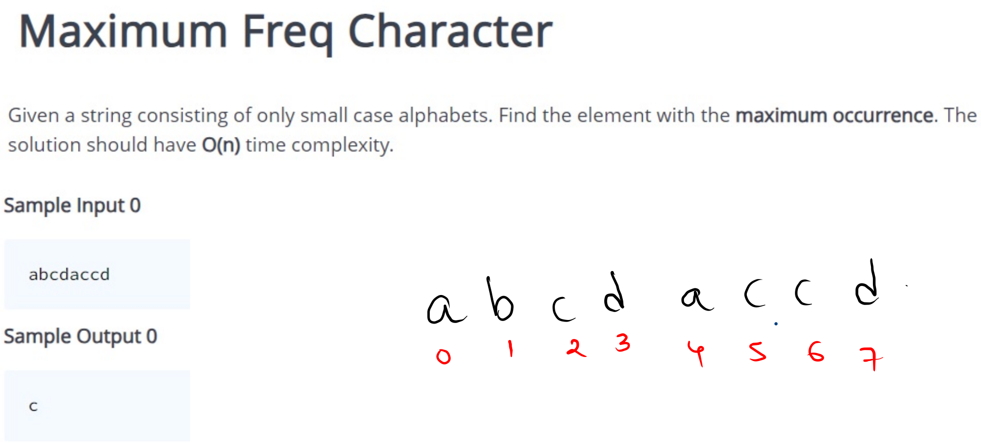


Maximum Freq Character

Given a string consisting of only small case alphabets. Find the element with the **maximum occurrence**. The solution should have **O(n)** time complexity.

Sample Input 0

```
abcdaccd
```

Sample Output 0

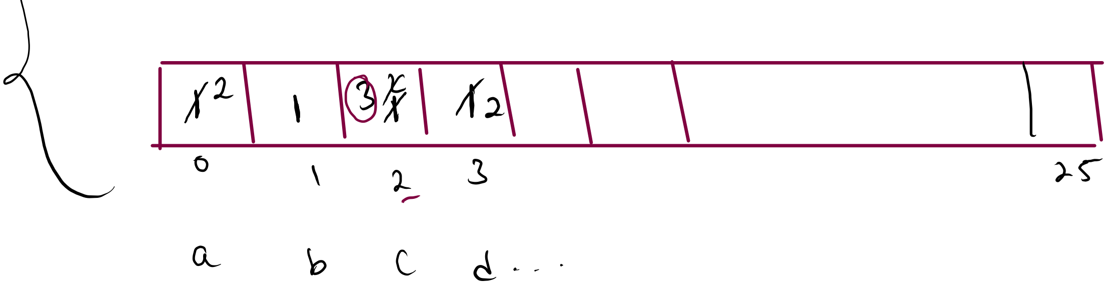
```
c
```

a	b	c	d	a	c	c	d.
0	1	2	3	4	5	6	7

freq.

a	2
b	1
c	3 ✓
d	2

1. freq [ ] → size = 26.



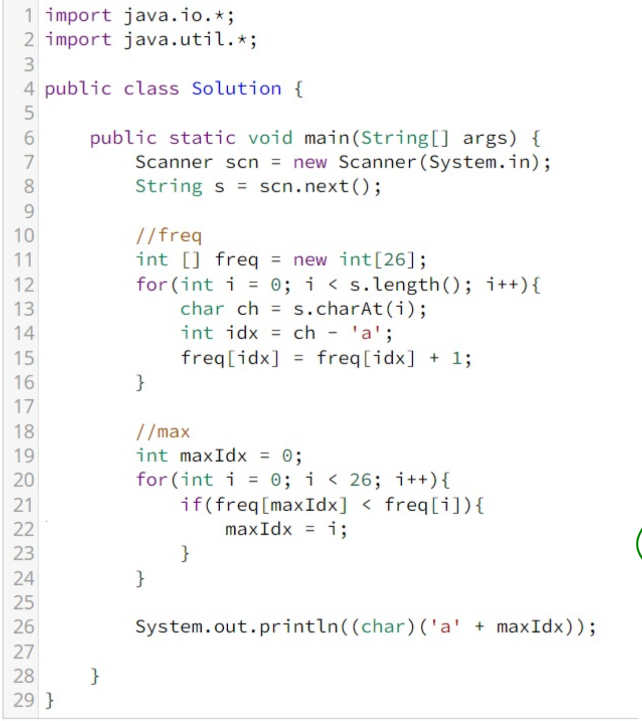
2. at which idx I have max. = 2

$\boxed{'a' + 2.}$

ans = b

O(n) = TC

O(1) = SC

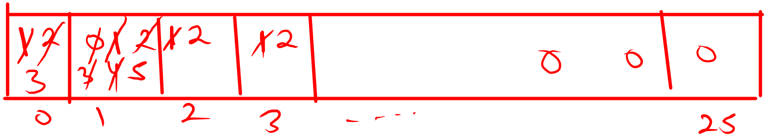


```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();

        //freq
        int [] freq = new int[26];
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            int idx = ch - 'a';
            freq[idx] = freq[idx] + 1;
        }

        //max
        int maxIdx = 0;
        for(int i = 0; i < 26; i++){
            if(freq[maxIdx] < freq[i]){
                maxIdx = i;
            }
        }

        System.out.println((char)('a' + maxIdx));

    }
}
```

" b a a c c d d a b b b b "

0 1 2 3 4 5 6 7 8 9 10 11

i



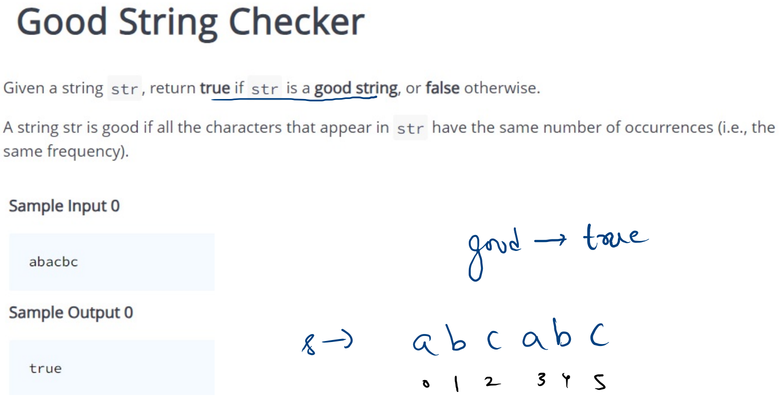
i

mI = ~~0~~ 1

(char) ('a' + 1)

'a' + 1

97 + 1 = 98



Good String Checker

Given a string `str`, return **true** if `str` is a **good string**, or **false** otherwise.

A string str is good if all the characters that appear in `str` have the same number of occurrences (i.e., the same frequency).

Sample Input 0

```
abacbc
```

Sample Output 0

```
true
```

good → true

s → a b c a b c
0 1 2 3 4 5

a - 2
b - 2
c - 2
good.
true.

s → a b c b a

a - 2
b - 2
c - 1
false.

logic.

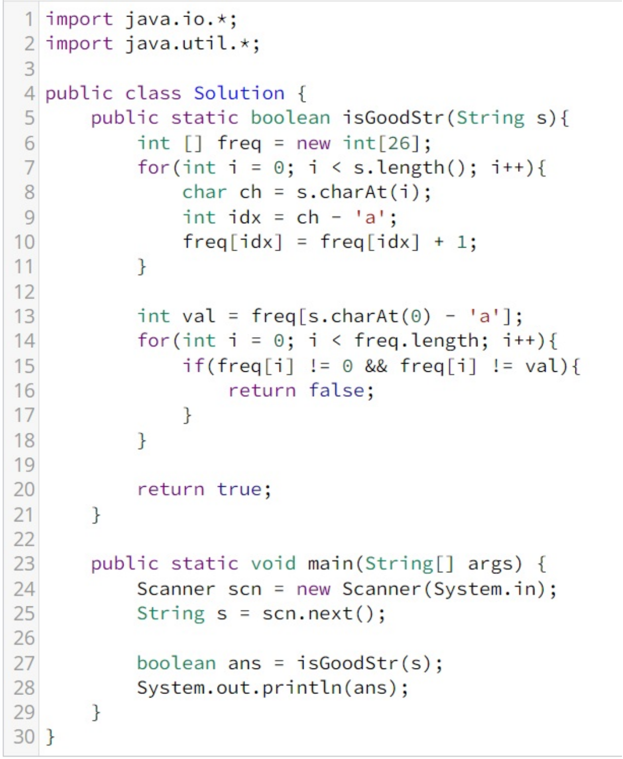
s → a d c c d a
0 1 2 3 4 5

→ val == 0 || val == 2

1. freq.

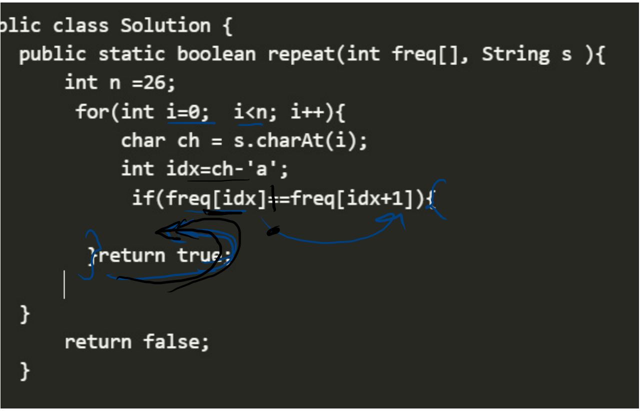
0	1	2	3					25
2	0	2	2	0	0	0	0	0

val {freq} = f['a'] = 2

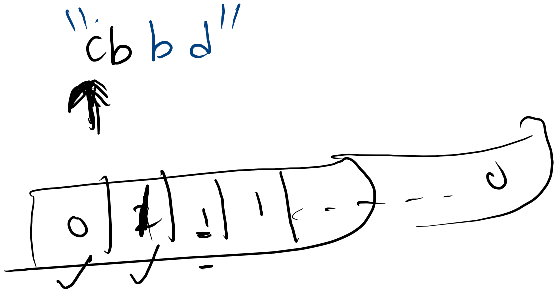


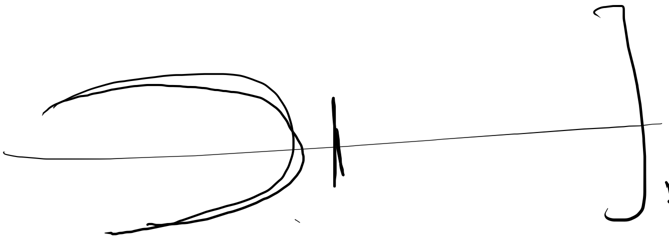
```java
import java.io.*;
import java.util.*;

public class Solution {
    public static boolean isGoodStr(String s){
        int [] freq = new int[26];
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            int idx = ch - 'a';
            freq[idx] = freq[idx] + 1;
        }

        int val = freq[s.charAt(0) - 'a'];
        for(int i = 0; i < freq.length; i++){
            if(freq[i] != 0 && freq[i] != val){
                return false;
            }
        }

        return true;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();

        boolean ans = isGoodStr(s);
        System.out.println(ans);
    }
}
```

s → "c b d b c d"



```
plic class Solution {
  public static boolean repeat(int freq[], String s ){
      int n =26;
       for(int i=0;  i<n; i++){
           char ch = s.charAt(i);
           int idx=ch-'a';
            if(freq[idx]==freq[idx+1]){

        }return true;

  }
      return false;
  }
```





true.

val = 1

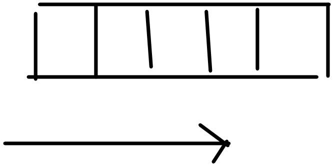
freq[i]
val == 0

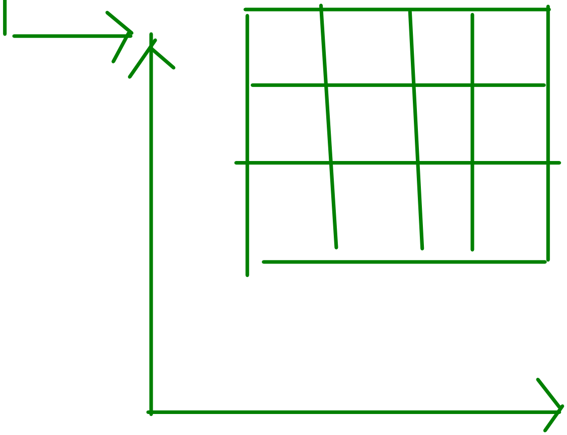
val == freq[i]

return false

2D Array.

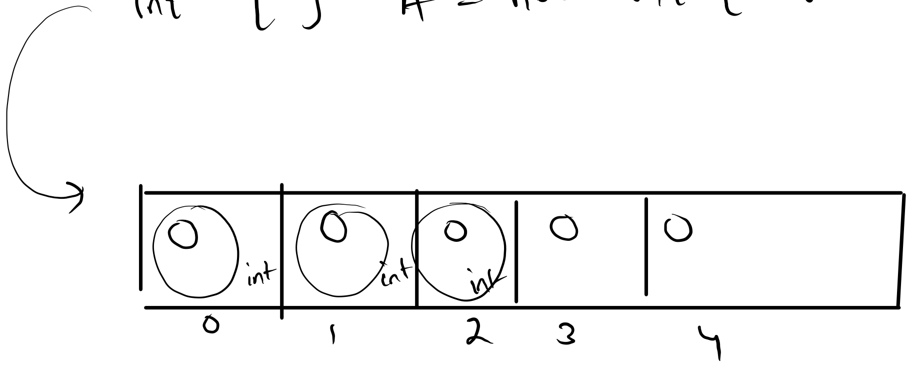
Array → 1D Array.

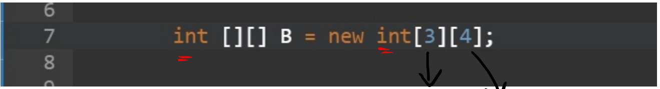




Array

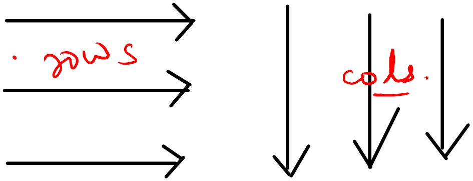
```
int [] A = new int [5];
```





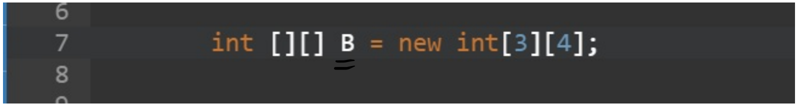
```
        int [][] B = new int[3][4];
```

rows cols.



0	0	0	0
0	0	0	0
0	0	0	0

3x4

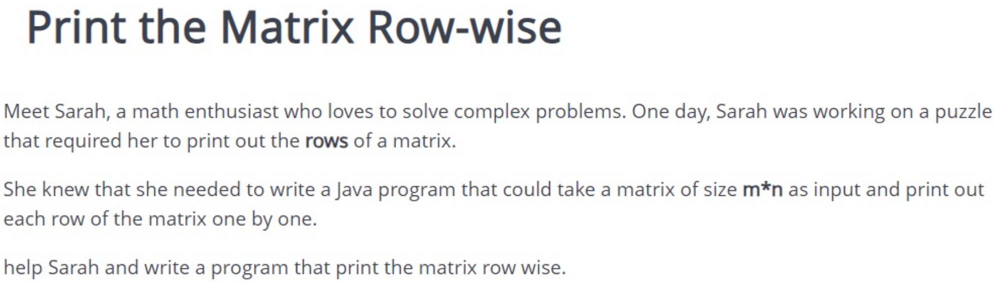


```
int [][] B = new int[3][4];
```

	0	1	2	3
0				
1				○
2		~~0~~ 18		

Syso(B[1][3])

$\boxed{B[2][1] = 18}$



Print the Matrix Row-wise

Meet Sarah, a math enthusiast who loves to solve complex problems. One day, Sarah was working on a puzzle that required her to print out the **rows** of a matrix.

She knew that she needed to write a Java program that could take a matrix of size **m*n** as input and print out each row of the matrix one by one.

help Sarah and write a program that print the matrix row wise.

2D

line → row

10	20
row 30	40
row 50	60

eg. 3 2

10 20

30 40

50 60

m=3

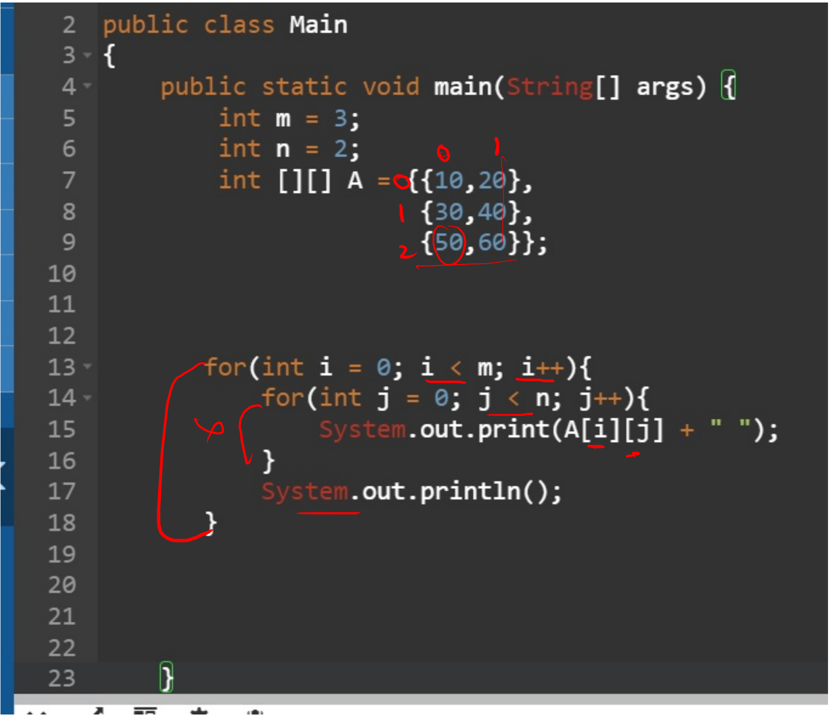
n=2

$$\begin{array}{c|c|c} & 0 & 1 \\ \hline 0 & 10 & 20 \\ 1 & 30 & 40 \\ 2 & 50 & 60 \end{array}$$

i,j

0,0 0,1 1,0 1,1 2,0 2,1

i	j
0 ✓	0 1
1	0 1
2	0 1



```
public class Main
{
    public static void main(String[] args) {
        int m = 3;
        int n = 2;
        int [][] A ={{10,20},
                    {30,40},
                    {50,60}};



        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                System.out.print(A[i][j] + " ");
            }
            System.out.println();
        }



    }
```

i=~~0~~ 0 < 3 ✓ 00 01

~~1~~ 1 < 3 ✓ 10 11

~~2~~ 2 < 3 ✓ 20

3 3 < 3 ✗

j=~~0~~

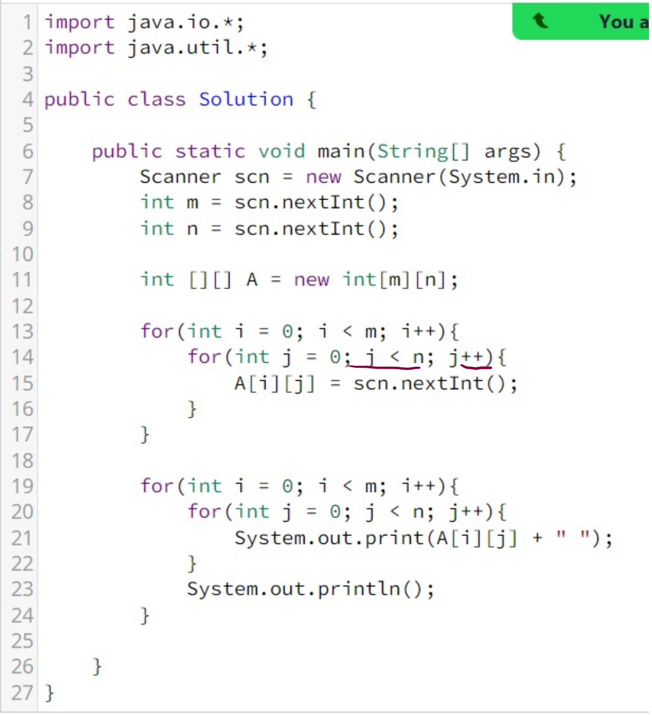
~~1~~ 2 < 2 ✗

2

10 20 __

30 __ 40 ---

50 60 ---



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int m = scn.nextInt();
        int n = scn.nextInt();

        int [][] A = new int[m][n];

        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                A[i][j] = scn.nextInt();
            }
        }

        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                System.out.print(A[i][j] + " ");
            }
            System.out.println();
        }

    }
}
```

m=2

n=2

	0	1
0	10 ✓	20
1	30 ✓	40 ✓

00 01 10 11

i=~~0~~ 0<2 ✓

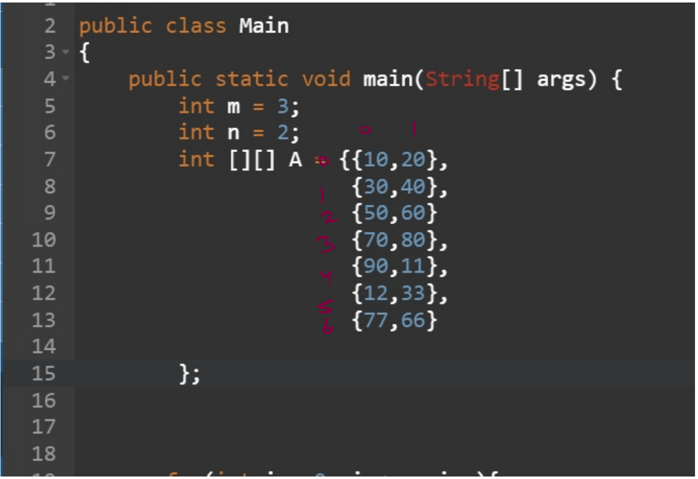
1 1<2 ✓

j=~~0~~ 0<2

+ 1<2 ✓

Print Alternate Rows:

i = even



```java
public class Main
{
    public static void main(String[] args) {
        int m = 3;
        int n = 2;
        int [][] A = {{10,20},
                      {30,40},
                      {50,60}
                      {70,80},
                      {90,11},
                      {12,33},
                      {77,66}

        };
```

10 20

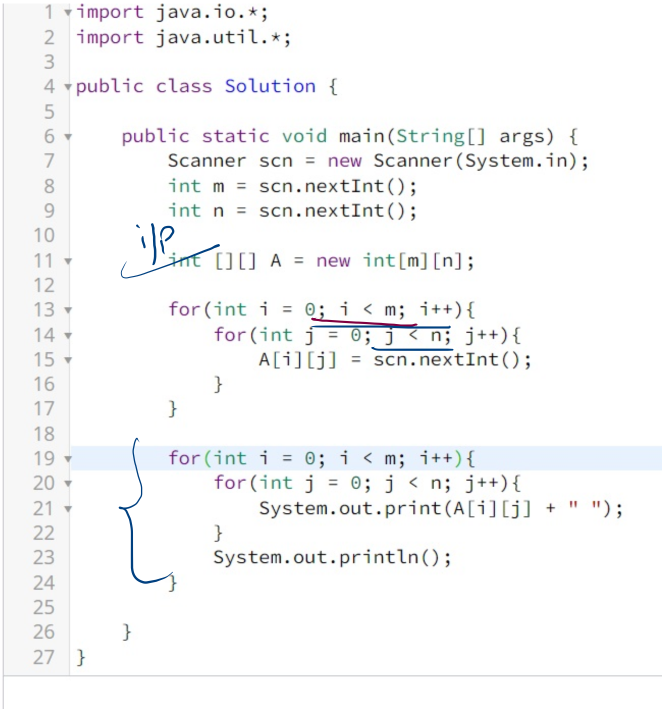
50 60

90 11

77 66

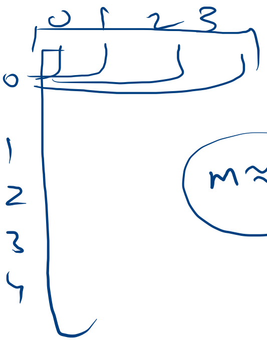
✓0,0	0,1
✓2,0	2,1
✓4,0	4,1
✓6,0	6,1

$SC = O(1)$



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int m = scn.nextInt();
        int n = scn.nextInt();

        int [][] A = new int[m][n];

        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                A[i][j] = scn.nextInt();
            }
        }

        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                System.out.print(A[i][j] + " ");
            }
            System.out.println();
        }

    }
}
```

$\boxed{m \times n}$



$\boxed{m \approx n}$

$i=0 \longrightarrow n$

$i=1 \longrightarrow n$

$i=2 \longrightarrow n$

$\vdots$

$m \longrightarrow n$

---

m

$1 \longrightarrow n$

$m \longrightarrow \boxed{n \times m}$

$\boxed{O(nm)}$

$O(mn)$

$O(n^2)$

$O(m^2)$