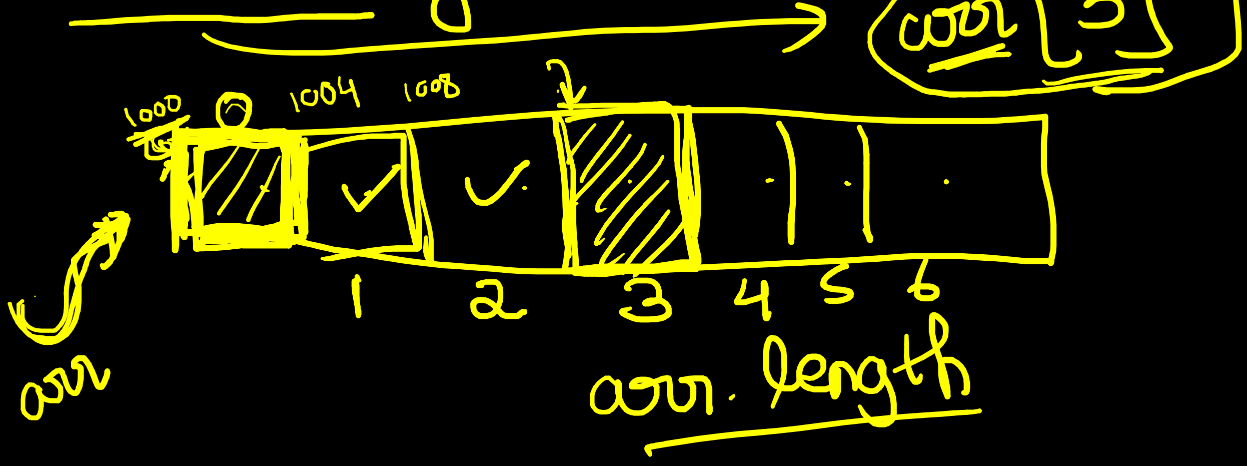
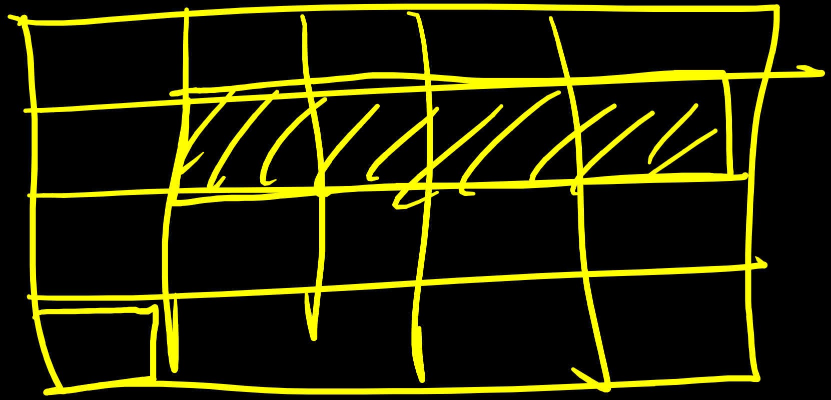
1-D Arrays



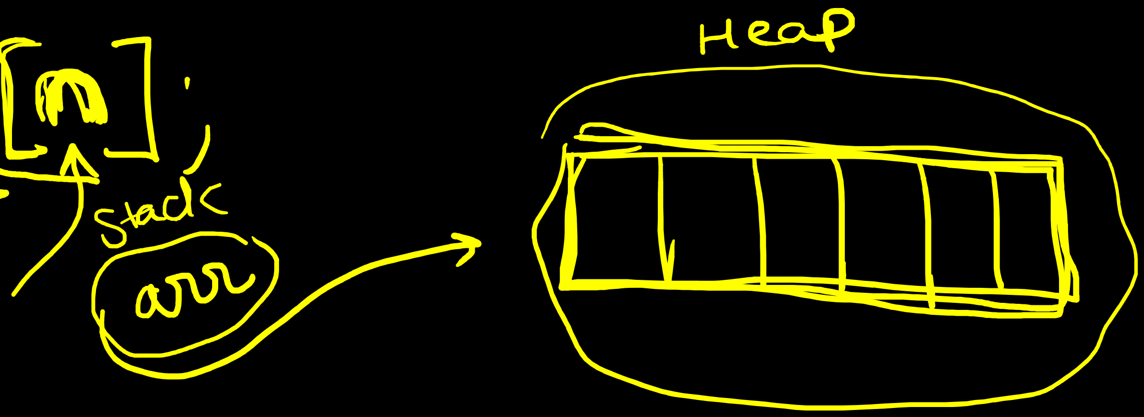
```
for (i=0;i<5;i++)
{ for (j=0;j<10;j++)
  {

  }
}
```



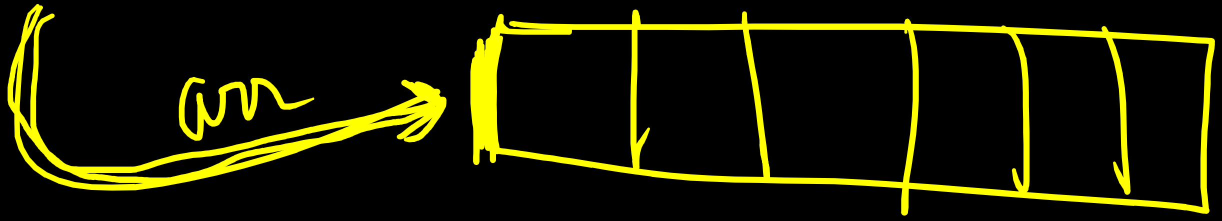
```
for (int i=0; i<arr.length; i++)
{ SOP(arr[i]); }
```

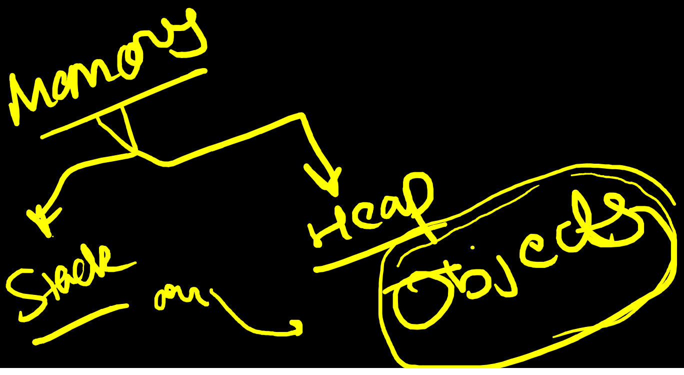
int [] arr = new int[n];



SOP(arr)

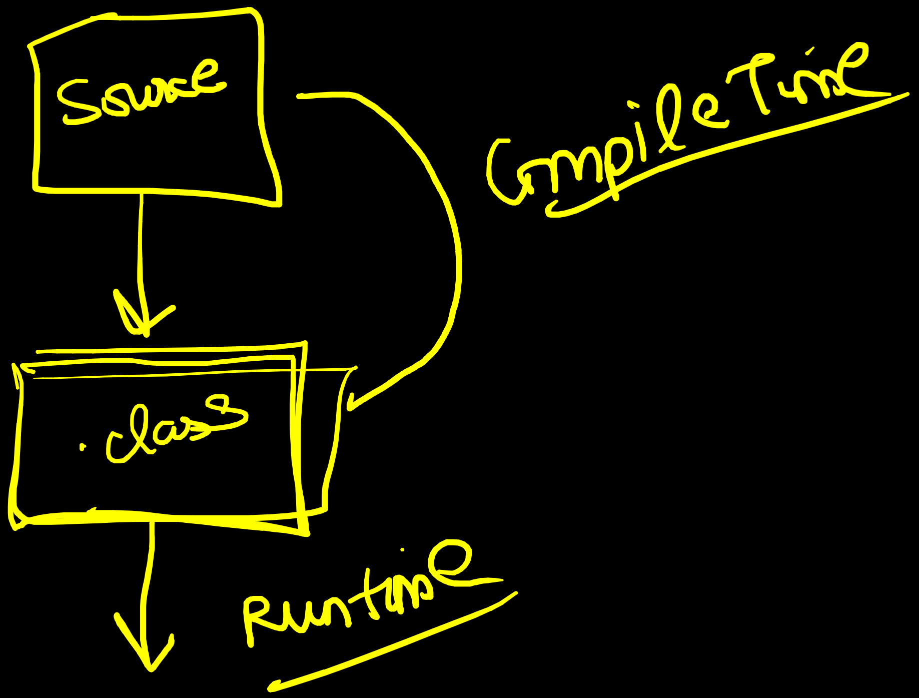
arr[i]





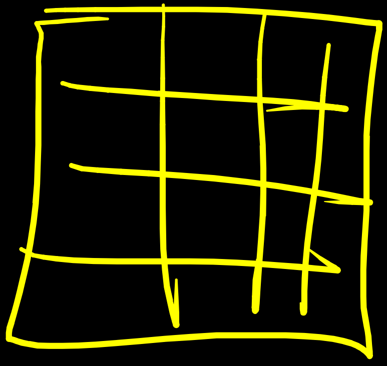
int [] barr = {8, 9, 10, 11}; size = 4

Runtime



2-D Arrays

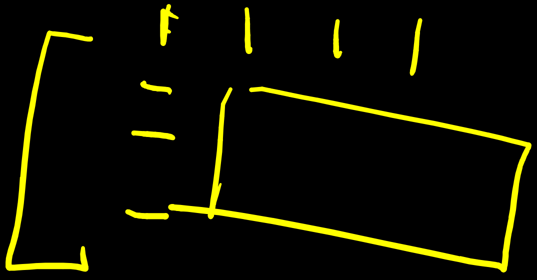
matrix



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

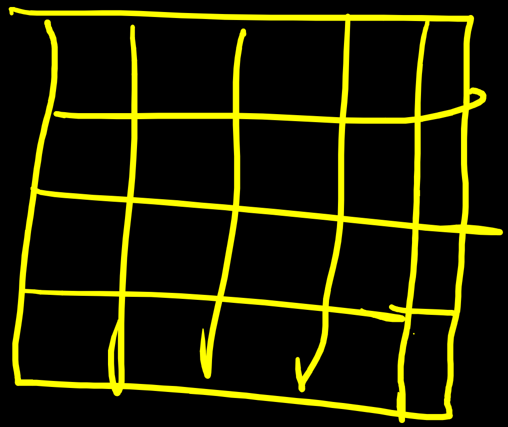
$\underline{m \times n} \longrightarrow$ m rows
n columns

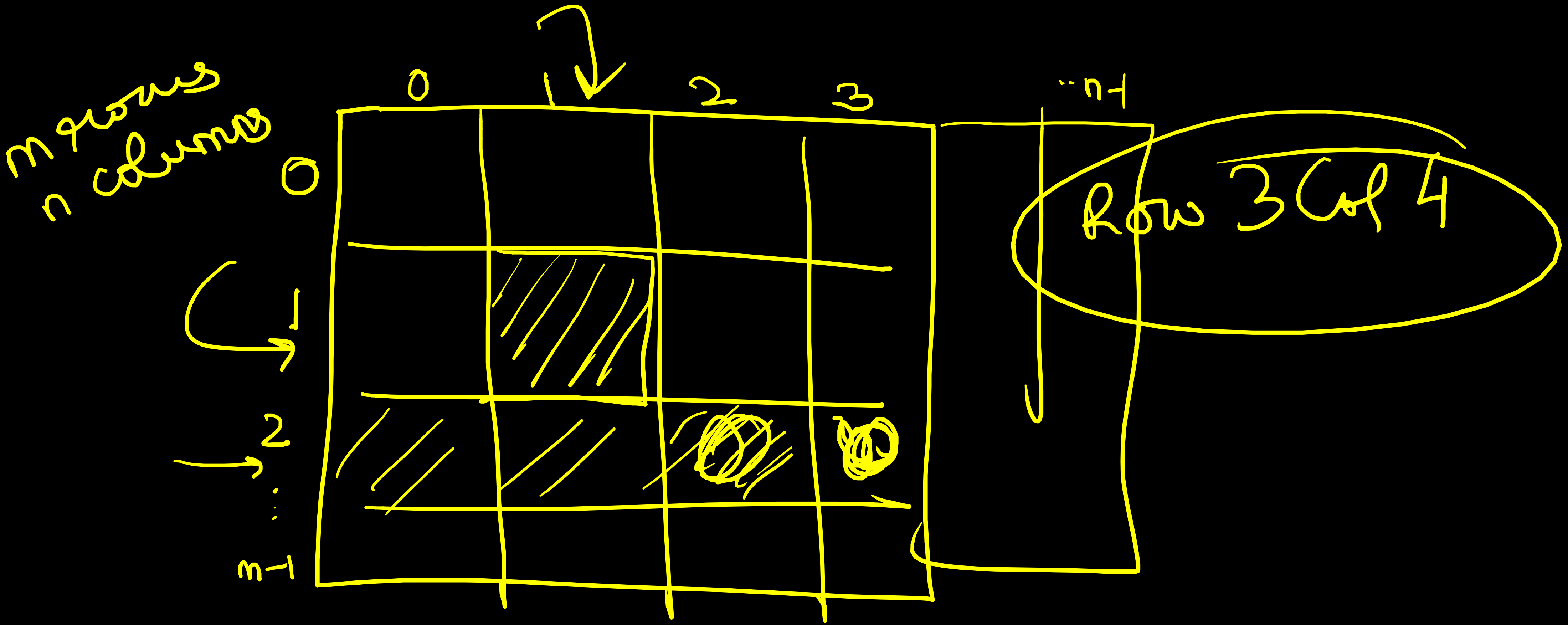
m
n



$$\texttt{int [ ][ ] mat = new int } \underbrace{[m]}_{\text{rows}}\underbrace{[n]}_{\text{columns}};$$

m




m rows
n columns
0
1
2
3
..n-1
0
1
2
...
m-1
Row 3 Col 4

3X4

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline \rightarrow & 1 & 2 & 3 & 6 \\ \rightarrow & 4 & 5 & \boxed{7} & 8 \\ 2 & 9 & 3 & 1 & 8 \end{array}$$

Nesting

gow

mat[1][2]

```
int [][] mat = new int[3][4] ;
for( int i=0; i<3 ; i++)
{  // i is the row number
  → for( int j=0; j<4 ; j++)
    {     → // ith row jth column
       mat[i][j]=sc.nextInt();
    }
}
```

m, n

mxn

2-D Arrays

int[] arr.length

$$\begin{bmatrix} 8 & 5 & 8 & 7 \\ 1 & 2 & 3 & 4 \\ 0 & 1 & 9 & 6 \end{bmatrix}$$

{ {8, 5, 8, 7}, {1, 2, 3, 4}, {0, 1, 9, 6} }

mat { {8, 5, 8, 7} {1, 2, 3, 4} {0, 1, 9, 6} } mat.length

{8, 5, 8, 5, 7}

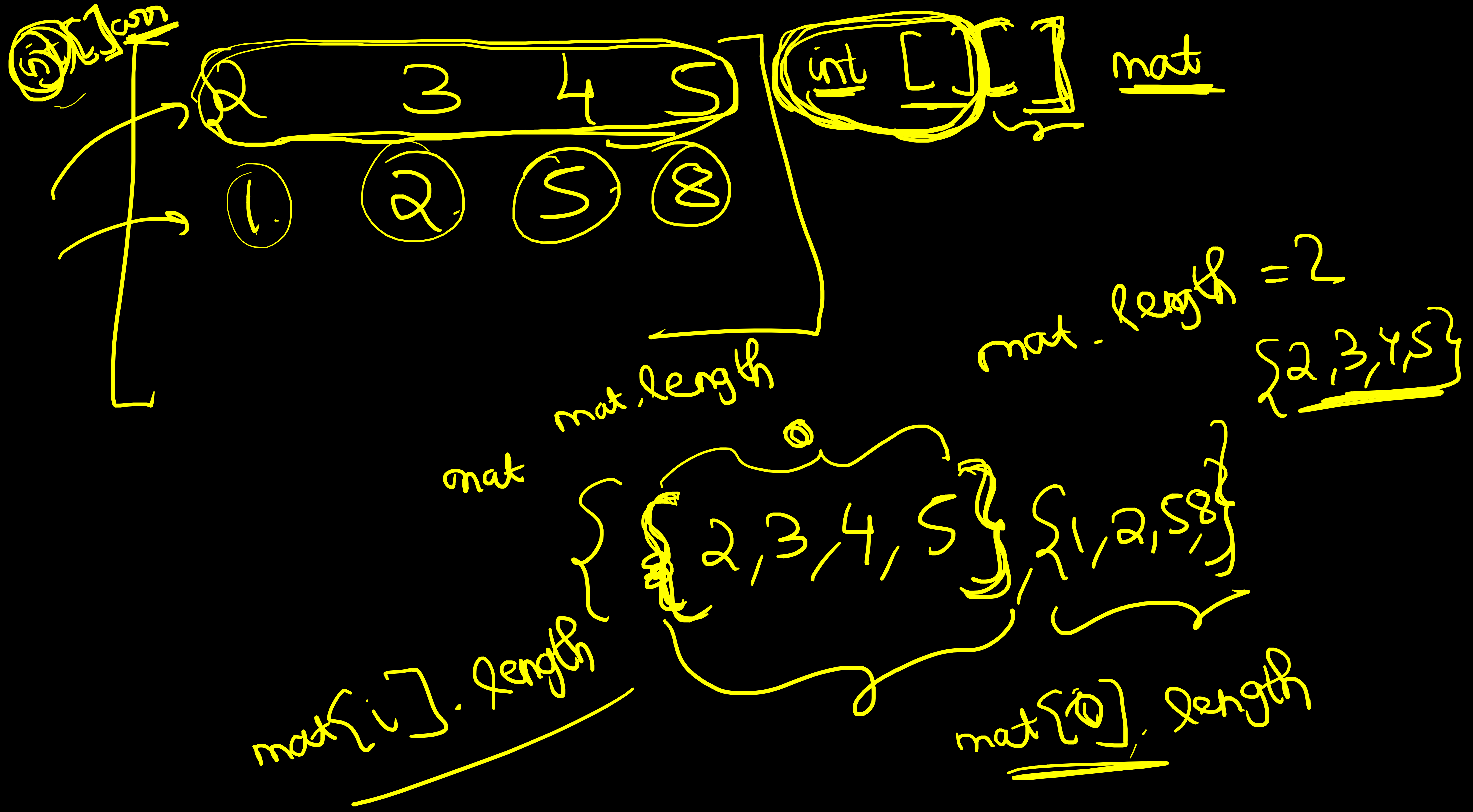
arr.length

{ {1, 2, 3, 4}, {8, 7, 6, 8} }

mat[i].length

mat.length

no. of rows



int[] arr
2 3 4 5
1 2 5 8
int [][] mat
mat.length = 2
{2,3,4,5}
mat
mat.length
0
{{2,3,4,5},{1,2,5,8}}
mat[i].length
mat[0].length

$\underbrace{\text{int [][]}}$ mat2 = { {1, 2}, {1, 2, 3} };

0 ... 1

~~mat2.length~~

Jagged Array?

```
int [][] mat = new int[m][n];

mat.length → no. of rows
mat[i].length → no. of col

for (i=0; i< mat.length; i++)
{   for ( j=0 ; j< mat[i].length; j++)
    {SOP(arr[i][j]);
```

int [ ] [ ] mat = { {8, 7, 6}, {5, 8} }

mat.length

mat [i]. length