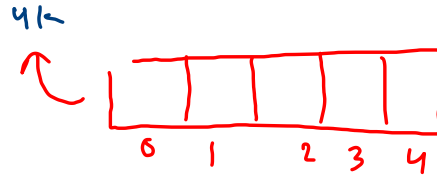


2d Arrays

Arrays

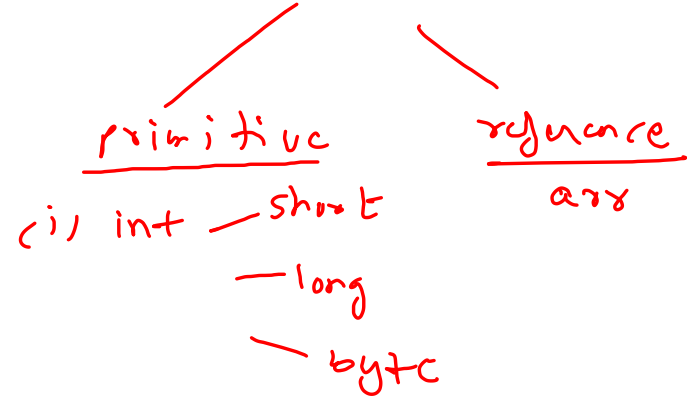
```
int [ ] arr = new int [5];
```



main

arr = 4k

Variable



(ii) double, float

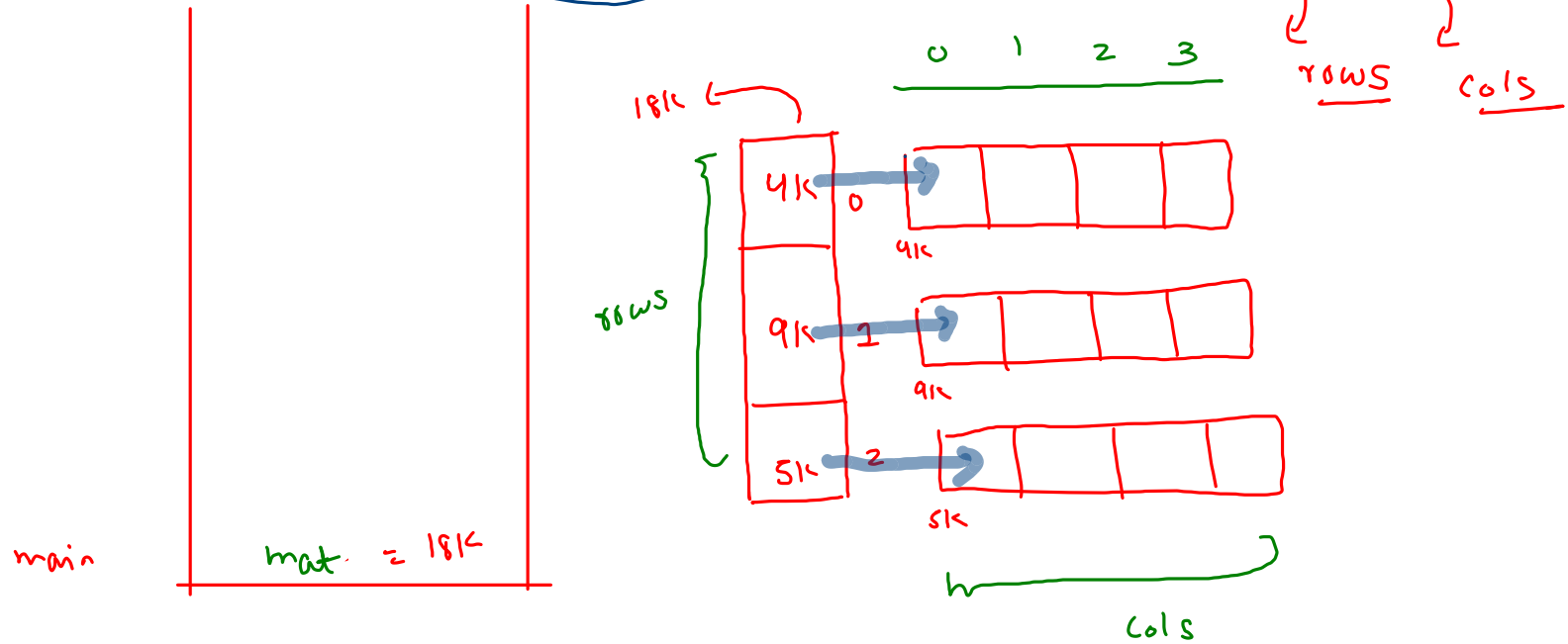
(iii) char

(iv) boolean

2d array

```
int [] a = new int [5];
```

```
int[][] mat = new int [3][4];
```



`rows = mat.length`

`cols = mat[0].length`

matrix
mult

$$\begin{matrix} A \\ []_{r1 \times c1} \end{matrix} \times \begin{matrix} B \\ []_{r2 \times c2} \end{matrix} = \begin{matrix} \text{res} \\ []_{r1 \times c2} \end{matrix}$$

mult is possible : $(c1 == r2)$

res dimension : $r1, c2$

$$\begin{matrix} A \\ \rightarrow 0 \\ 2 \\ [\begin{matrix} 1 & 2 \\ -1 & 0 \end{matrix}]_{2 \times 2} \end{matrix} \times \begin{matrix} B \\ [\begin{matrix} 0 & 1 & 2 \\ 2 & -1 & 1 \end{matrix}]_{2 \times 3} \end{matrix} = \begin{matrix} 0 & 1 & 2 \\ 0 & [\begin{matrix} 4 & -1 \end{matrix}]_{2 \times 3} \\ 1 \end{matrix}$$

res(i,j) \rightarrow A(ith row), B(jth col)
one on one mapping

$$\begin{array}{c} \rightarrow \end{array} \begin{array}{c} 0 \quad 1 \quad 2 \quad 3 \\ \begin{bmatrix} 1 & 2 & 0 & 4 \\ 6 & -1 & 6 & 4 \\ 3 & 0 & 8 & 1 \end{bmatrix} \end{array} \quad 3 \times 4 \quad \times \quad \begin{array}{c} \downarrow \\ \begin{array}{c} 0 \quad 1 \\ \begin{bmatrix} 2 & -1 \\ 9 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix} \end{array} \end{array} \quad 4 \times 2 = \begin{array}{c} 0 \quad 1 \\ \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \end{array} \quad 3 \times 2$$

$$i = 0, j = 0$$

$$\underline{\text{res}[0][0]} = A[0][0] * B[0][0] + A[0][1] * B[1][0] + A[0][2] * B[2][0] + A[0][3] * B[3][0]$$

→

$$\begin{matrix} & 0 & 1 & 2 & 3 \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} 1 & 0 & 2 & 3 \\ 2 & 0 & 4 & 1 \end{bmatrix}
 \end{matrix}$$

2×4

\times

↓

$$\begin{matrix} & 0 & 1 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} 1 & -1 \\ 0 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}
 \end{matrix}$$

4×2

$$\begin{matrix} & 0 & 1 \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} \bullet \\ \end{bmatrix}
 \end{matrix}$$

A B

| | | |
|------|---|------|
| 0, 0 | × | 0, 0 |
| 0, 1 | × | 1, 0 |
| 0, 2 | × | 2, 0 |
| 0, 3 | × | 3, 0 |

k

A ith row

0, 2

B jth cols 0, 1

$\left[\underline{0 \rightarrow 4} \quad k \right]$

$A[i][k] \times$

$B[k][j]$

$$\begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} 1 & 0 & 2 & 3 \\ 2 & 0 & 4 & 1 \end{bmatrix}_{2 \times 4}
 \end{matrix}
 \times
 \begin{matrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} 1 & -1 \\ 0 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix}_{4 \times 2}
 \end{matrix}$$

$k = 0, 1, 2, 3$

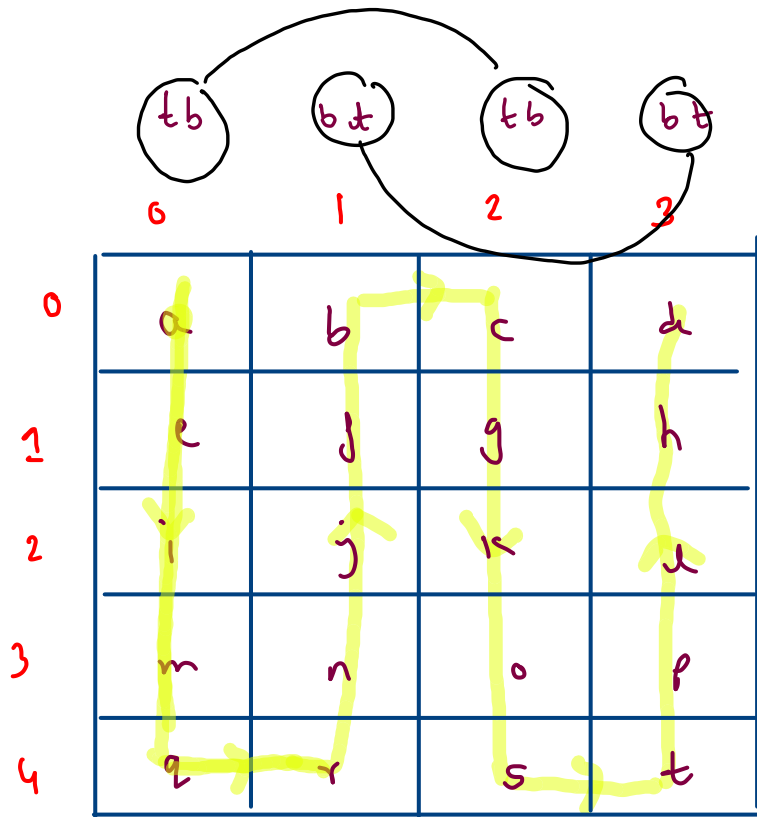
```

//to select ith row of A
for(int i=0; i < r1; i++) {
    //to select jth col of B
    for(int j=0; j < c2; j++) {
        //one on one mapping of A's ith row & B's jth col
        int val = 0;
        for(int k=0; k < c1; k++) {
            val += A[i][k] * B[k][j];
        }
        res[i][j] = val;
    }
}

```

$$\begin{matrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} 12 & 25 \\ \checkmark & \checkmark \end{bmatrix}_{2 \times 2}
 \end{matrix}$$

$$\text{val} = \underline{-1 + 0 + 8 + 18} = 25$$



However, a certain visitor decides to travel a different path as follows:

1. He first travels southwards till no further south places are available.
2. He then moves only 1 place eastwards.
3. He starts to move again towards north till any further north moves are available.

This continues till all the places are covered.

a e i m q r n j j b
c g k o s t p u h d

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | a | b | c | d |
| 1 | e | f | g | h |
| 2 | i | j | k | l |
| 3 | m | n | o | p |
| 4 | q | r | s | t |

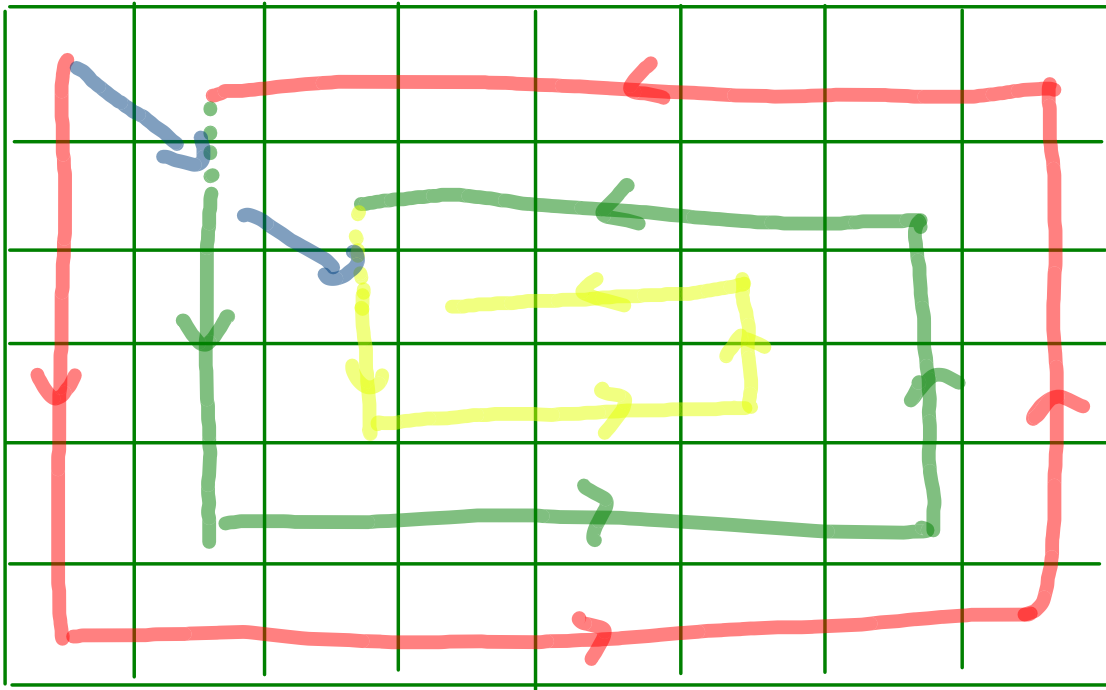
even cols \rightarrow top to
bottom

odd cols \rightarrow bottom
to top

a e i m q r n j f b
c g k o s t p u h d

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|----|----|----|----|----|----|----|
| 0 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 1 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 2 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| 3 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

11 21 31 41 42 43 44
 45 46 47 37 27 17 16
 15 14 13 12 22 32 33
 34 35 36 26 25 24 23



Spiral is made of
shells.

| | | CS | | | CE | | | |
|----|---|----|----|----|----|----|----|----|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| rs | 0 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| | 1 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| | 2 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| | 3 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |

right wall

```
for (int r = re; r >= rs; r--) {
```

```
    syso (mat[r][ce]);
```

```
}
```

```
ce--;
```

top wall

```
for (int c = ce; c >= cs; c--) {
```

```
    syso (mat[rs][c]);
```

```
}
```

```
rs++;
```

shell : 4 walls

left wall, bottom wall, right wall,
top wall.

left wall :

```
for (int r = rs; r <= re; r++) {
```

```
    syso (mat[r][cs]);
```

```
}
```

```
cs++;
```

bottom wall :

```
for (int c = cs; c <= ce; c++) {
```

```
    syso (mat[re][c]);
```

```
}
```

```
re--;
```

while (psj < tne) {

left wall :

```
for (int r = rs; r <= re; r++) {
    syso (mat[r][cs]); psj++;
}
```

cs++;

bottom wall;

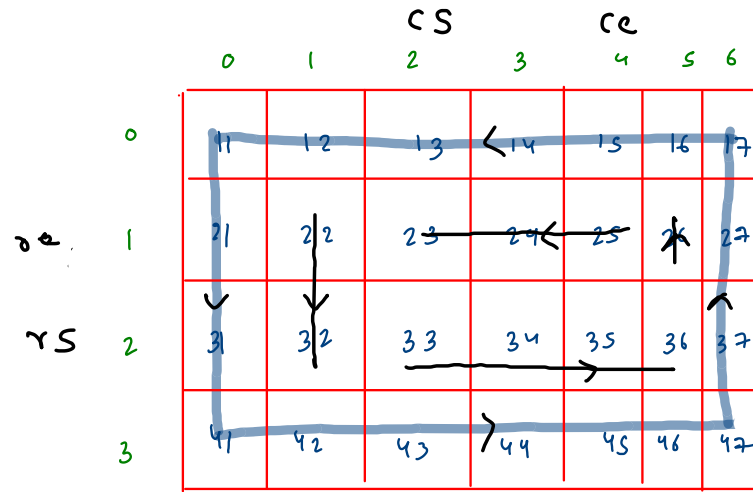
```
for (int c = cs; c <= ce; c++) {
    syso (mat[re][c]); psj++;
}
re--;
```

right wall

```
for (int r = re; r >= rs; r--) {
    syso (mat[r][ce]); psj++;
}
ce--;
```

top wall

```
for (int c = ce; c >= cs; c--) {
    syso (mat[rs][c]); psj++;
}
rs++;
```



tne = 28

11 21 31 41 42 43 44 45

psj = 28

46 47 37 27 17 16 15 14

13 12 22 32 33 34 35 36
26 25 24 23

}

```

while(psf < tne) {
    //left wall
    for(int r=rs; r <= re;r++) {
        System.out.println(mat[r][cs]);
        psf++;
    }
    cs++;

    //bottom wall
    for(int c=cs; c <= ce;c++) {
        System.out.println(mat[re][c]);
        psf++;
    }
    re--;

    //right wall
    for(int r=re; r >= rs;r--) {
        System.out.println(mat[r][ce]);
        psf++;
    }
    ce--;

    //top wall
    for(int c=ce; c >= cs;c--) {
        System.out.println(mat[rs][c]);
        psf++;
    }
    rs++;
}

```

ce
cs

| | 0 | 1 | 2 | 3 | 4 |
|------|----|----|---------|----|----|
| re 0 | 11 | 12 | 13 ← 14 | | 15 |
| 1 | 21 | 22 | 23 → 24 | | 25 |
| rs 2 | 31 | 32 | 33 | 34 | 35 |

11 21 31 32 33 34 35 25 15

tne = 15

14 13 12 22 23 24 23

psf = 12 15 16

```

while(psf < tne) {
    //left wall
    for(int r=rs; r <= re && psf < tne; r++) {
        System.out.println(mat[r][cs]);
        psf++;
    }
    cs++;

    //bottom wall
    for(int c=cs; c <= ce && psf < tne; c++) {
        System.out.println(mat[re][c]);
        psf++;
    }
    re--;

    //right wall
    for(int r=re; r >= rs && psf < tne; r--) {
        System.out.println(mat[r][ce]);
        psf++;
    }
    ce--;

    //top wall
    for(int c=ce; c >= cs && psf < tne; c--) {
        System.out.println(mat[rs][c]);
        psf++;
    }
    rs++;
}

```

ce cs

| | | | |
|------|---|--------------|---|
| | 0 | 1 | 2 |
| 0 | a | b | c |
| rs 1 | d | e | j |
| re 2 | g | h | i |
| 3 | j | k | l |
| 4 | m | n | o |

tne = 15

psf = ~~0~~ / ~~5~~ / ~~A~~

~~11~~

~~12~~

15

if

c1 → T

c2 → T

```
if ( c1 ) {  
    // w1  
}
```

w1 ✓
w2 ✓

```
    if ( c2 ) {  
        // w2  
    }  
    else {  
        // w3  
    }  
}
```

else if

```
if ( c1 ) {  
    // w1
```

w1 ✓

```
    }  
    else if ( c2 ) {  
        // w2  
    }  
    else {  
        // w3  
    }  
}
```