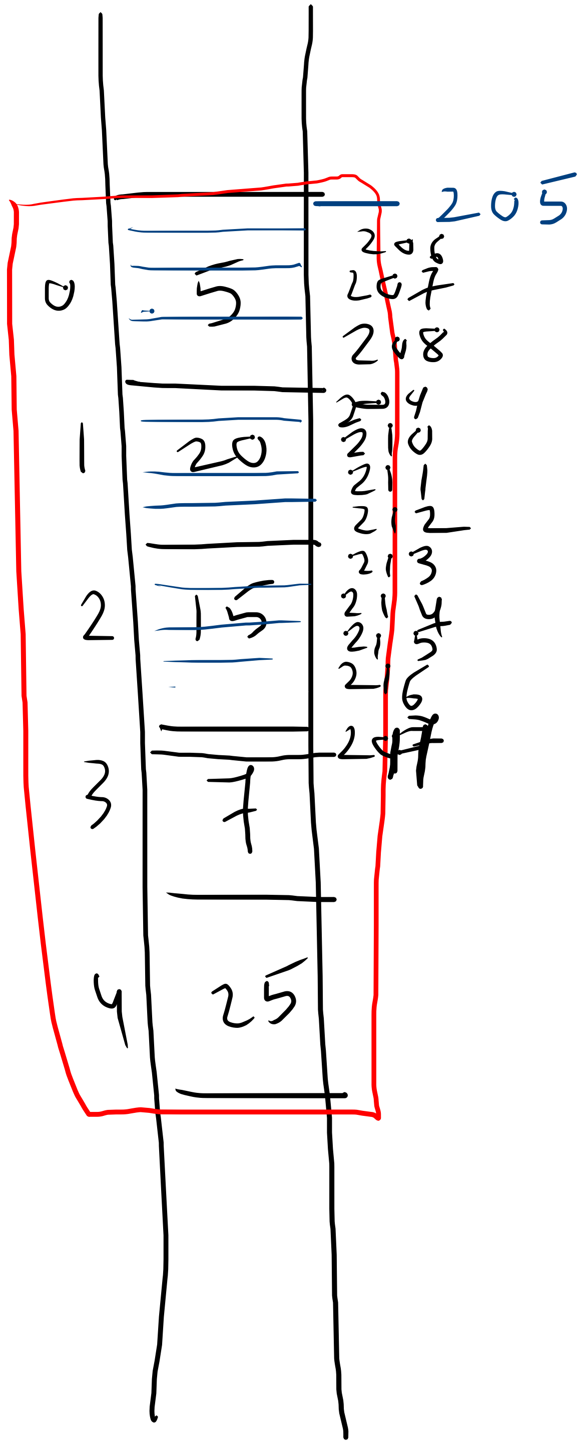
Array

int mark[5] = { 5, 20, 15, 7, 25 };

mark	5	20	15	7	25
	0	1	2	3	4



Formula for calculating address of an element:

i - be an index.

and we have an array A [L .. U]

$$\text{Address}(A[i]) = M + [i - L] * W$$

M ⇓ base address

W — size of each element.

Exm

Address of mark[3] is . $205 + [3 - 0] * 4$

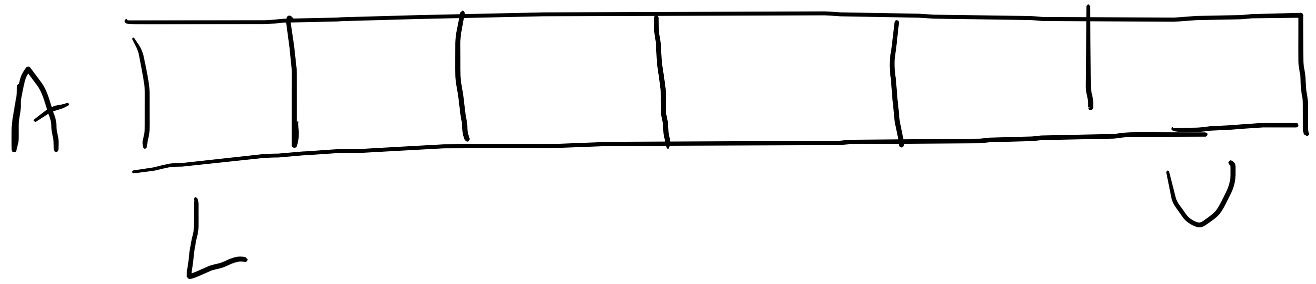
$= 205 + 12$

$= 217$

$L = 0 \rightarrow$ 0-indexing.

$L = 1 \rightarrow$ 1 indexing.

Operations on array



1. Traversing

Traverse (A, L, U)

$i = L$

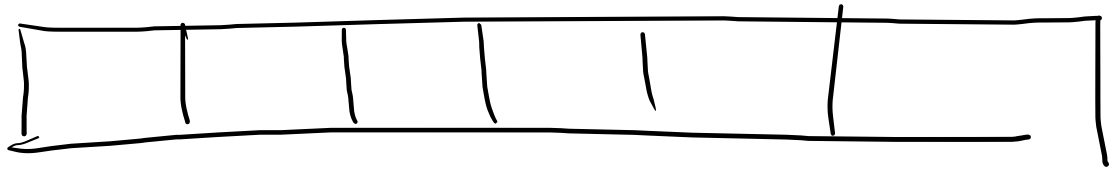
while $i \leq U$

  Process (A[i]) → print, sum,

  $i = i + 1$

$O(n)$

Searching

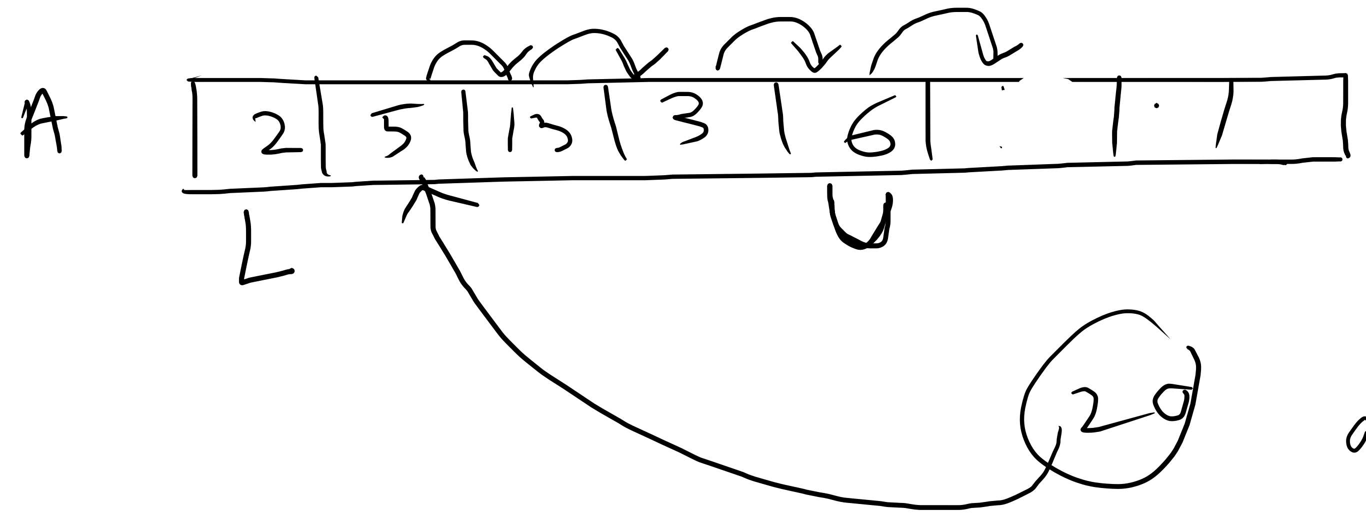


an element that to be searched.

```
Search (A, L, U, item)
i = L, found = 0, K = -1
while i ≤ U and found = 0
        if compare (A[i], item)
            found = 1, K = i
        else
            i = i + 1
if found = 0
        Print "not found"
else
        Print "found"
return K.
```

$O(n)$

Insert



at position L+1

Insert (A, L, U, K, item)

if $A[U] \neq$ Null

  Print "Insert not possible"

  return.

$i = U$

while $i \geq K$

  $A[i+1] = A[i]$

  $i = i - 1$

$A[K] = $ item

$U = U + 1$

2 5 13 3 6

2 5 13 3 6 6

2 5 13 3 3 6

$O(n)$

Deletion

A	5	3	9	2	6	4	7

k

item .

```
Delete (A, L, U, item)

  i = Search (A, L, U, item)
  if  i = -1
        Print "item not found"
        return
  while  i < U
        A[i] = A[i+1]
        i = i+1

X  A[U] = null
   U = U-1
```

$O(n)$

2-dimensional array

- 2 indices are required to reference all the element of an array.
- array of arrays
- rectangular arrangements of array.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & \cdots & a_{0n} \\ a_{10} & a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & & & & \\ a_{m0} & a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$\text{int } A[m][n] = \{ \underbrace{\{ \quad \}}_{\text{row 1}} \; \underbrace{\{ \quad \}}_{\text{row 2}} \; \{ \quad \} \; \cdots \; \{ \quad \} \};$$

Two ways to represent a 2-D array in the memory.

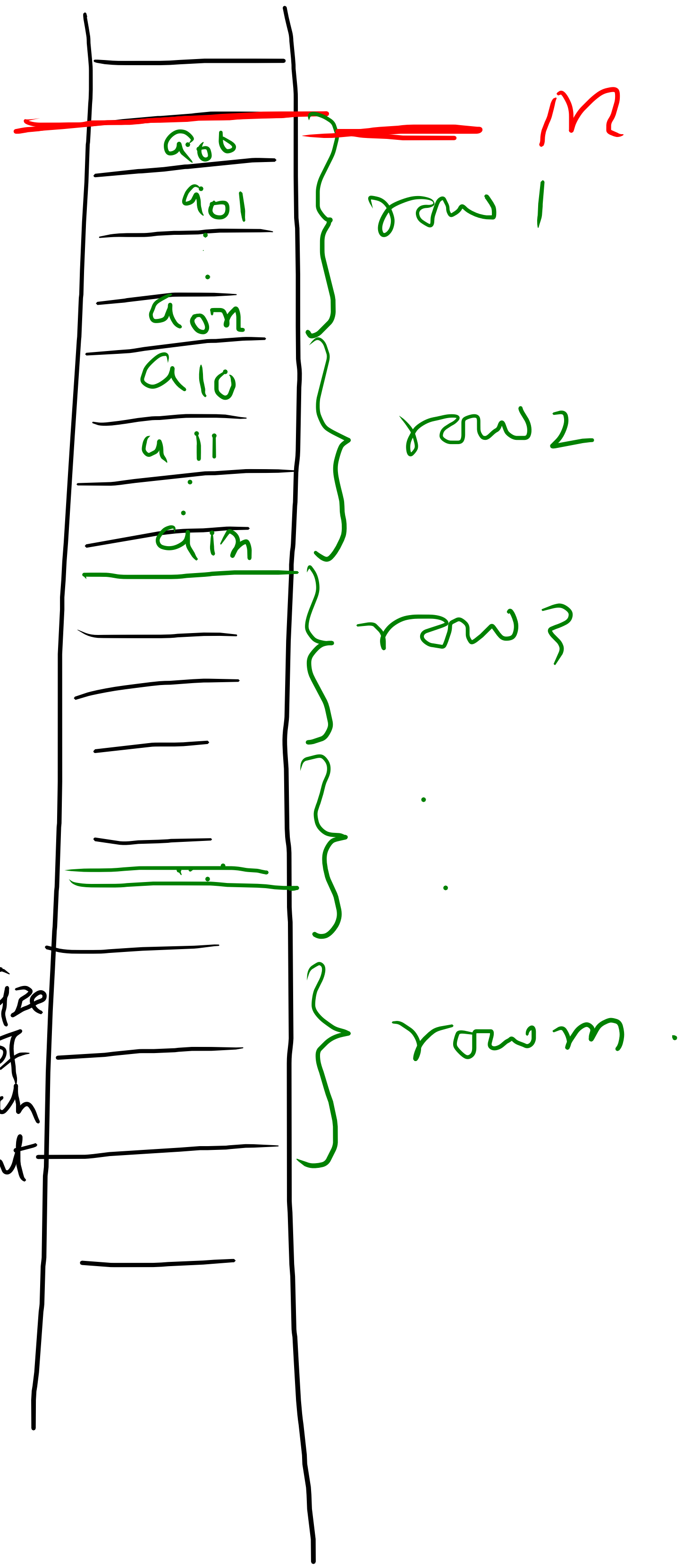
1. row-major

2. column major.

row major

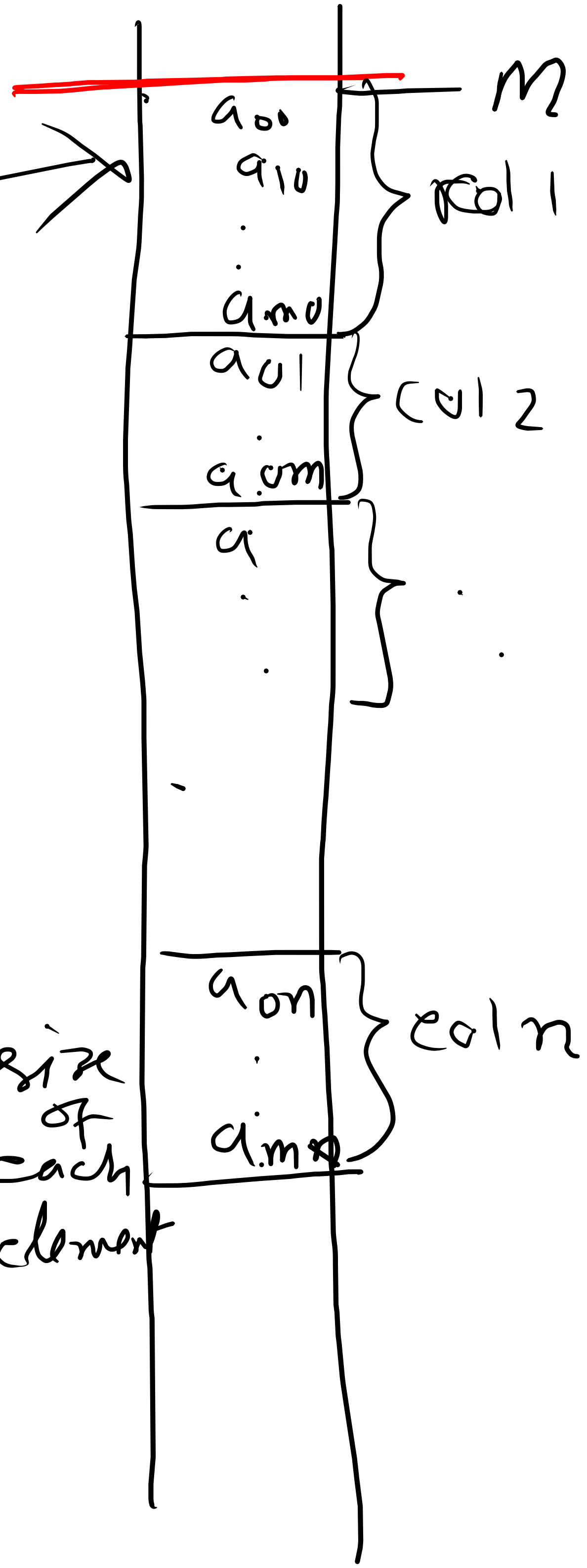
$$A = \begin{array}{cccc} a_{00} & a_{01} & \cdots & a_{0n} \\ a_{10} & a_{11} & \cdots & a_{1n} \\ i & & a_{ij} & \\ a_{m0} & a_{m1} & \cdots & a_{mn} \end{array}$$

$$\text{Address}(A[i][j]) = M + [(i \times \#\text{ elements in each row}) + j] \times \text{size of each element}$$



column major

$$A = \begin{array}{c|cccccc} & & & & j & & \\ \hline & a_{00} & a_{01} & \cdots & \cdot & & a_{0n} \\ & a_{10} & a_{11} & & \cdot & & a_{1n} \\ & \vdots & & & & & \\ i & \cdot & & & a_{ij} & & \\ & a_{m0} & a_{m1} & \cdots & \cdot & & a_{mn} \end{array}$$



$$\text{Adddress}(A[i][j]) = M + \left[\left(j \times \#\,\text{element in one column}\right) + i\right] \times \text{size of each element}$$

Ex^m

$$A = \begin{matrix} 5 & 9 & 3 & 6 \\ 2 & 1 & 4 & 15 \\ 12 & 7 & 13 & 19 \end{matrix}$$

Base address = 105 and integer data ~~type~~ takes 4 byte.

Address (A[2][3]) =

row major — 149

column major — 149

Address (A[1][2])

row major - 129

col major — 133

classic Data Structures

Debasis Samanta.