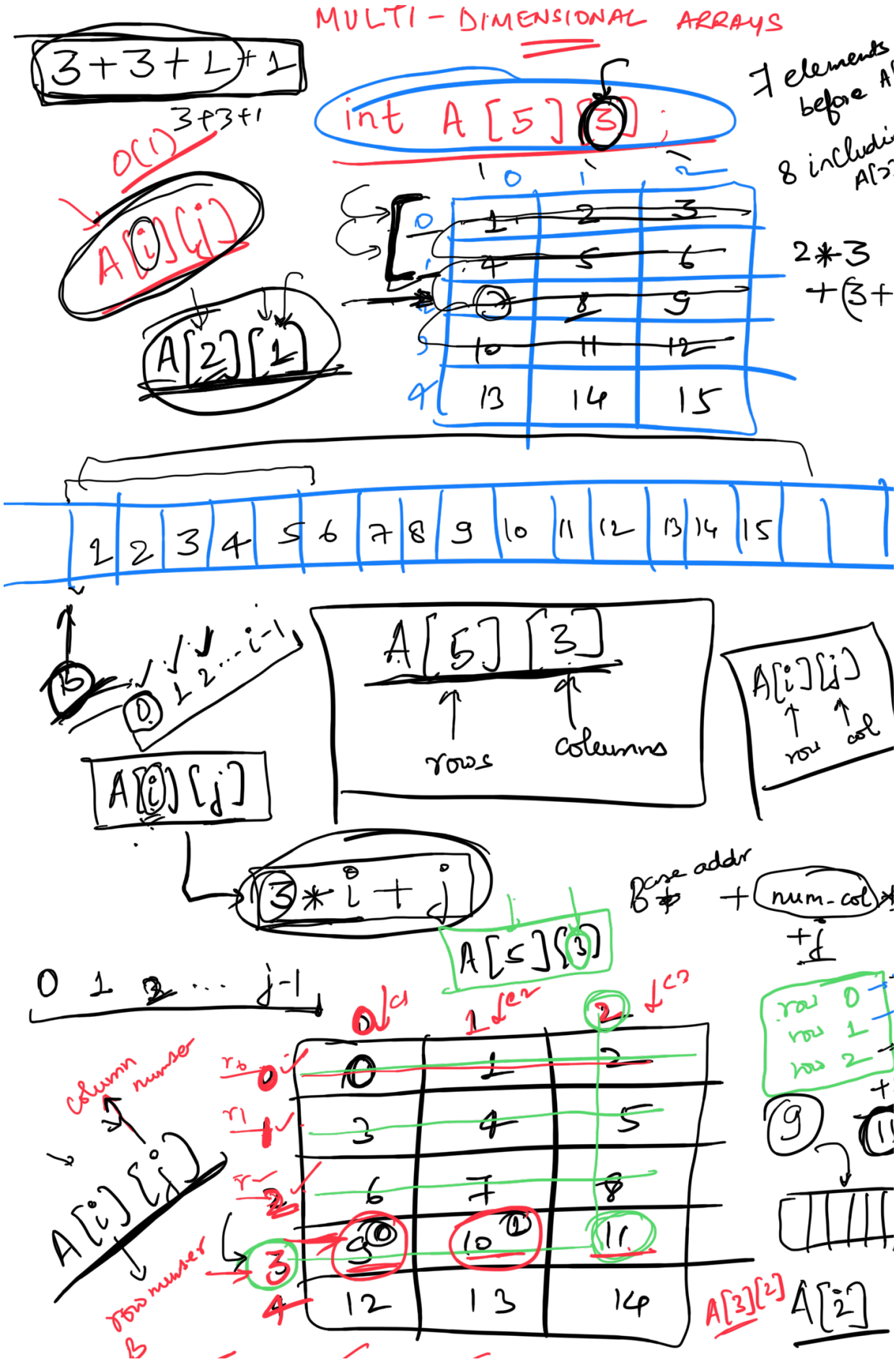
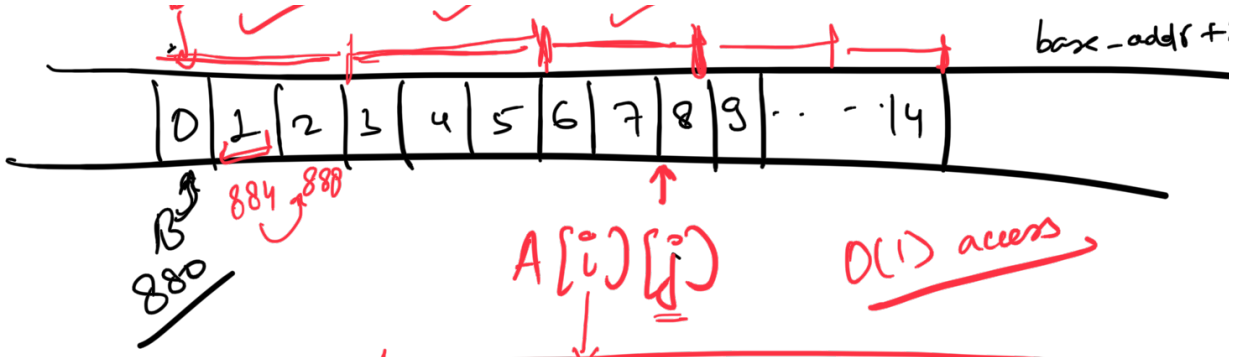
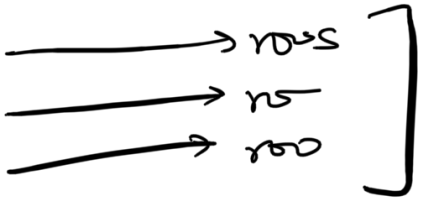
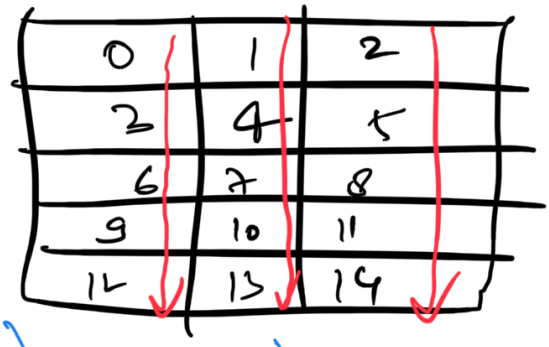

MULTI - DIMENSIONAL ARRAYS
3+3+L+1
3+3+1
O(1)
A[i][j]
A[2][1]
int A[5][3];
elements before A
8 including A[2
2*3
+(3+
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
A[5][3]
rows
columns
A[i][j]
row col
A[i][j]
3*i+j
base addr
num_col
A[5][3]
0 1 2 ... j-1
column number
row number
A[i][j]
row 0
row 1
row 2
A[3][2]
A[2]



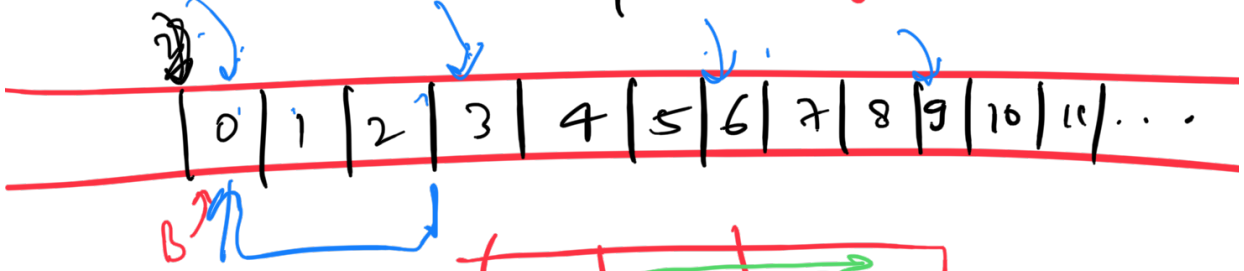
Static A[5][3]

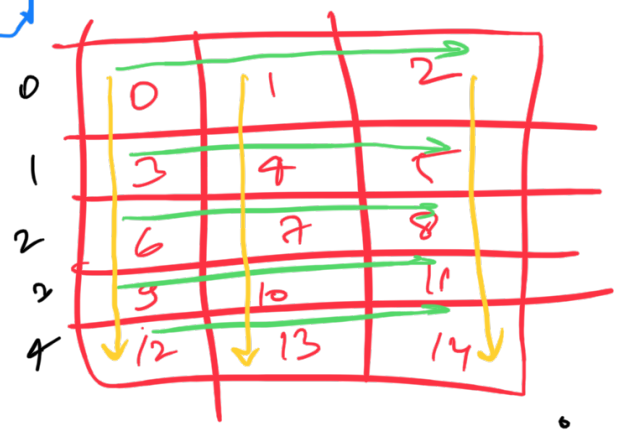


row major way of storing numbers



column major





row majo

Code #1

i → 0 to 5 :
  j → 0 to 3 :

j → 0 to 3 :
  i → 0 to 5
  print A

print A[i][j]

0 3 6 9 12

0 1 2 3 4 5 6 7 8 ...

[2, "abcd", 'a', ...]

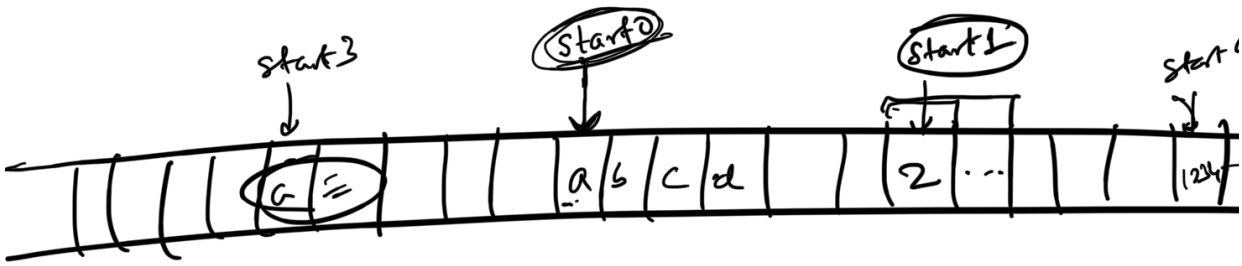
4 bytes

A[i] base_addr + i * sizeof(A[0])

A[i] ← O(1)

index

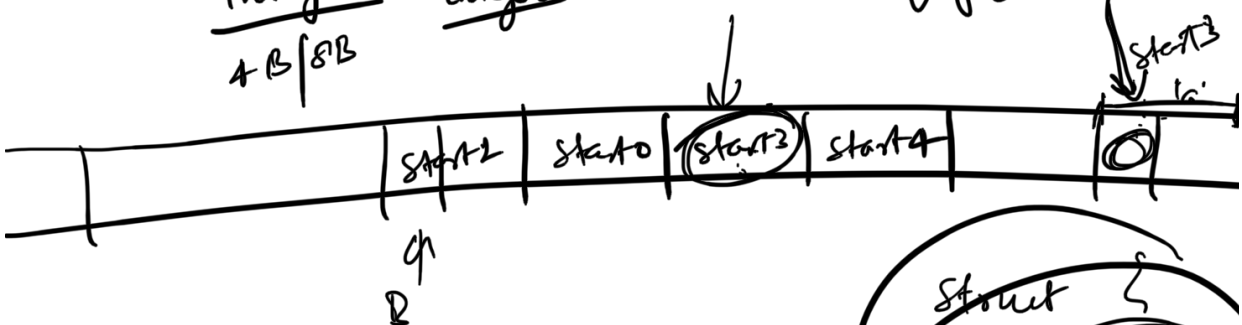
[2, "abcd", "a", 123] A[2]



[start2, start0, start3, start4] str = "abcd"

integer integer A[2] object → character

4B 8B



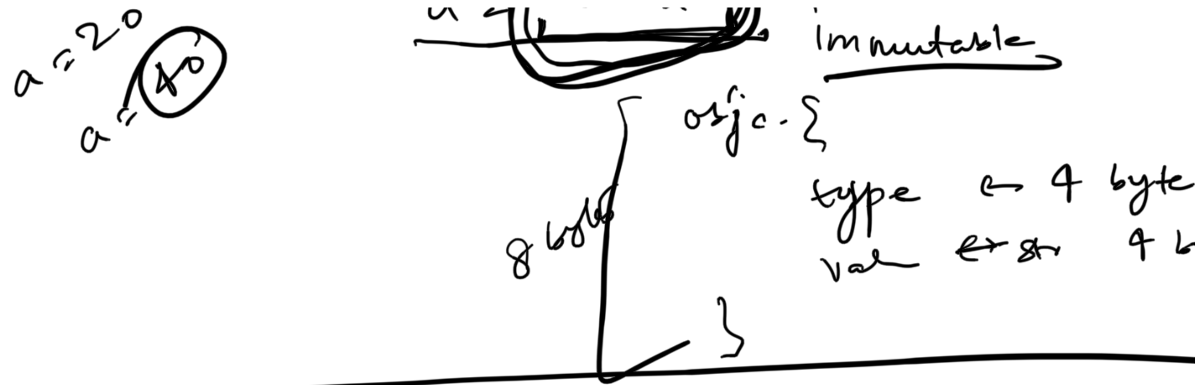
[10, 12, 13, 14]

a: "abcd"

a = "abcd"

struct

bool

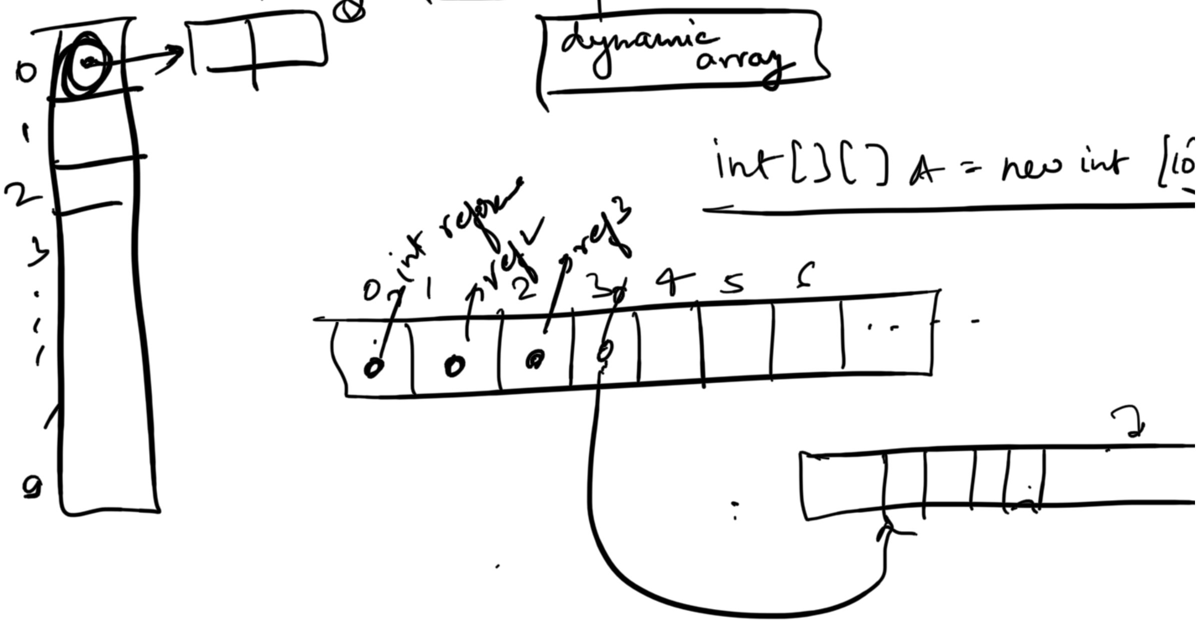


int A[5][3]   15 integers

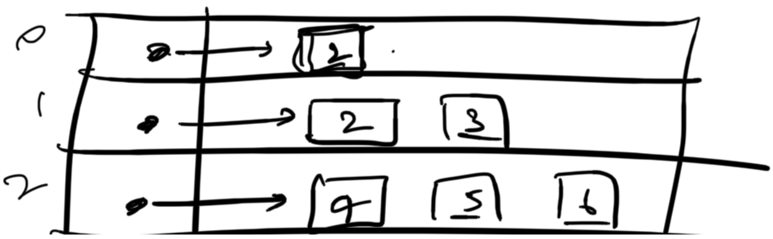
vector<int> A[10];

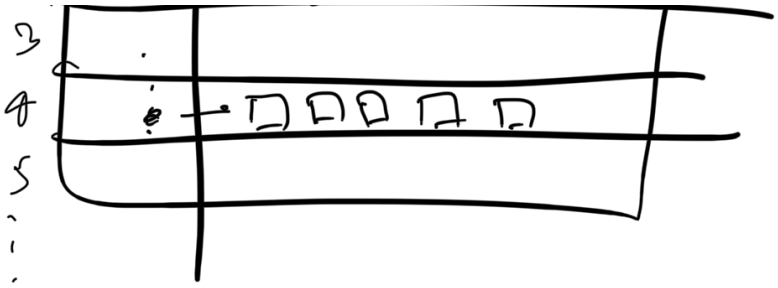
dynamic array

int[][] A = new int [10



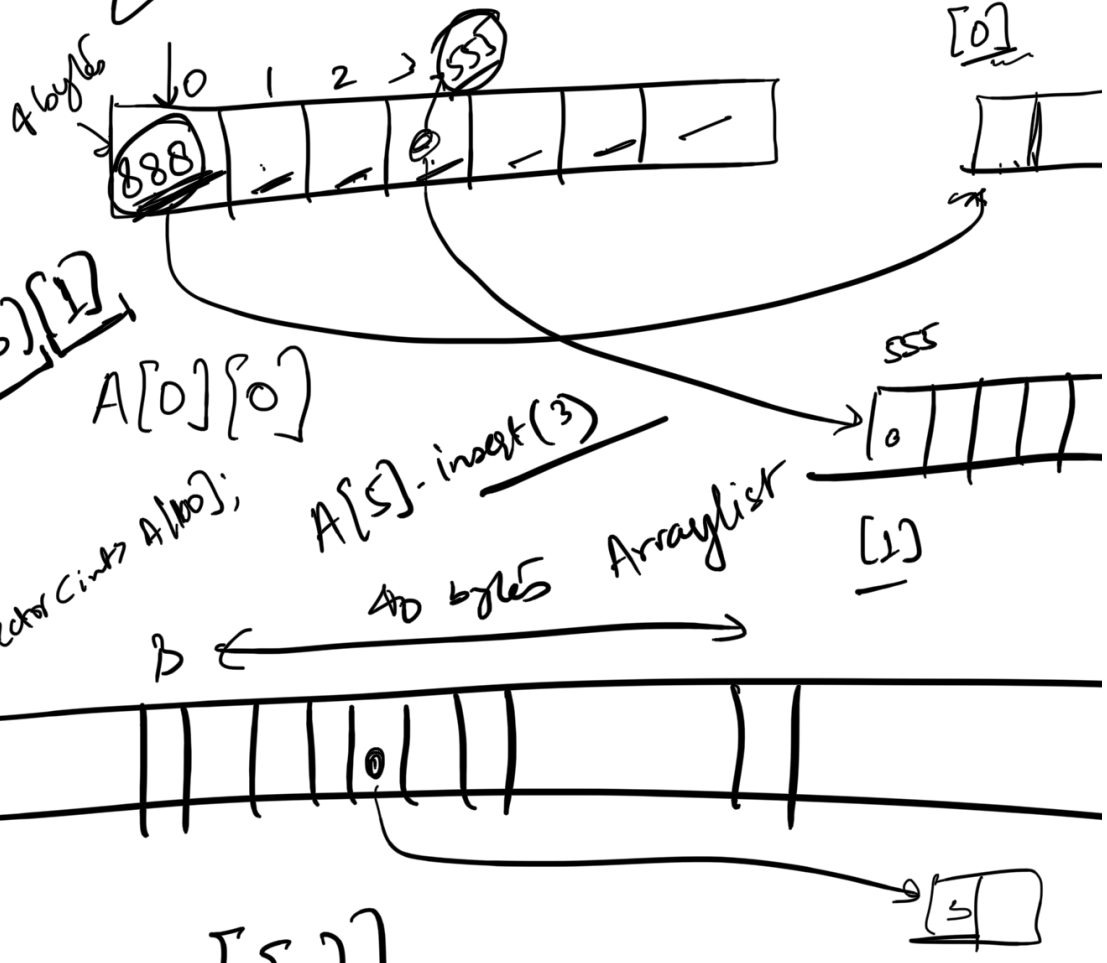
A[10] → Static no. of rows.
Variable no of columns.





Arraylist < Arraylist < Integer >>

vector < vector < int > >



Arraylist = vector

= list

C++

vector<vector<int>> A;

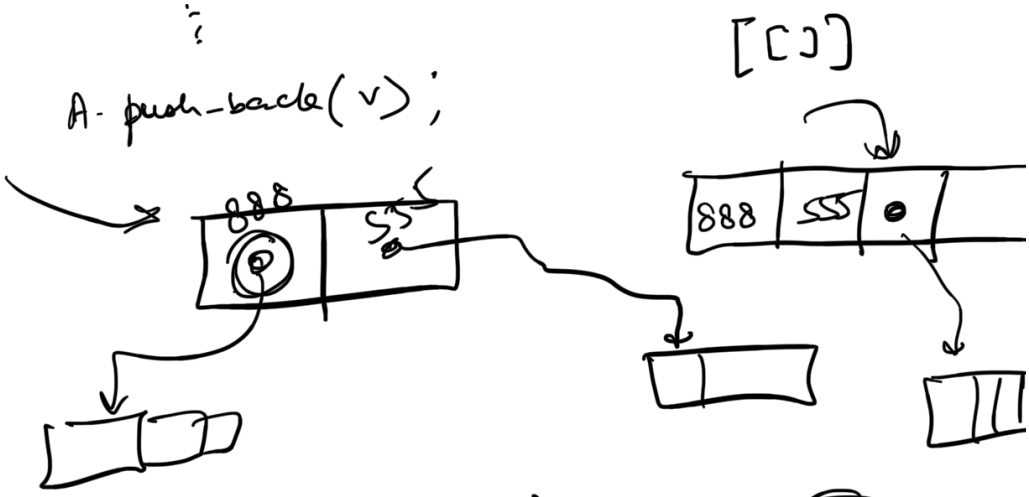
vector<int> V;

JAVA

Arraylist< Arraylist< Integer

Python

A.push_back(v);



int A[5][3] ← static

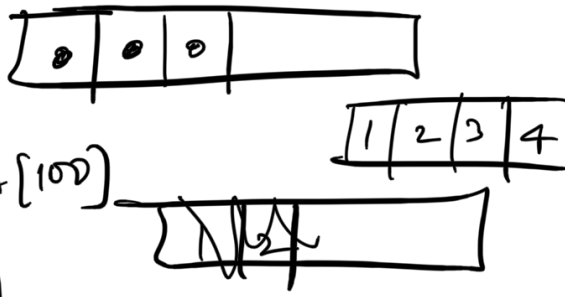
15

Vector c

Array<Integer> A=

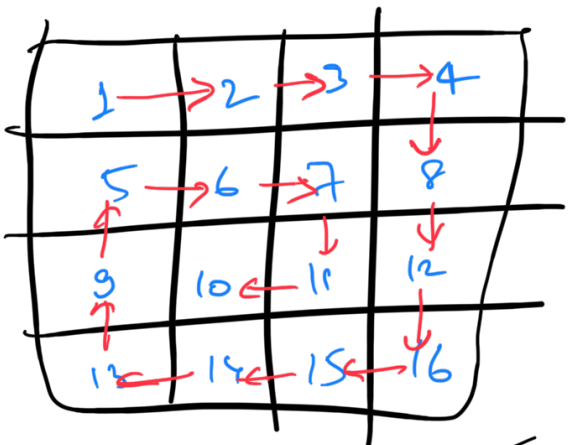
A.set(100,100); A[100]

A[1]

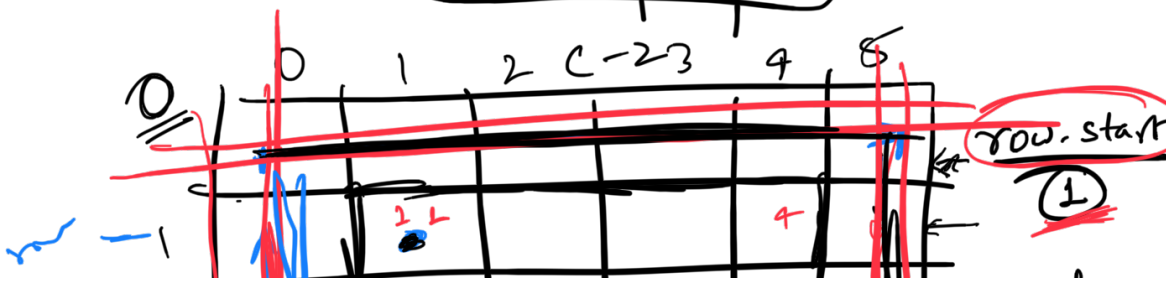


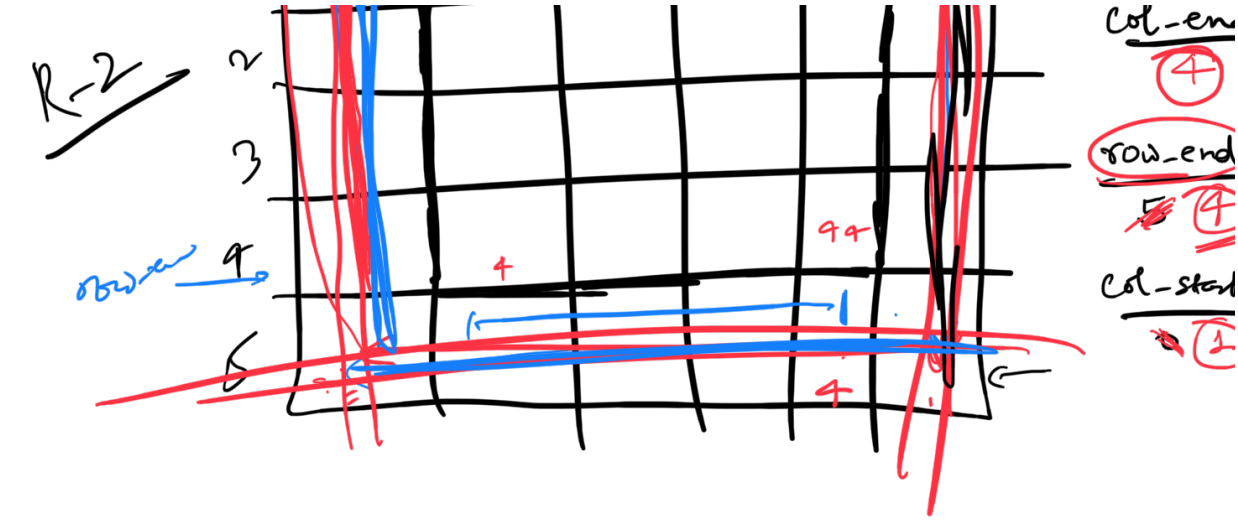
A.insert(1)

A=(2)
(3)
(4)



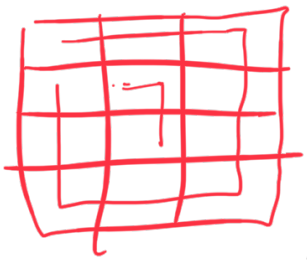
1 2 3 4 8
12 16 15 14
9 5 6 7
10





row_start = 0 , row_end = R-1, col_start = 0, col_end = C-

while (col_start < col_end && row_start < row_end) {

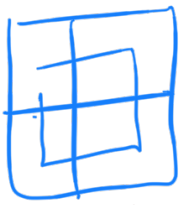


print 1st row ← [c → col_start to col_end : print A[row_start][c]

row_start += 1

print last col ← [r → row_start to row_end : print A[r][col_end]

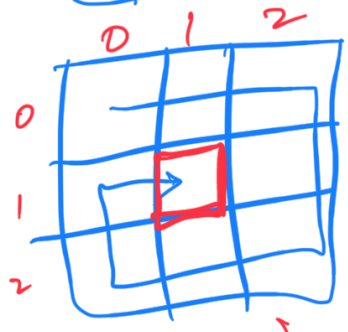
col_end --



C → col_end to col_start ; print A[row_end][c]

col_start == col_end
row_s == row_e

row_end --;



r → row_end to row_start
print A[r][col_start]

col_start ++ ;

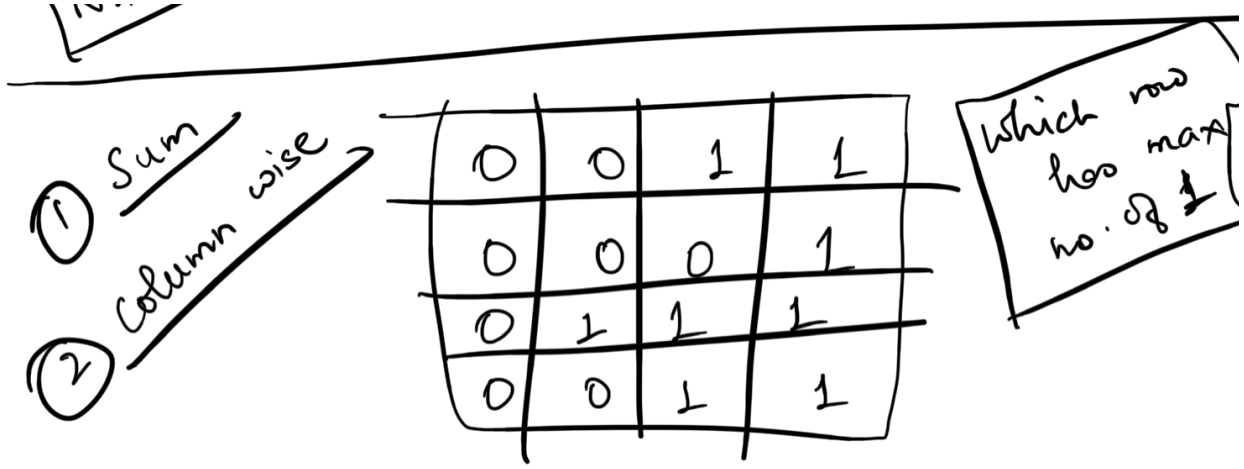
}

N ← row
M ← col

O(N*M)

if (row_st == row_end && col_st == col_end) print A[row_s][col_s]

N*M elements


1. Sum
2. Column wise
0 0 1 1
0 0 0 1
0 1 1 1
0 0 1 1
which row has max no. of 1