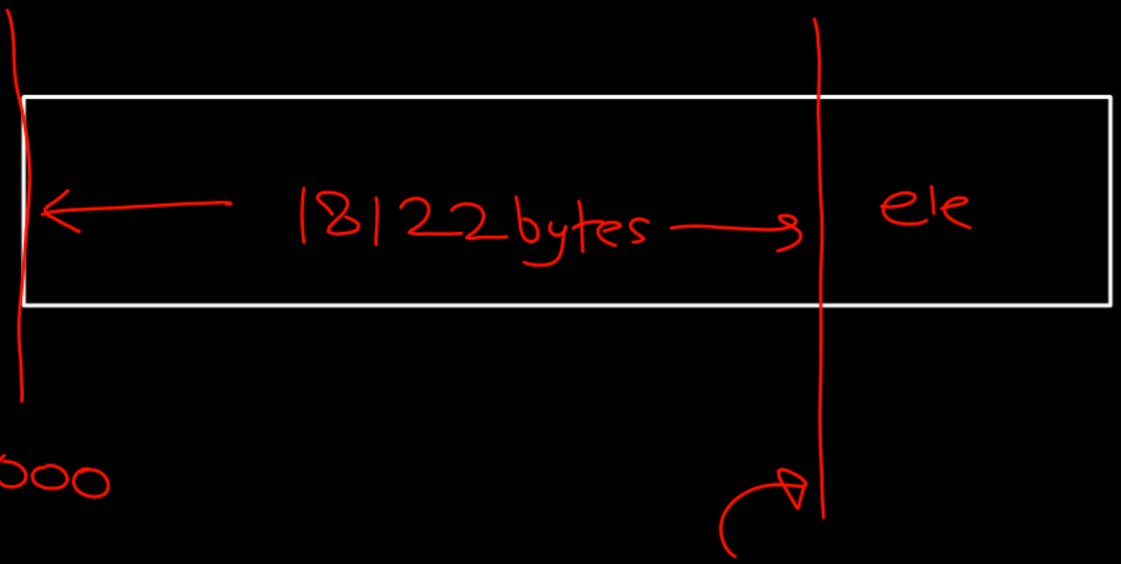
RMO

2-D

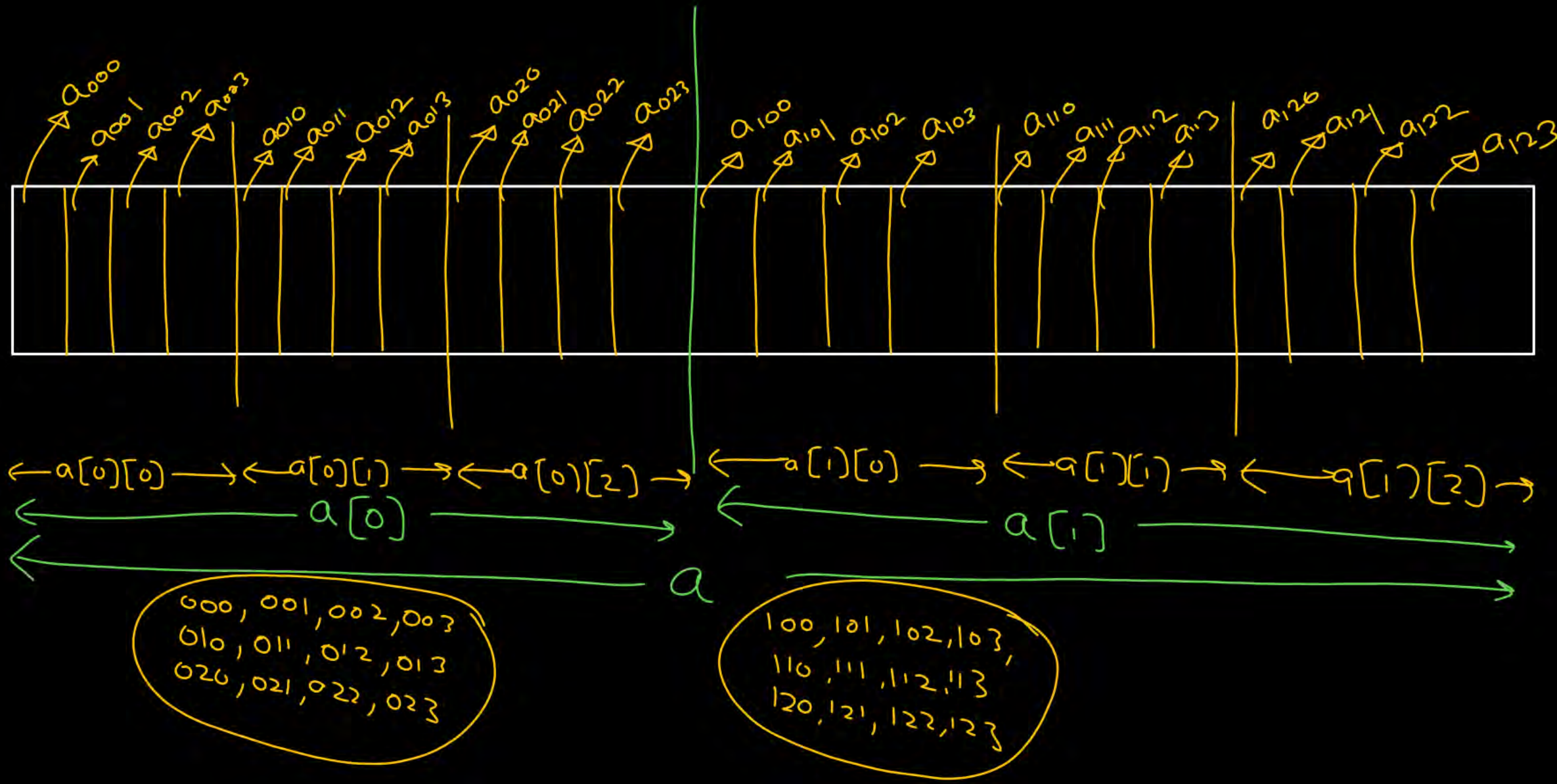
Pagefault

Cache

Hit miss



19122


a000 a001 a002 a003
a010 a011 a012 a013
a020 a021 a022 a023
a100 a101 a102 a103
a110 a111 a112 a113
a120 a121 a122 a123
a[0][0] a[0][1] a[0][2]
a[1][0] a[1][1] a[1][2]
a[0]
a[1]
a
000, 001, 002, 003
010, 011, 012, 013
020, 021, 022, 023
100, 101, 102, 103,
110, 111, 112, 113
120, 121, 122, 123


THANK YOU GW
SOLDIERS !