Getting Started - Lecture 9 (continued)

Arrays Intro

Memory Mapping

Array Traversal & Printing ($HashCode & toString())

Data Structure

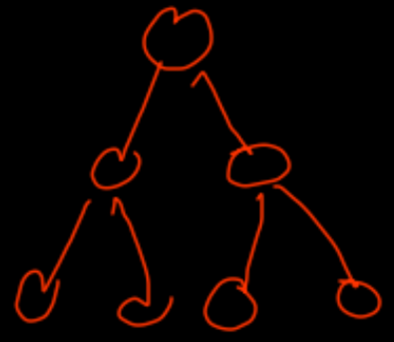
Linear

- Array
- Linked List
- Queue, Dequeue
- Stack

Non-Linear

(Cyclic)
Circular
- Circular LL

(Non-cyclic)
Non-circular



Array

int marks = 98;

int marks2 = 82;

int marks3 = 76;

⋮

Contiguous collection of same datatype.

int x; → declaration

x = 10; → initialize

} Java is a fool-proof language

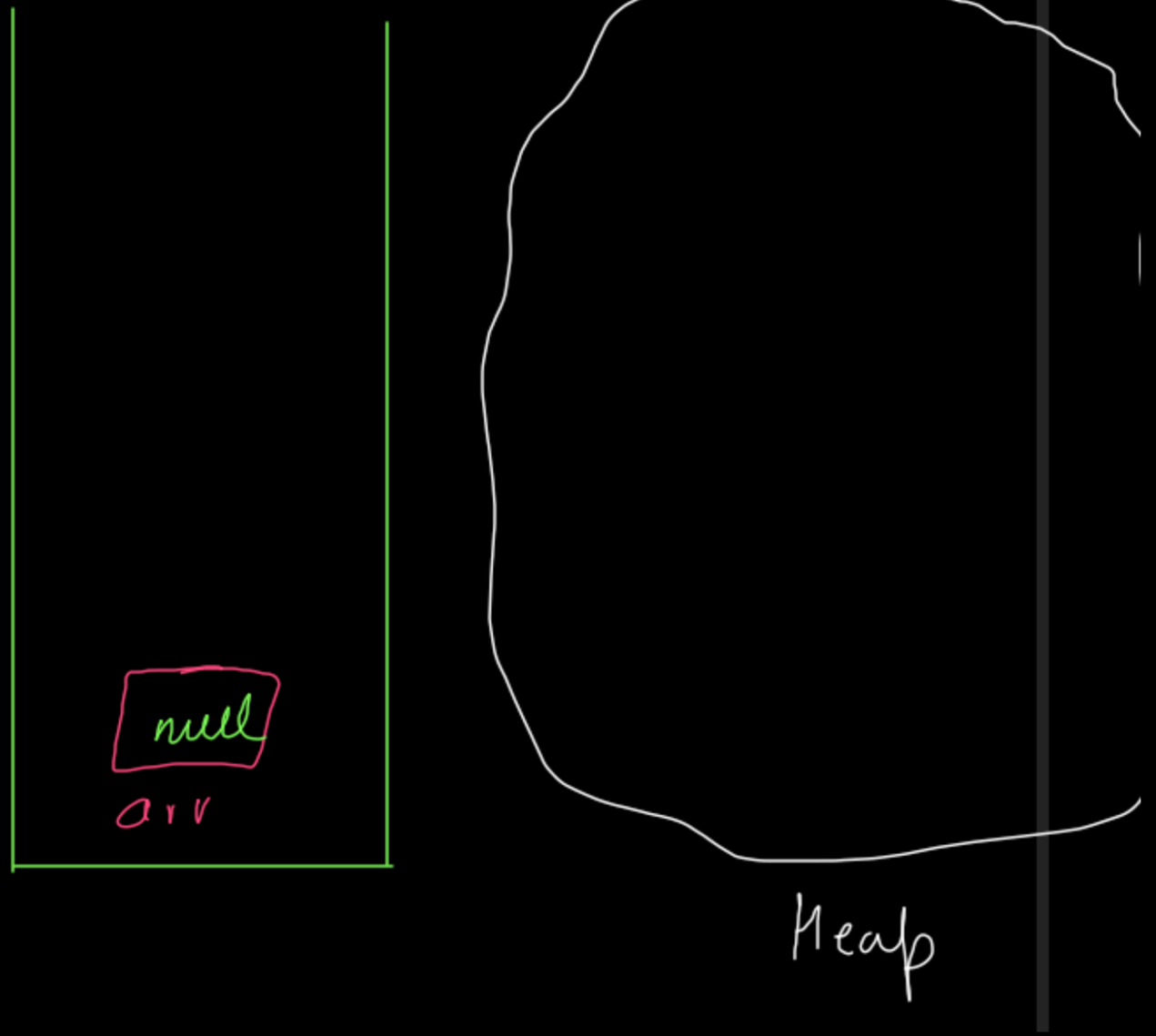
Inbuilt exception handling

```
int [] arr;  // array declaration

arr = new int[10]; // array initialization
```

Array declaration (Memory Mapping)

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration

    }
}
```

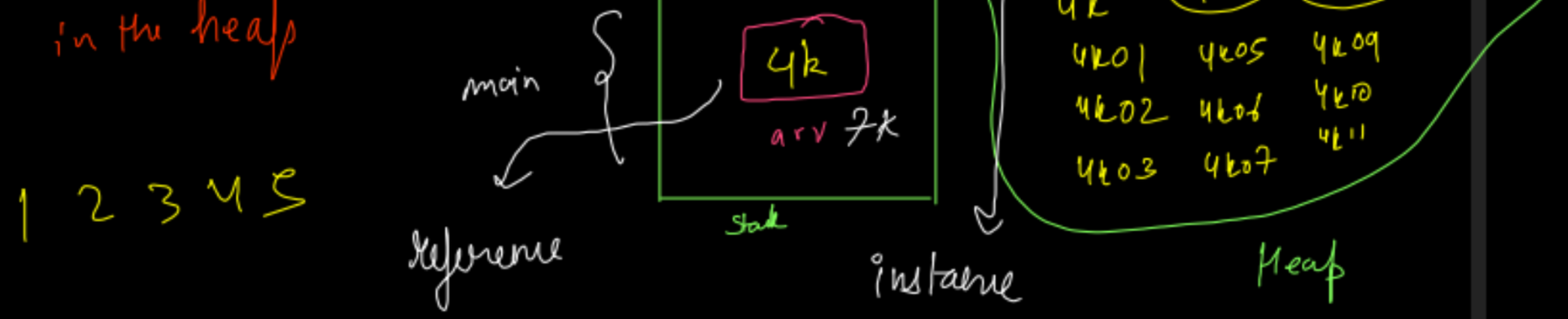
null → it is a type of value that is the intial value for all the arrays / objects



Array Initialization (Memory Mapping)

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        arr = new int[5]; //initialization
    }
}
```

Allocate memory in the heap



The variable that stores the address of actual memory & itself exists in the Stack is called a reference.

The actual memory (formed in the up) is called instance.

Size of a reference variable can never be known in Java.
(In C++ it is : sizeof(int) = 4bytes)

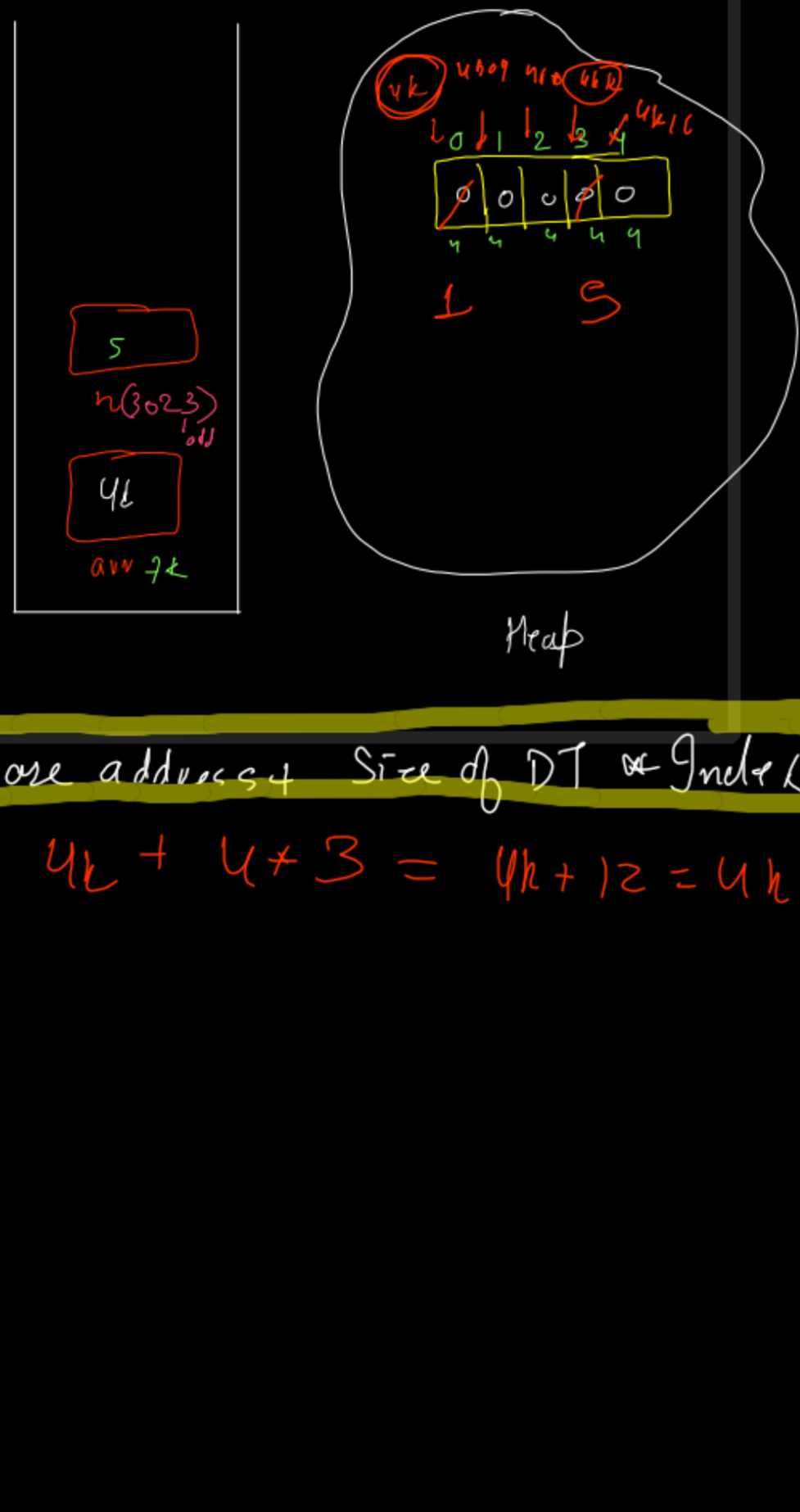
Array will always be created inside heap in Java.

When we don't initialize an array, the reference will store null by default. Null means that the reference is currently empty & does not store any address.

When we initialize an array but have not stored any values in it, Java stores 0 by default. (C++ stores garbage value)

In case of Boolean array, false is stored by default.

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        arr[0] = 1;
        arr[3] = 5;
    }
}
```



Min index = 0

Max index = length of Array - 1

LOC = Base address + Size of DT * Index → O(1)

4k + 4*3 = 4k + 12 = 4k2

Traversing the Array

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        for(int i=0;i<n;i++) {
            arr[i] = i+1;
        }

        for(int i=0;i<n;i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        for(int i=0;i<n;i++) {
            arr[i] = i+1;
        }

        System.out.println(arr);
    }
}
```

```
Finished in 80 ms
[I@2503dbd3
```

This is a hashcode

Whenever you try to print a reference, toString() function is automatically calle

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        for(int i=0;i<n;i++) {
            arr[i] = i+1;
        }

        System.out.println(arr.toString());
    }
}
```

toString is automatically invoked.

Getting Started (Lecture -10)

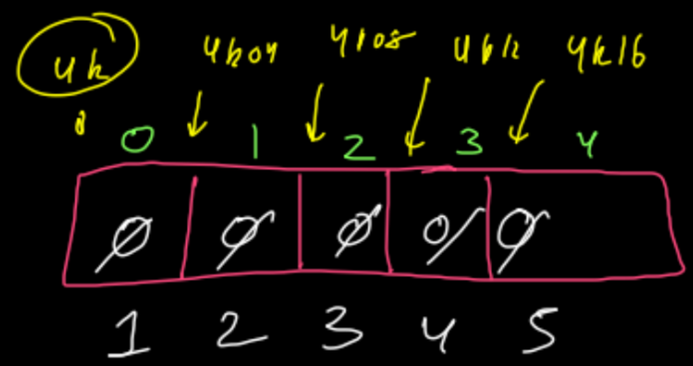
→ Array Length Property
→ Array as a function parameter & as a return type. → Shallow & deep Copy concepts

→ Linear Search
→ Find Min & Max
→ Reverse an Array
→ Left Rotate
→ Right Rotate

→ Bar Chart
→ Binary Search

Length Property

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr; //declaration
        int n = 5;
        arr = new int[n]; //initialization

        for(int i=0;i<arr.length;i++) {
            arr[i] = i+1;
        }

        for(int i=0;i<arr.length;i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```



arr.length = 5

LOC = Base add + DT * index

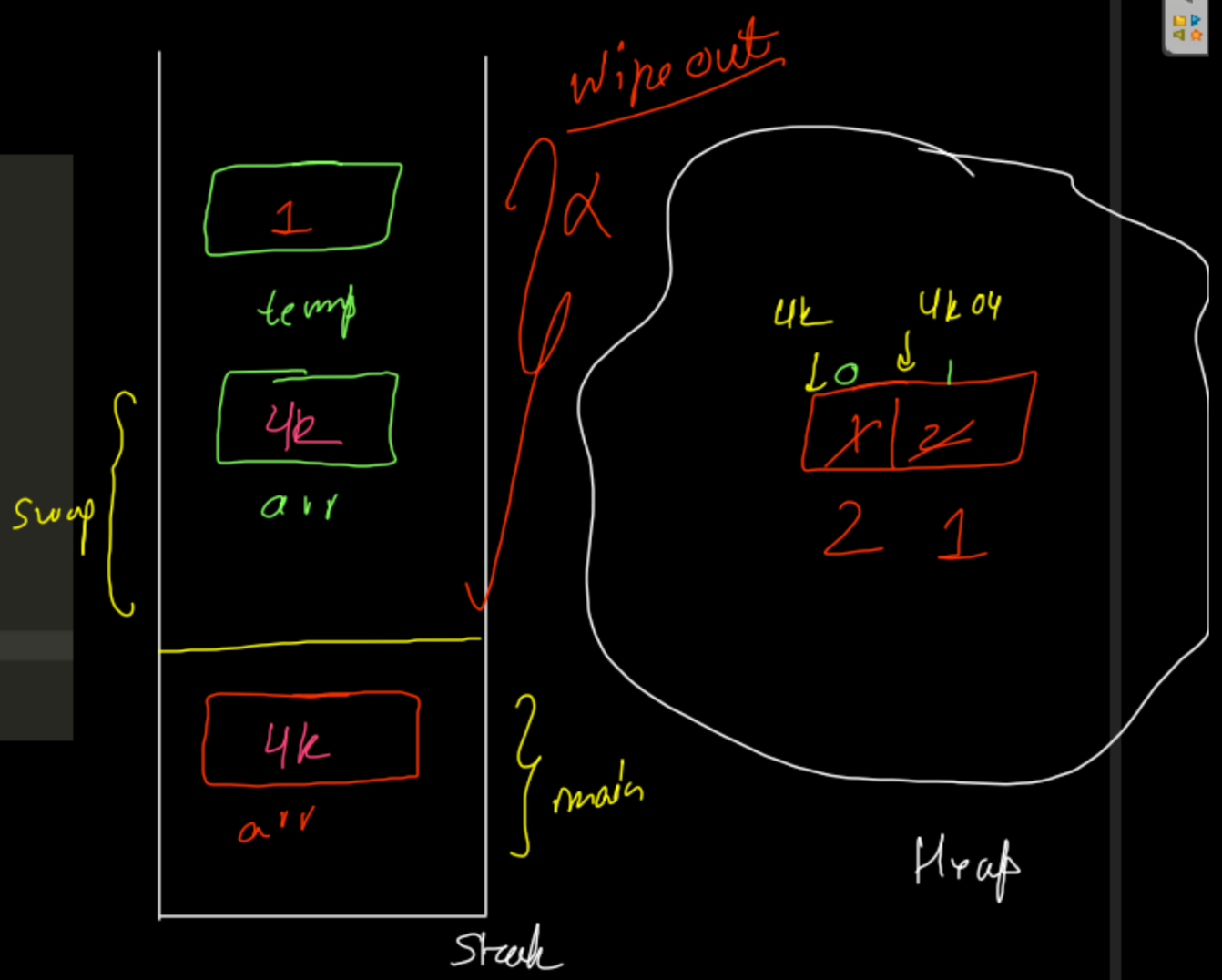
4k + 4*3 = 4k+12

4a12

Array as a function Parameter

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr = {1,2}; //another method for array creation

        System.out.println("arr[0] = " + arr[0] + " arr[1]" + arr[1]);

        //function call
        swap(arr);

        System.out.println("arr[0] = " + arr[0] + " arr[1]" + arr[1]);
    }

    public static void swap(int[] arr) {
        int temp = arr[0];
        arr[0] = arr[1];
        arr[1] = temp;
    }
}
```

1 2 # This is shallow copy.



Deep Copy

(If it were to happen)

fun {

7203

arr

4k

arr

}

Stack

4k

0 1

1 2

7203

0 1

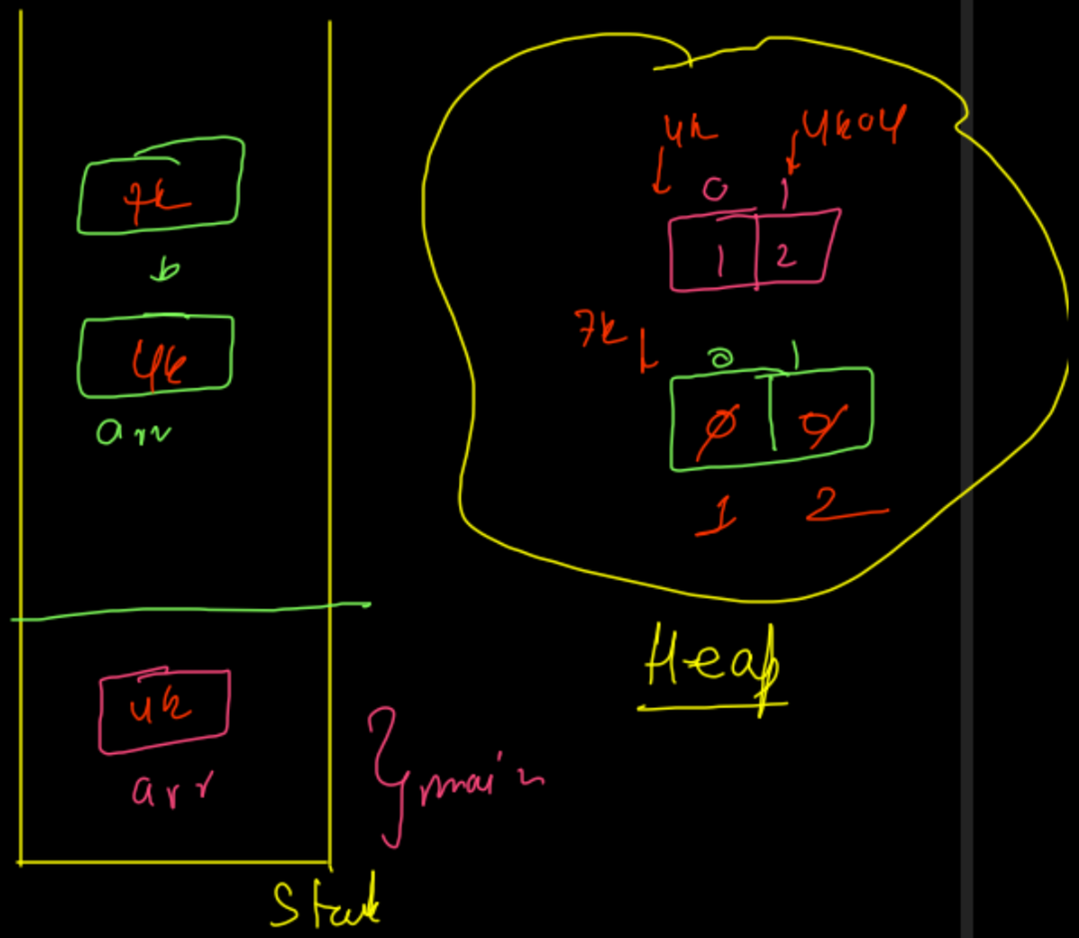
1 2

Heap

How to Deep Copy an Array

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] arr = {1,2}; //another method for array creation

        System.out.println("arr[0] = " + arr[0] + " arr[1]" + arr[1]);

        //function call
        swap(arr);

        System.out.println("arr[0] = " + arr[0] + " arr[1]" + arr[1]);
    }

    public static void swap(int[] arr) {

    }
}
```

```java
public static void swap(int[] arr) {
    int[] b = new int[arr.length];

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



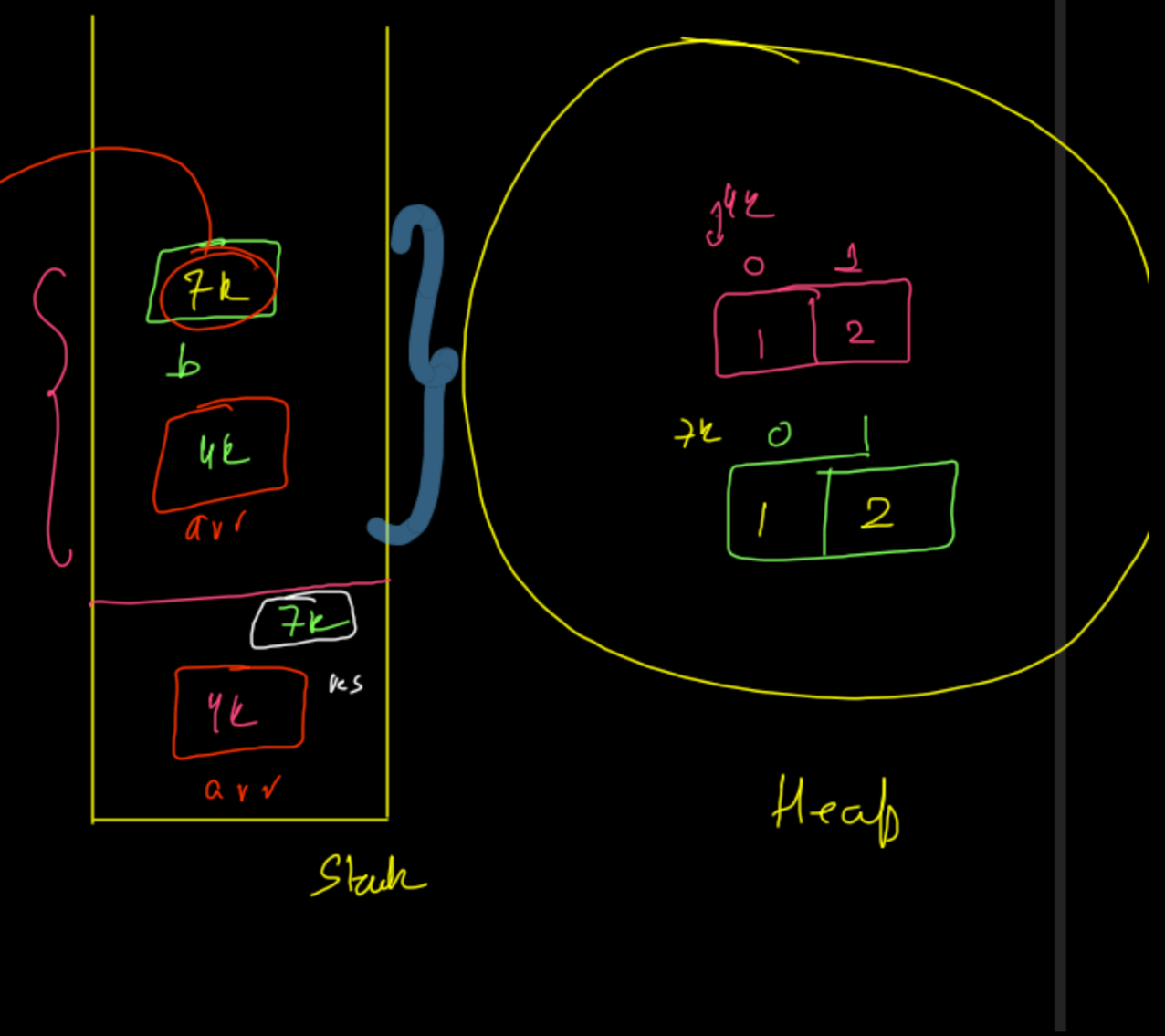

```
int [] a = new int [10];   → 4k
int [] b = new int [10000]; → 7k

fun(a);   (4k)
fun(b);   (7k)
```

Since only address is passed, both will take same time.

Array as a return type -

```java
public class Main {
    public static void main(String[] args) {
        int[] arr = {1,2}; //another method for array creation

        System.out.println("arr[0] = " + arr[0] + " arr[1] = " + arr[1]);

        //function call
        swap(arr);

        int[] res = swap(arr);

        System.out.println("res[0] = " + res[0] + " res[1] = " + res[1]);
    }

    public static int[] swap(int[] arr) {
        int[] b = new int[arr.length];

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

        return b;
    }
}
```

```
Finished in 77 ms
arr[0] = 1 arr[1] = 2
res[0] = 1 res[1] = 2
```



Search an Element in an array

Basic Accuracy: **50.0%** Submissions: **100k+** Points: **1**

Given an integer array and another integer element. The task is to find if the given element is present in array or not.

Linear Search O(N)

↳ Unsorted

for (i = 0 ; i < n ; i++) → return

0 1 2 3 4 5 6

{ 9, 8, 6, 3, 4, 7, 2 }

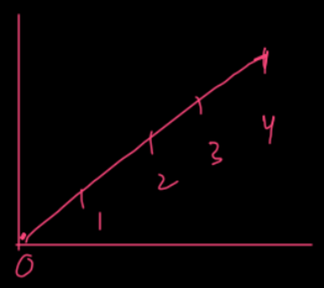
X = 7 # Successful Search

for (i = 0 ; i < n ; i++) i = arr.length

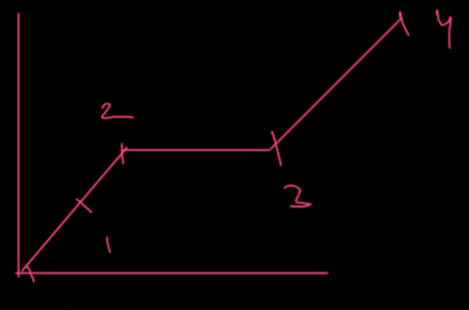
0 1 2 3 4 5 6

{ 9, 8, 6, 3, 4, 7, 2 }

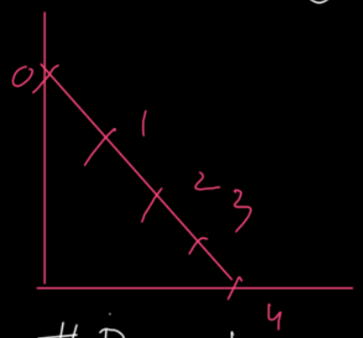
X = 5 # Unsuccessful Search



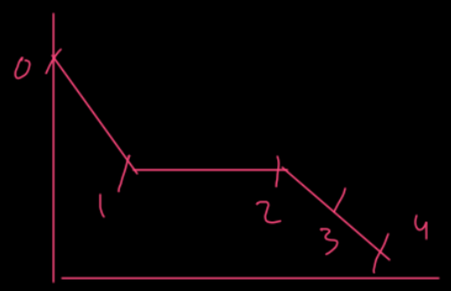
Increasing



Non-decreasing



Decreasing



Non increasing

```java
class Solution{

    static int search(int arr[], int N, int X)
    {

        // Your code here

        for(int i=0;i<N;i++) {
            if(arr[i] == X) return i;
        }

        return -1;
    }

}
```

Linear Search

→ It is usually used when the array (data) is unsorted.

→ Time Complexity

Best Case : O(1)

Worst Case : O(N)

Avg Case : O(N)

Find minimum and maximum element in an array

Basic Accuracy: **59.4%** Submissions: **100k+** Points: **1**

Given an array **A** of size **N** of integers. Your task is to find the **minimum and maximum** elements in the array.

$min = \infty$
$max = -\infty$
} Identities

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

Identity is a term using which when we perform an operation with N (any no), we get N itself.

0 → additive identity.

1 → multiplicative identity

N + 0 = N

N * 1 = N

Minimum Identity

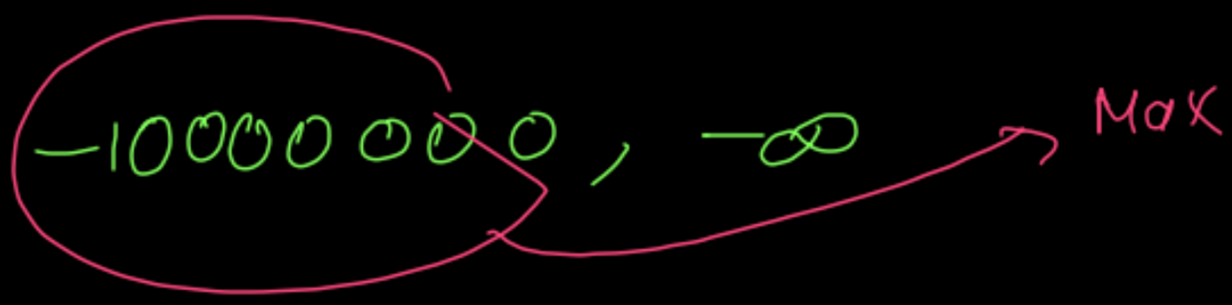
∞ → is the min identity

$100, \infty \rightarrow 100$

$100000, \infty \rightarrow 100000$

Maximum Identity

$-\infty$ → is the max identity



max = ~~-∞~~ 8 9

min = ~~∞~~ 8 6 3 2

Max & min store the max & min values so far.

```java
class Compute
{
    static pair getMinMax(long a[], long n)
    {
        //Write your code here
        long max = Long.MIN_VALUE;
        long min = Long.MAX_VALUE;

        for(int i=0;i<a.length;i++) {
            if(a[i] > max) max = a[i];
            if(a[i] < min) min = a[i];
        }

        return new pair(min,max);
    }
}
```

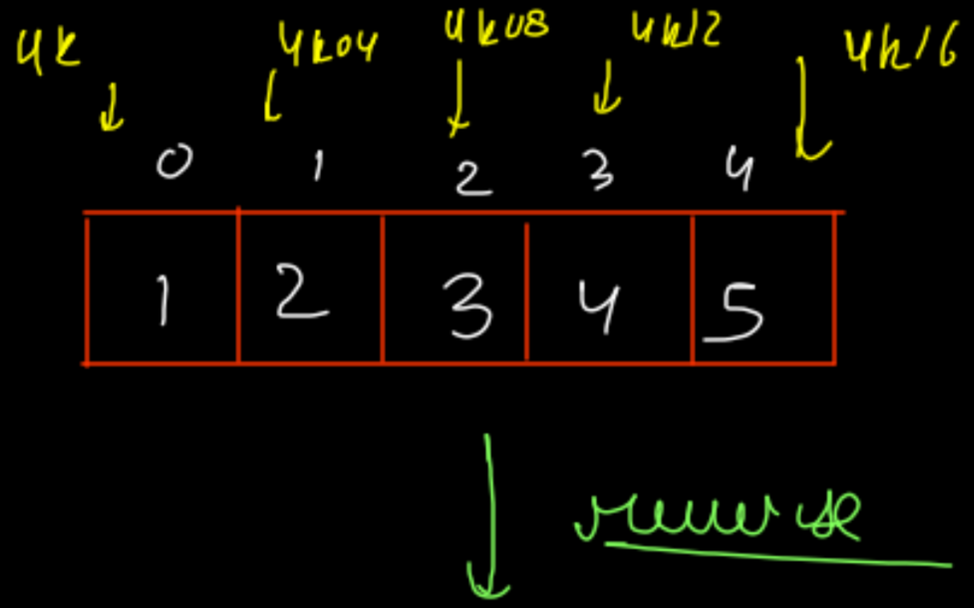
Reverse an Array

School Accuracy: **40.0%** Submissions: **100k+** Points: **0**

Given an array **A** of size **N**, print the reverse of it.

Example:

Input:
1
4
1 2 3 4
Output:
4 3 2 1



reverse

lo hi

0 1 2 3 4

1 2 3 4 5

5 4 2 1

```
int temp = a[lo];
a[lo] = a[hi];
a[hi] = temp;
```

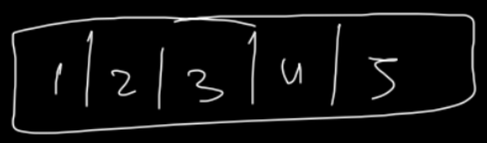
hi · lo

0 1 2 3 4 5

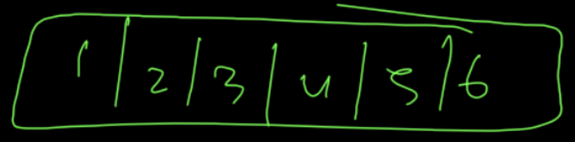
{1, 2, 3, 4, 5, 6}

6 5 4 3 2 1

2 (T)
5 (N)
1 2 3 4 5
6
1 2 3 4 5 6



5 4 3 2 1



6 5 4 3 2 1

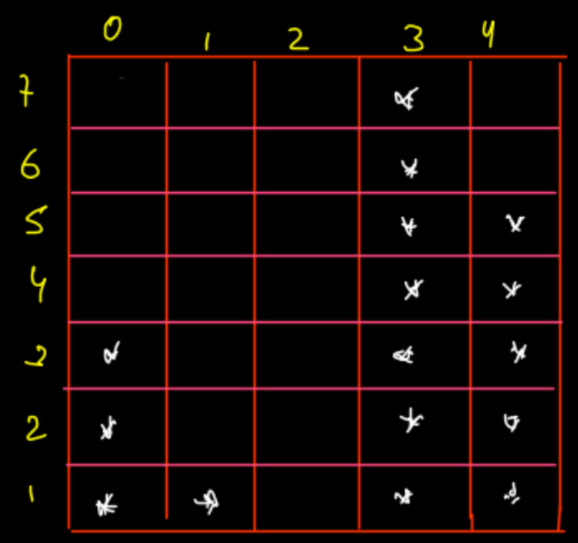
```java
public static void reverse(int[] arr) {
    int lo = 0;
    int hi = arr.length -1;

    while(lo < hi) {
        int temp = arr[lo];
        arr[lo] = arr[hi];
        arr[hi] = temp;
        lo++;
        hi--;
    }
}

public static void display(int[] arr) {
    for(int i=0;i<arr.length;i++) {
        System.out.print(arr[i] + " ");
    }
    System.out.println();
}
```

```java
public static void main (String[] args) {
    //code

    Scanner scn = new Scanner(System.in);
    int t = scn.nextInt();

    while(t-- > 0) {
        int n = scn.nextInt();
        int[] arr = new int[n];

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

        reverse(arr);
        display(arr);
    }
}
}
```

Bar chart

0 1 2 3 4

3 1 0 7 5




```
for(i=7; i>=1; i--)
{
    for(j=0; j<4; j++) {

    }
```

```java
public static void main(String[] args) throws Exception {
    // write your code here
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    int[] arr = new int[n];

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

    int max = Integer.MIN_VALUE;

    for(int i=0;i<n;i++) {
        max = (int)Math.max(max,arr[i]);
    }

    for(int i=max;i>=1;i--) {
        for(int j=0;j<n;j++) {
            if(arr[j] >= i) {
                System.out.print("*\t");
            } else {
                System.out.print("\t");
            }
        }
        System.out.println();
    }

 }

}
```

Lecture - 11 (Arrays Continued)

① Left Rotate
② Right Rotate
} Reversal Algorithm

③ Binary Search (A glimpse into BS topic)

④ Add 2 Arrays

⑤ Add 1 to an Array (H.W)

⑥ Subtract 2 Arrays

⑦ Subarrays of an Array

Left Rotate

Rotate Array

Basic Accuracy: **59.87%** Submissions: **100k+** Points: **1**

Given an unsorted array **arr[]** of size **N**, rotate it by **D** elements (Anti - clockwise).

Input:
The first line of the input contains **T** denoting the number of testcases. First line of each test case contains two space separated elements, **N** denoting the size of the array and an integer **D** denoting the number size of the rotation. Subsequent line will be the N space separated array elements.

{1, 2, 3, 4, 5}

k = 2

{3, 4, 5, 1, 2}

$$\{\overset{0}{1},\overset{1}{2},\overset{2}{3},\overset{3}{4},\overset{4}{5}\} \qquad N=5$$

$k=0$	$\{1,2,3,4,5\}$	$k=5$
$k=1$	$\{2,3,4,5,1\}$	$k=6$
$k=2$	$\{3,4,5,1,2\}$	$k=7$
$k=3$	$\{4,5,1,2,3\}$	$k=8$
$k=4$	$\{5,1,2,3,4\}$	$k=9$

$\#\ k = k \% N$

$\#\ k = k + N$

↓

if $k<0$

0 1 2 3 4
{ 1, 2, 3, 4, 5 }

k = 2

lo hi
① Reverse(arr, 0, k-1)

lo hi
② Reverse(arr, k, N-1)

lo hi
③ Reverse(arr, 0, N-1)

lo hi
0 1 2 3 4
{ 2, 1, 3, 4, 5 }

0 1 2 3 4
{ 2, 1, 5, 4, 3 }

0 1 2 3 4
{ 3, 4, 5, 1, 2 }

```java
public static void reverse(int[] arr,int lo, int hi) {
    while(lo < hi) {
        int temp = arr[lo];
        arr[lo] = arr[hi];
        arr[hi] = temp;
        lo++;
        hi--;
    }
}

public static void rotate(int[] arr, int k) {

    int n = arr.length;

    reverse(arr,0,k-1);
    reverse(arr,k,n-1);
    reverse(arr,0,n-1);
}

public static void display(int[] arr) {
    for(int i=0;i<arr.length;i++) {
        System.out.print(arr[i] + " ");
    }
    System.out.println();
}
```

```java
public static void main (String[] args) {
    //code

    Scanner scn = new Scanner(System.in);
    int t = scn.nextInt();

    while(t-- > 0) {
        int n = scn.nextInt();
        int k = scn.nextInt();

        int[] arr = new int[n];

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

        rotate(arr,k);
        display(arr);
    }
}
}
```

O(N)

189. Rotate Array

Medium 11203 1397 Add to List Share

Given an array, rotate the array to the right by `k` steps, where `k` is non-negative.

Example 1:

```
Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]
```

N-k N-1

0 1 2 3 4 5 6

{1, 2, 3, 4, 5, 6, 7} k = 3

0 1 2 3 4 5 6

{5, 6, 7, 1, 2, 3, 4}

0 N-k-1

Reverse(arr, N-k, N-1) → {1, 2, 3, 4, 7, 6, 5}

Reverse(arr, 0, N-k-1) → {4, 3, 2, 1, 7, 6, 5}

Reverse(arr, 0, N-1) → {5, 6, 7, 1, 2, 3, 4}

N-k-1 N-k N-1

0 1 2 3 4 5 6

{1, 2, 3, 4, 5, 6, 7} k = 3

{5, 6, 7, 1, 2, 3, 4}

```java
class Solution {

     public void reverse(int[] arr,int lo, int hi) {
        while(lo < hi) {
            int temp = arr[lo];
            arr[lo] = arr[hi];
            arr[hi] = temp;
            lo++;
            hi--;
        }
    }

    public void rotate(int[] nums, int k) {

        int n = nums.length;

        k = k%n;

        reverse(nums,n-k,n-1);
        reverse(nums,0,n-k-1);
        reverse(nums,0,n-1);
    }
}
```

Binary Search

704. Binary Search

Easy 6477 138 Add to List Share

Given an array of integers `nums` which is sorted in ascending order, and an integer `target`, write a function to search `target` in `nums`. If `target` exists, then return its index. Otherwise, return `-1`.

You must write an algorithm with `O(log n)` runtime complexity.

while (lo < hi)

target = 40

lo ~~lo~~

mid ↓ lo

hi

0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

$$mid = \frac{(lo + hi)}{2} = \frac{0 + 10}{2} = 5$$

if (arr[mid] < target)

lo = mid + 1;

lo hi mid ↓

0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

$$mid = \frac{lo + hi}{2} = \frac{16}{2} = 8$$

if (arr[mid] > target)

hi = mid - 1;

mid lo
lo hi

0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

$$mid = \frac{lo + hi}{2} = \frac{6+7}{2} = \frac{13}{2} = 6$$

$$lo = mid + 1$$

lo mid
hi

0	1	2	3	4	5	6	7	8	9	10
-40	-30	-20	-10	10	20	30	40	60	70	80

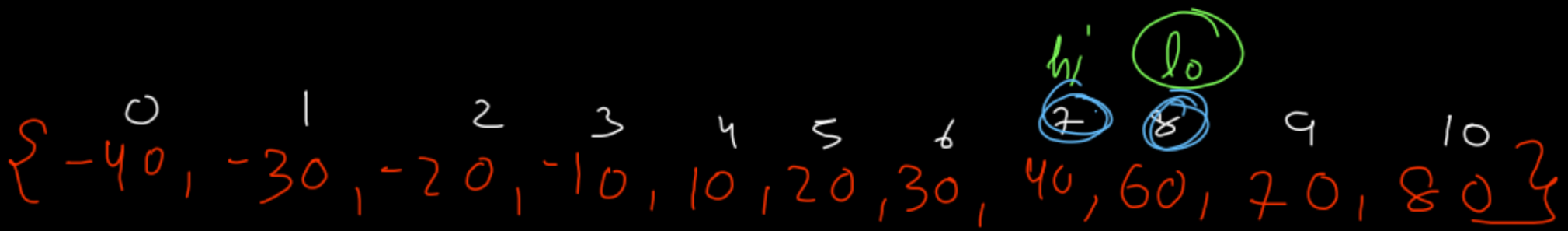
$$mid = \frac{lo + hi}{2} = \frac{7+7}{2} = 7$$

$\approx \log_2 N$

```
if (arr[mid] == target)

return mid;
```

Unsuccessful Search

target = 50



mid = $\frac{0+10}{2}$ = 5

1

while(lo ≤ hi)

```java
class Solution {
    public int search(int[] nums, int target) {
        int lo = 0;
        int hi = nums.length -1;

        while(lo <= hi) {
            int mid = (lo + hi)/2;

            if(nums[mid] == target) return mid;
            else if(nums[mid] < target) lo = mid + 1;
            else hi = mid -1;
        }

        return -1; //only hits when unsucessful search
    }
}
```

Overflow Condition

$$mid = (lo + hi) / 2$$

lo → ∞ (Out of the Range)

hi → ∞ (Out of the Range)

```java
class Solution {
    public int search(int[] nums, int target) {
        int lo = 0;
        int hi = nums.length -1;

        while(lo <= hi) {
            int mid = lo + (hi-lo)/2;

            if(nums[mid] == target) return mid;
            else if(nums[mid] < target) lo = mid + 1;
            else hi = mid -1;
        }

        return -1; //only hits when unsucessful search
    }
}
```

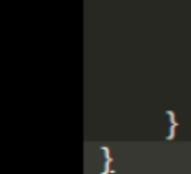
Binary Search

Note: This is not complete.

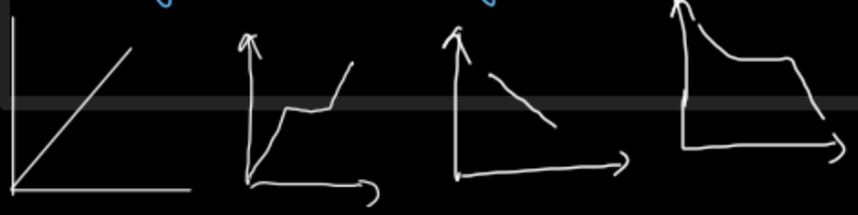
We will complete it when we will study BS.

Monotonic Array

- Increasing
- Decreasing
- Binary
  - 000... 111..
  - 111... 000...



- Strictly Increasing
- Non Strictly Increasing
- Strictly Decreasing
- Non Strictly Decreasing



Sum of two numbers represented as arrays

Easy Accuracy: **43.7%** Submissions: **5244** Points: **2**

Given two numbers represented by two different arrays **A** and **B**. The task is to find the sum array. The sum array is an array representation of addition of two input arrays.

Example 1:

Input:
N = 3, M = 3
A[] = {5, 6, 3}
B[] = {8, 4, 2}
Output: 1 4 0 5
Explanation: As 563 + 842 = 1405.

i

0 { 9, 9, 3 } → a

0 1 2

j i

0 { 1, 5 } → b

0 1

ca = 1

if (a[i] + b[j] + ca < 10) {

c[k] = a[i] + b[j] + ca;

ca = 0

}

k

0 1 2 3

{ 1 0 0 8 } → c

if (a[i] + b[j] + ca >= 10) {

c[k] = (a[i] + b[j] + ca) % 10;

ca = (a[i] + b[j] + ca) / 10

}

Array can contain one 0 in the beginning.

{9,3,0}

{2,3}

{0,9,5,3}

{9,5,3}

{9,5,3}

```java
int[] sum(int[] a, int[] b) {
    int size = (a.length > b.length) ? a.length + 1 : b.length +
    int[] res = new int[size];

    int i = a.length -1, j = b.length-1, k = size-1;
    int ca = 0;

    while(i >= 0 || j >= 0 || ca > 0) {
        int d1 = (i >= 0) ? a[i] : 0;
        int d2 = (j >= 0) ? b[j] : 0;

        if(d1 + d2 + ca < 10) {
            res[k] = d1 + d2 + ca;
            ca = 0;
        } else {
            res[k] = (d1 + d2 + ca) % 10;
            ca = (d1 + d2 + ca) / 10;
        }

        i--;j--;k--;
    }

    return res;

}

ArrayList<Integer> findSum(int a[], int b[]) {
    // code here
    int[] res = sum(a,b);

    ArrayList<Integer> list = new ArrayList<>();

    for(int i=0;i<res.length;i++) {
        if(i == 0 && res[i] == 0) {
            continue;
        } else {
            list.add(res[i]);
        }
    }

    return list;
```

Lecture - 12

Subtraction of 2 Arrays

Subarrays of an Array

2-D Arrays (Matrices)

Demo and Memory Mapping Concepts.

#Subtraction of 2 Arrays

Difference Of Two Arrays

Easy

Prev Next

1. You are given a number n1, representing the size of array a1.
2. You are given n1 numbers, representing elements of array a1.
3. You are given a number n2, representing the size of array a2.
4. You are given n2 numbers, representing elements of array a2.
5. The two arrays represent digits of two numbers.
6. You are required to find the difference of two numbers represented by two arrays and print the arrays. a2 - a1

Assumption - number represented by a2 is greater.

j j0 j1 j2 j3

{1, 0, 2, 3} → a_2

i i0 i1 i2

{4, 5, 1} → a_1

$b_o = \phi - 1$

k k0 k1 k2 k3

{0 5 7 2}

{1, 0, 2, 3}

{0}

```
if (d2 - d1 + bo < 0) {
    d = d2 - d1 + bo + 10;
    bo = -1;
} else {
    d = d2 - d1 + bo;
    bo = 0;
}
```

} This is the condition when we can have the max size of res array. It is = size of larger array

0 1 2 3 4
{1, 0, 0, 0, 0}

0 1 2
{9, 0, 0}

{0 9 1 0 0}

{1, 0, 0, 0}

{9, 9, 9}

idx idx idx idx
{0, 0, 0, 2}

There can be multiple 0s at the start of res.

If both Nos are equal

{9, 9, 8}

{9, 9, 8}

~~idx~~ ~~idx~~ ~~idx~~

{0, 0, 0}

→ index out of bound

```java
public static int[] diff(int[] arr1,int[] arr2) {
    int[] res = new int[arr2.length]; //size of the larger array
    int i = arr1.length-1, j = arr2.length -1, k = res.length-1;

    int bo = 0;

    while(j >= 0) {
        int d1 = (i >= 0) ? arr1[i] : 0; //this is to be safe from index out of bounds
        int d2 = arr2[j];

        if(d2 - d1 + bo < 0) {
            res[k] = d2 - d1 + bo + 10;
            bo = -1;
        } else {
            res[k] = d2 - d1 + bo;
            bo = 0;
        }

        i--;j--;k--;
    }

    return res;
`
    //differnce
    int[] res = diff(arr1,arr2);

    int idx = 0;

    while(idx < res.length && res[idx] == 0) {
        idx++;
    }

    if(idx == res.length) {
        System.out.println("0");
    }

    while(idx < res.length) {
        System.out.println(res[idx]);
        idx++;
    }
```

Subarrays of an Array

Subarray Problem

Easy

Prev Next

1. You are given an array of size 'n' and n elements of the same array.
2. You are required to find and print all the subarrays of the given array.
3. Each subarray should be space seperated and on a seperate lines.

Refer to sample input and output.

Sample Input

```
3
10
20
30
```

Sample Output

```
10
10 20
10 20 30
20
20 30
30
```

$\{10, 20, 30\}$ (indices 0, 1, 2)

$\{10, 30\}$ ✗ this is not a subarray.

Subarray is a part of an array that contains elements in continuation.

$\{10, 20, 30\}$ (indices 0, 1, 2)

1 $\{10\}$
2 $\{10, 20\}$
3 $\{10, 20, 30\}$ } Starting from index 0

4 $\{20\}$
5 $\{20, 30\}$ } Starting from index 1

6 $\{30\}$ } Starting from index 2

Number of subarrays of an Array

$$= \frac{N(N+1)}{2}$$

Here N is the size of the array

n = 3 (sp) (ep)

0 1 2

{10, 20, 30}

1 {10}

2 {10, 20}

3 {10, 20, 30}

} Starting from index 0

4 {20}

5 {20, 30}

} Starting from index 1

6 {30}

} Starting from index 2

```
for(int sp = 0; sp < n; sp++)
{
  for(int ep = i; ep < n; ep++)
  {
    for(int k = sp; k <= ep; k++)
    {
      Print( arr[k] + " ");
    }
  }
}
```

→ Leave a line

i 1 SP

{ 10, 20, 30, 40, 50, 60, 70 }

30

30, 40

30, 40, 50

30, 40, 50, 60

30, 40, 50, 60, 70

```java
    for(int sp=0;sp<arr.length;sp++) {
        for(int ep=sp;ep<arr.length;ep++) {
            for(int k=sp;k<=ep;k++) {
                System.out.print(arr[k] + "\t");
            }
            System.out.println();
        }
    }
}
```

All subsets

All subarrays

All paths

} Brute force

2-D Arrays

	1	2	3	4
1	a_{11}	a_{12}	a_{13}	a_{14}
2	a_{21}	a_{22}	a_{23}	a_{24}
3	a_{31}	a_{32}	a_{33}	a_{34}

↓

Matrix (Maths)

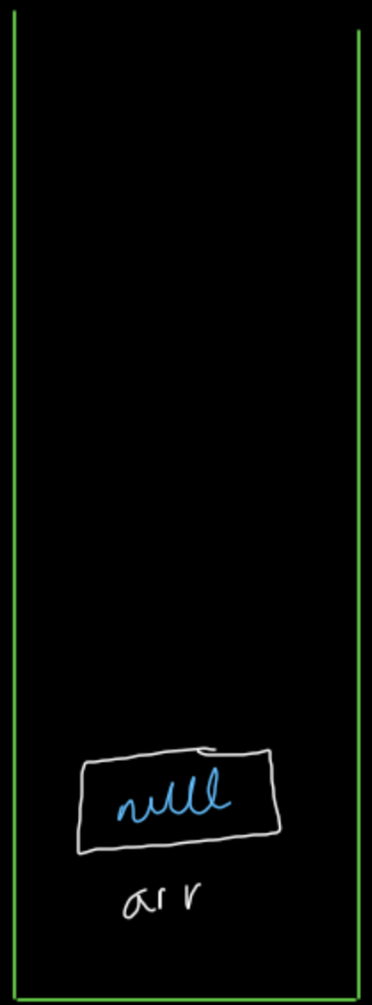
	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{20}	a_{21}	a_{22}	a_{23}

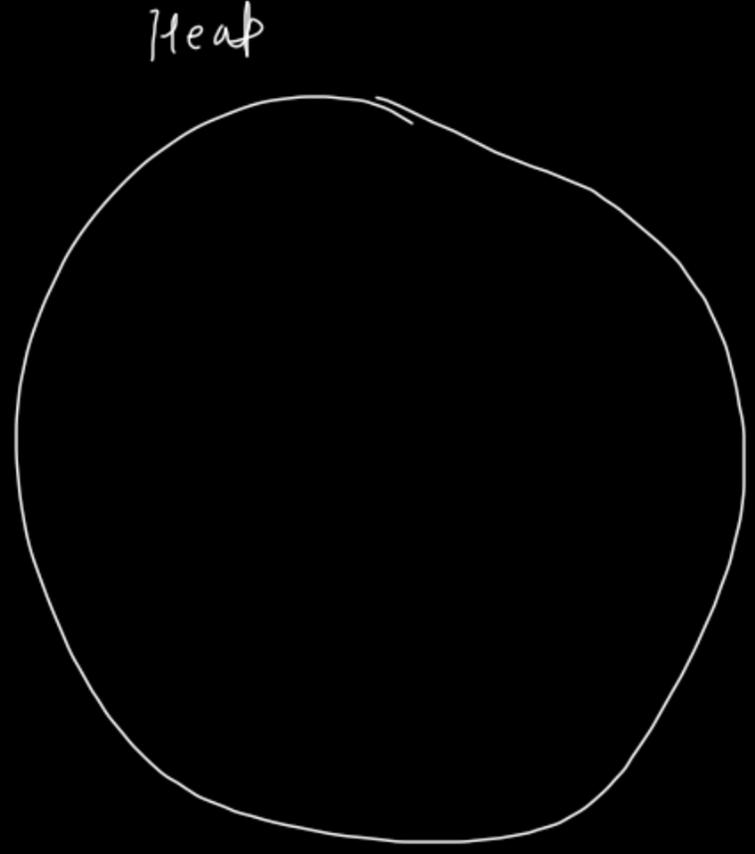
↓

2-D Array

Declare

```
int[][] arr;
```

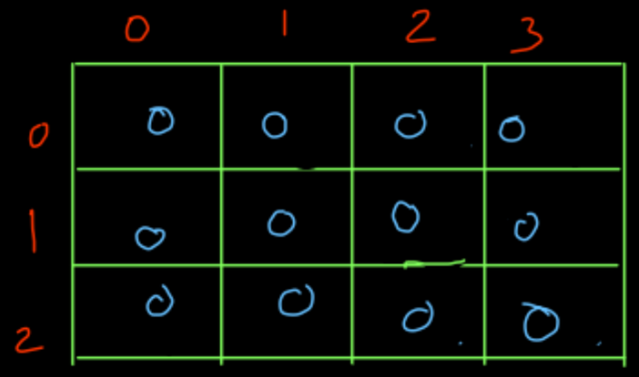




Initialization

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

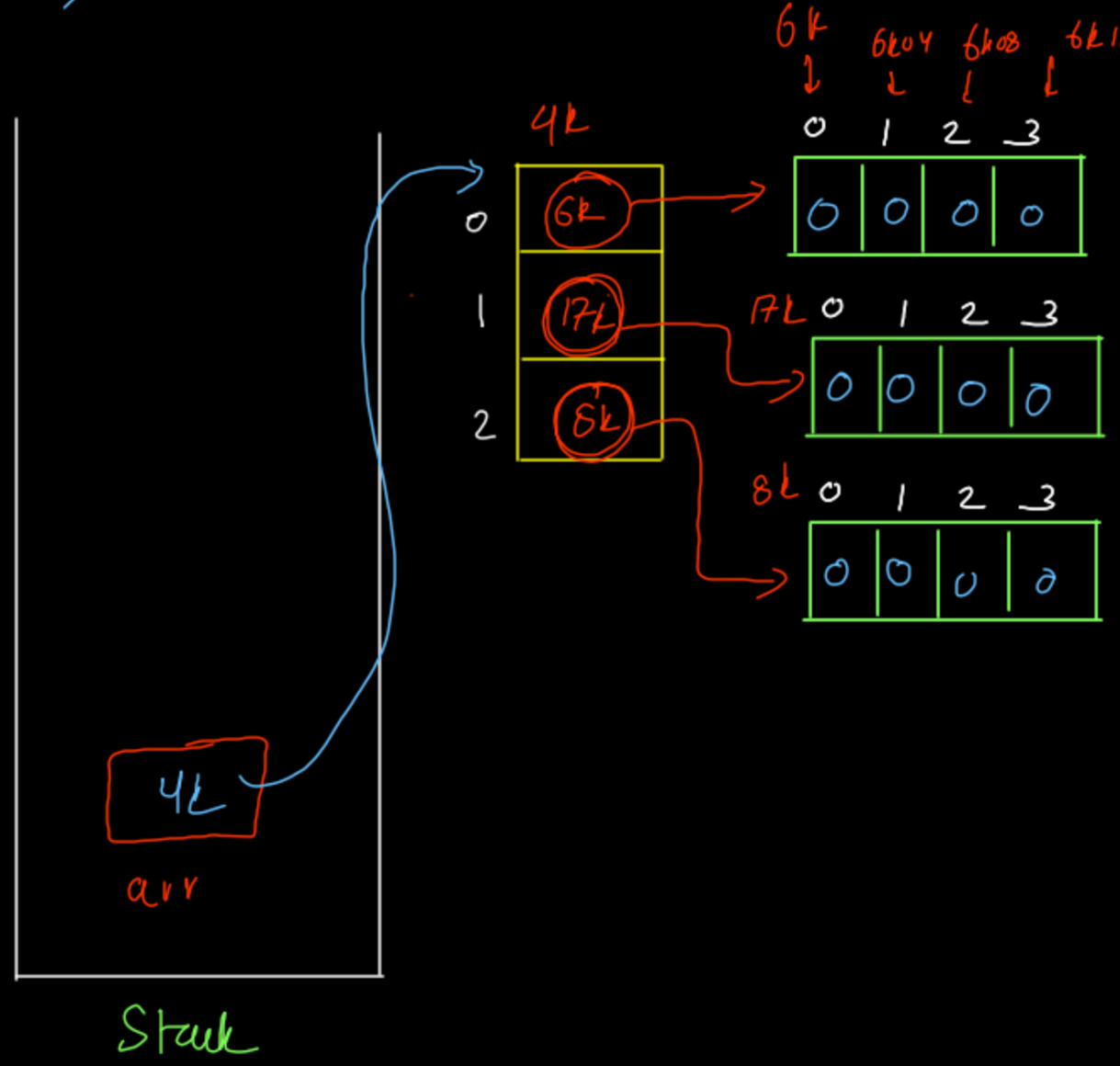
row col

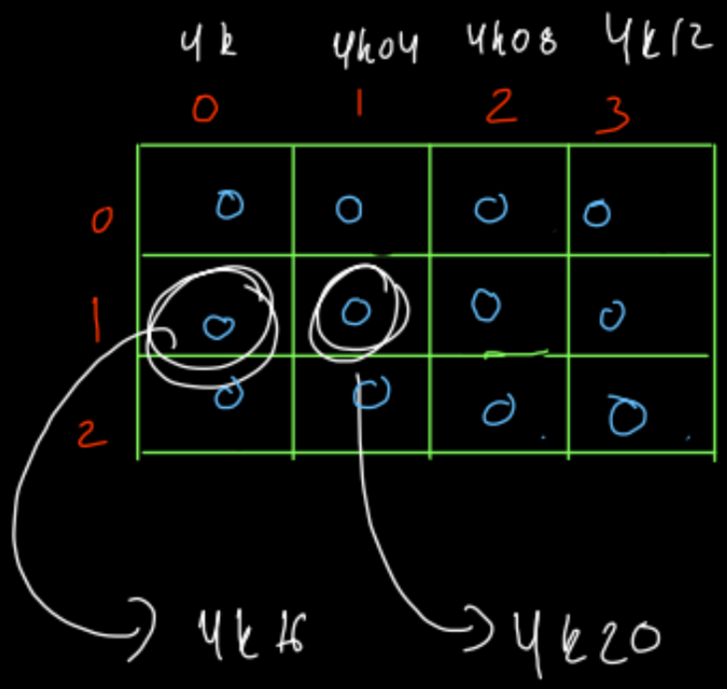


Memory Mapping (Initialization)

```
int[][] arr;  ①
arr = new int[3][4];
```

row col





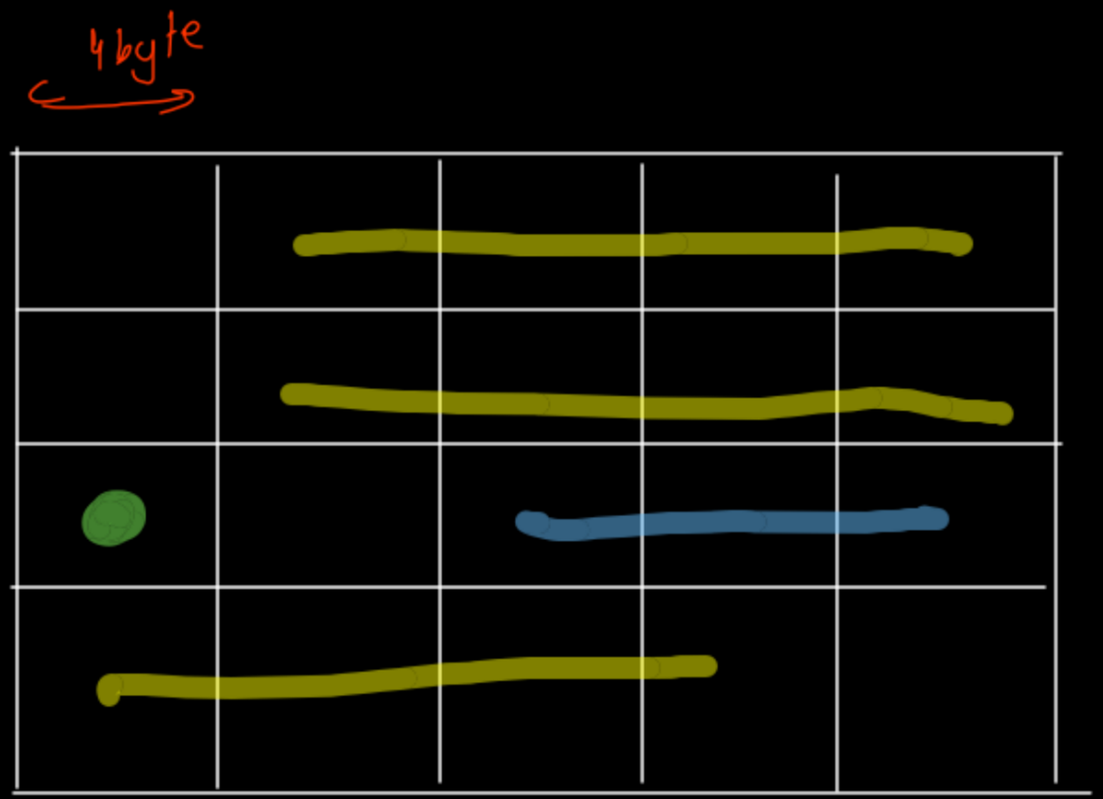
→ 1-D Array of Size 12

4 × 12 = 48 bytes

C++ Row-Major Mapping

fragmentation

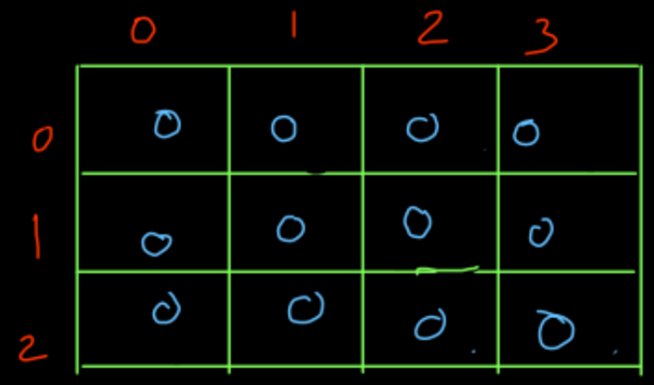
In C++ fragmentation is more likely to happen.



as there is contiguos memory allocation throughout the matrix.

In Java, it is rare to happen

Length Property



arr

arr.length = gives the length of array of rows (no of rows)

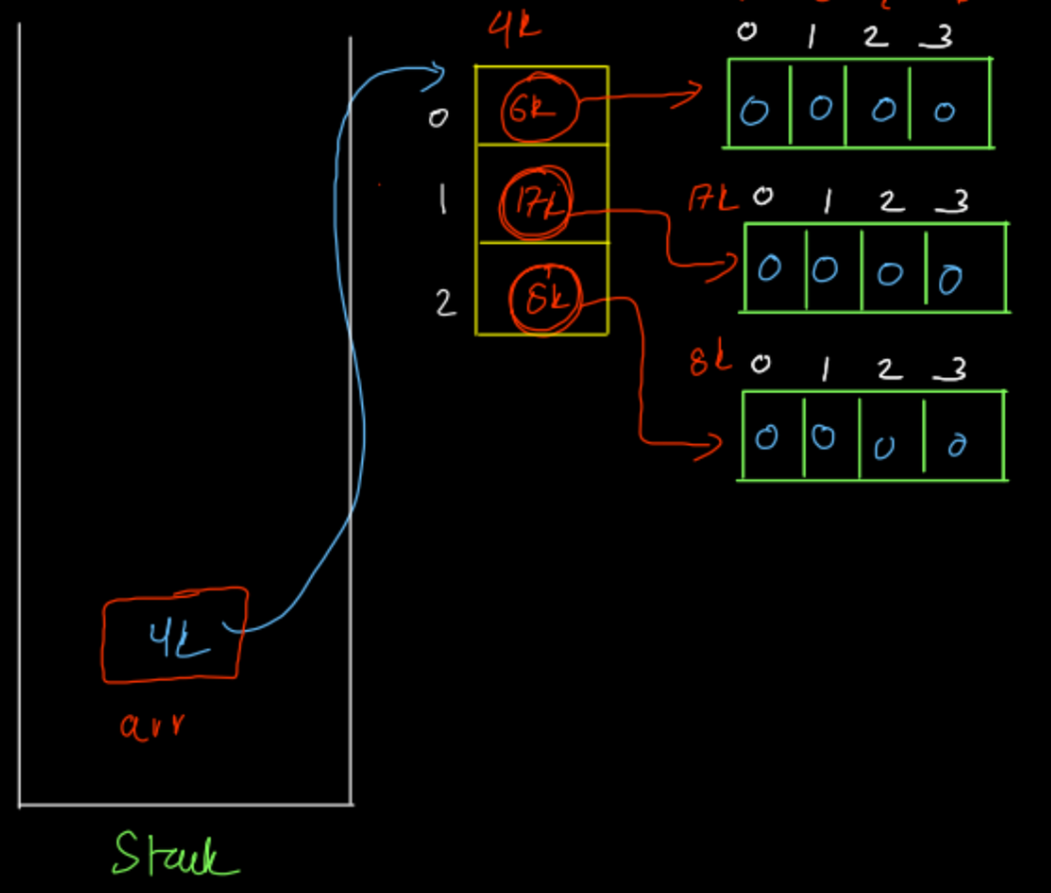
arr[0].length = gives the no of cols.

Direct Access

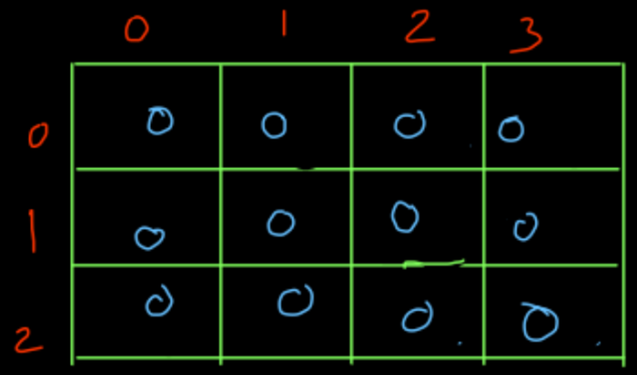
```
int[][] arr;          ①
arr = new int[3][4];
        ↑   ↑
       row  col
```

arr[1][2] = ?

LOC = base + DT * index
      add



Traverse




```java
public static void main(String[] args) throws Exception {
    // write your code here
    Scanner scn = new Scanner(System.in);
    int row = scn.nextInt();
    int col = scn.nextInt();

    //declaration + initialization
    int[][] mat = new int[row][col];

    for(int i=0;i<mat.length;i++) {
        for(int j=0;j<mat[0].length;j++) {
            mat[i][j] = scn.nextInt();
        }
    }

    for(int i=0;i<mat.length;i++) {
        for(int j=0;j<mat[0].length;j++) {
            System.out.print(mat[i][j] + " ");
        }
        System.out.println();
    }
}
```

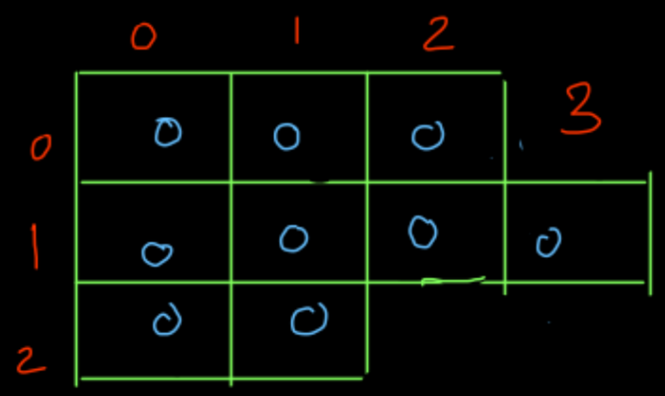
Lecture : 13

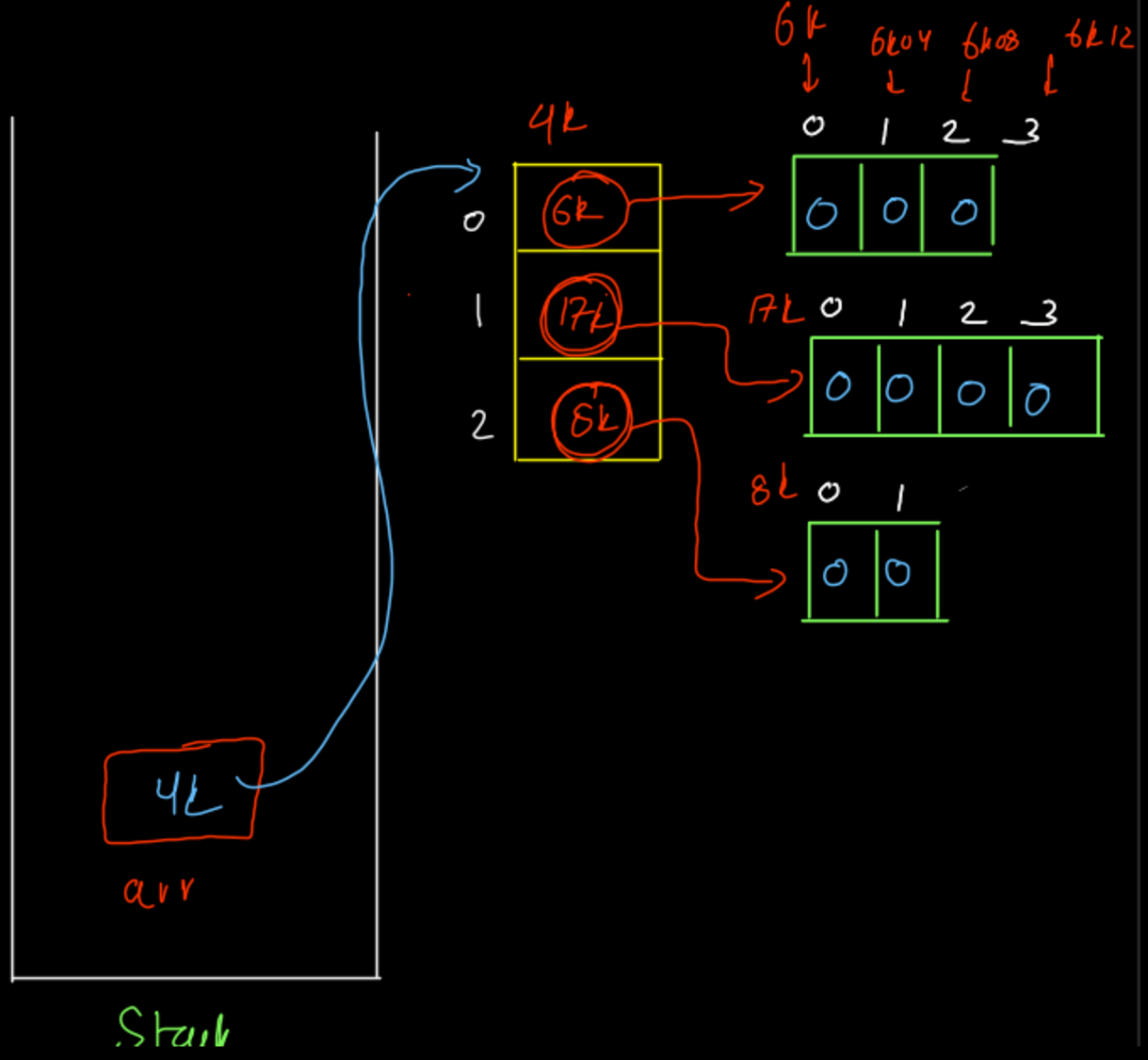
Jagged Array

2-D Array as a function parameter & return type (Shallow & Deep copy)

1. Matrix Multiplication
2. Rotate Image

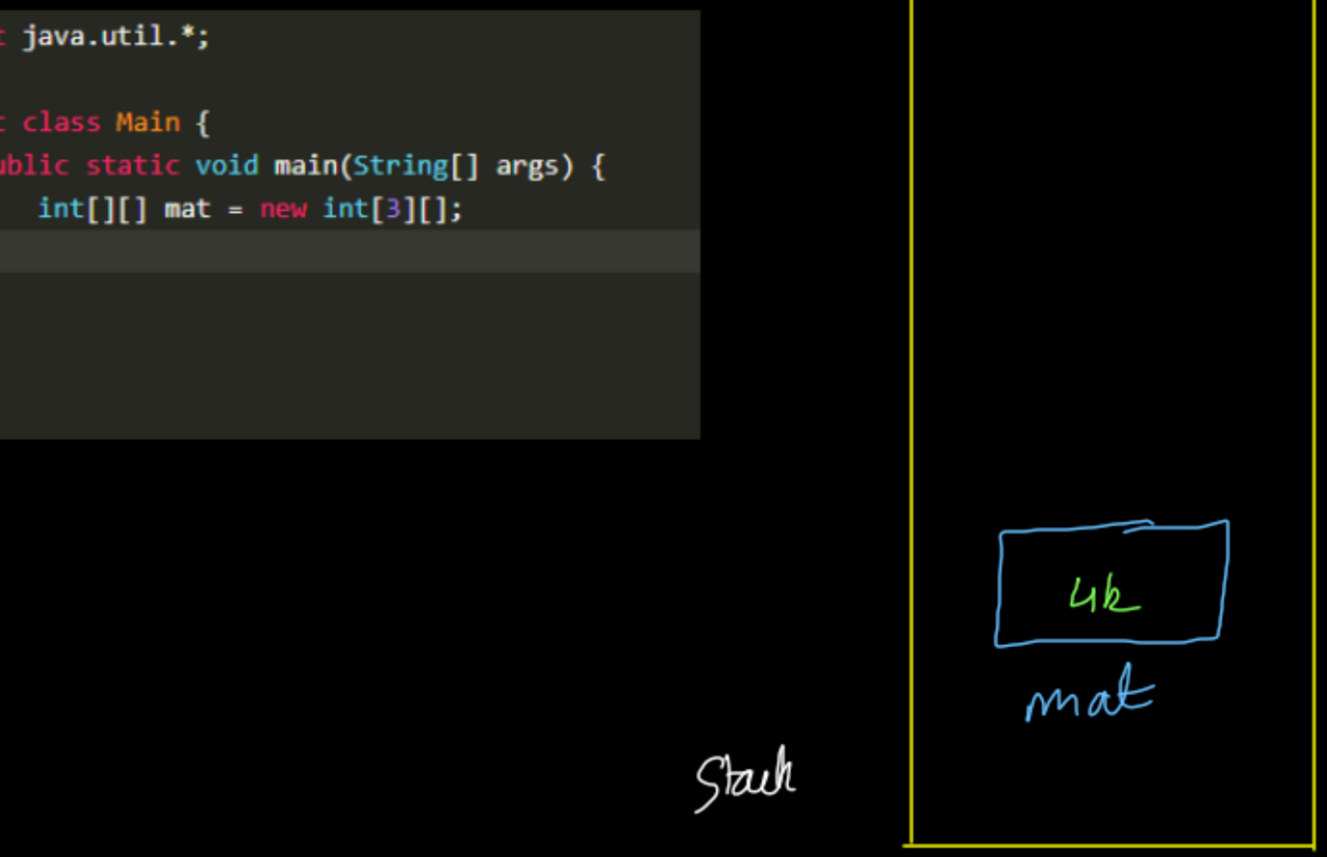
Jagged Array





Stack

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];

    }
}

```



LOC = BA + DT * index

= BA

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;

    }
}
```

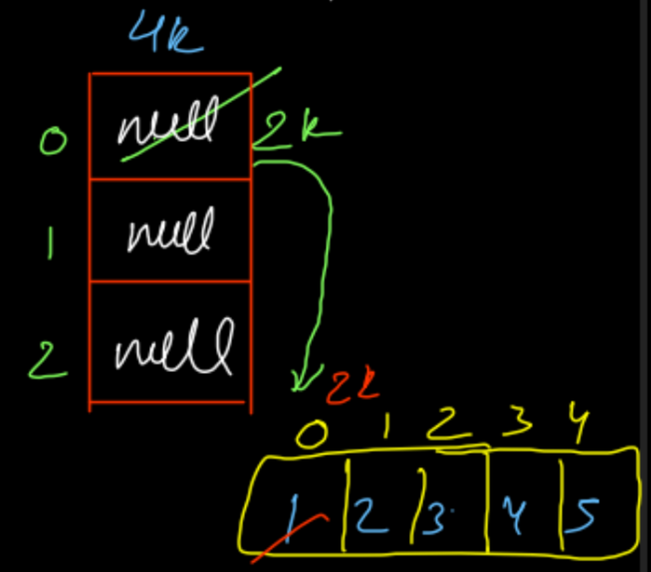
mat[0] = arr1

arr1[0] = 10

ash

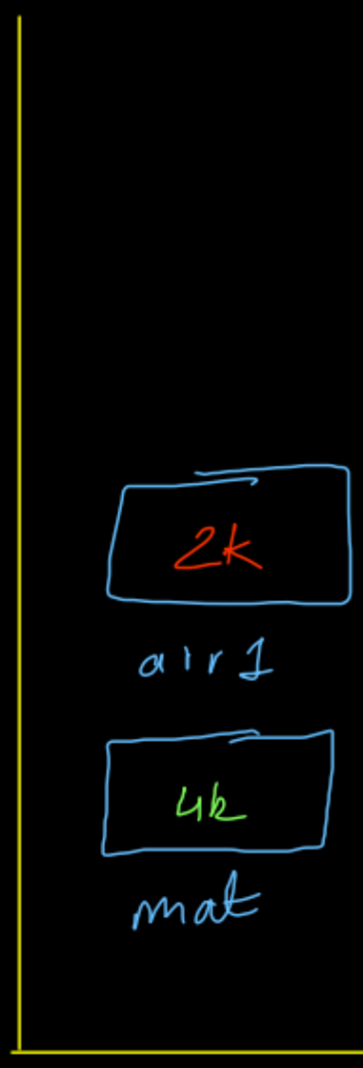
arr1

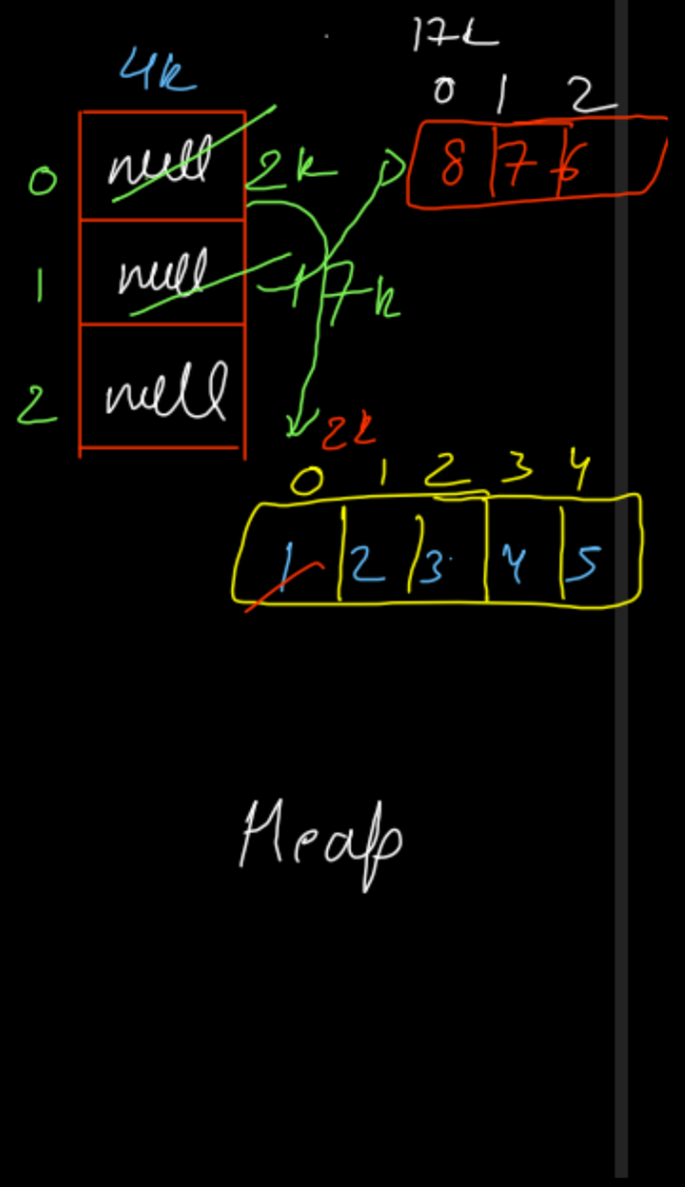
mat



Heap

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;

        mat[1] = new int[]{8,7,6};
    }
}

```

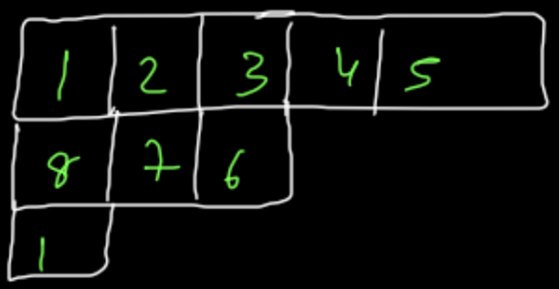


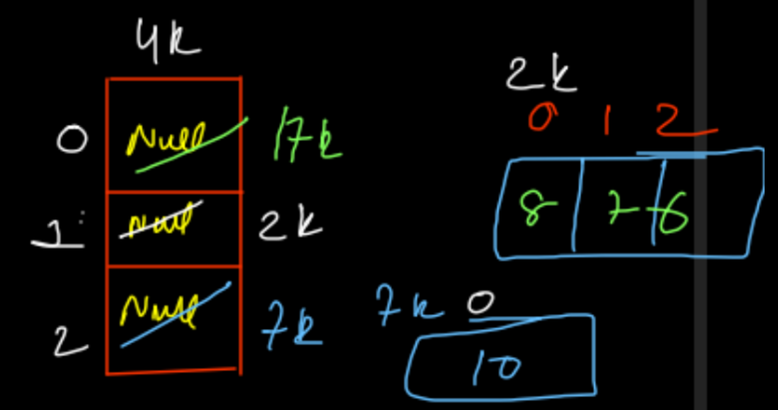


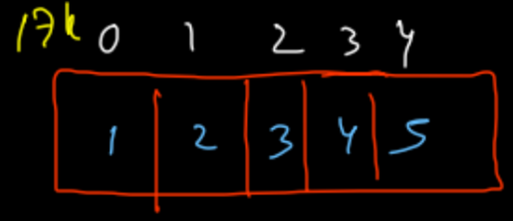
```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;

        mat[1] = {8,7,6};
    }
}
```

This is syntax error.

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;
        mat[1] = new int[]{8,7,6};
        mat[2] = new int[]{10};


    }
}
```



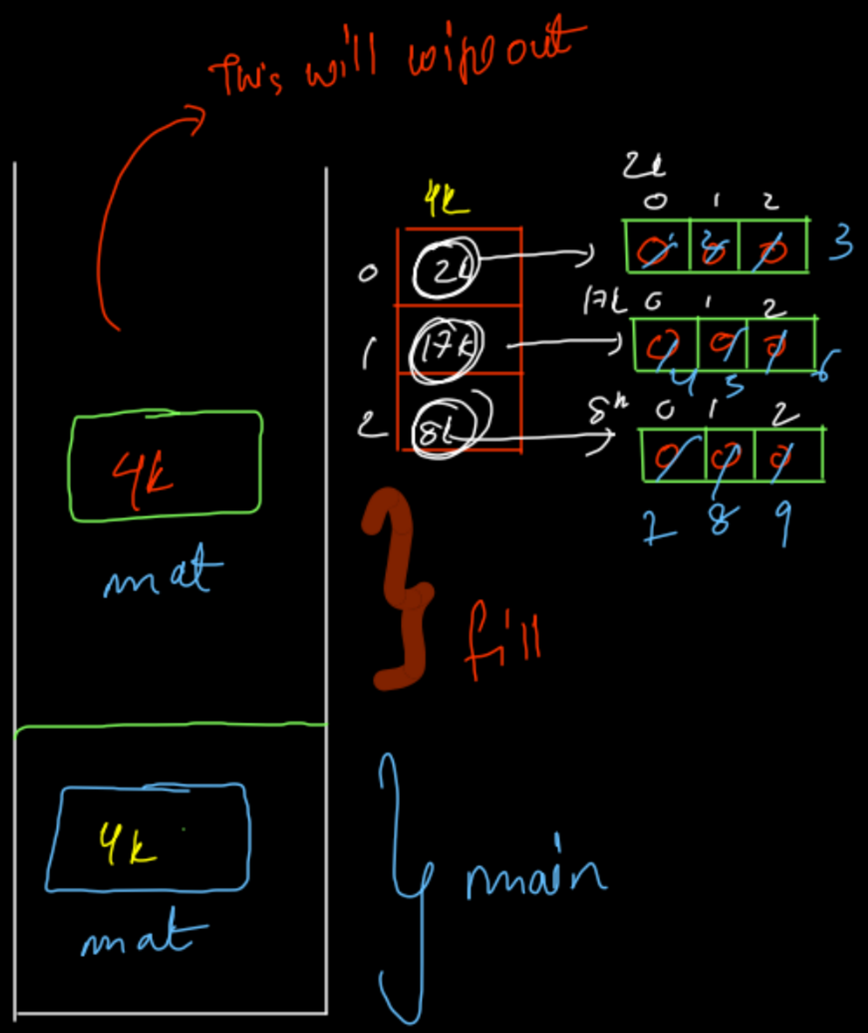




```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int[][] mat = new int[3][];
        int[] arr1 = {1,2,3,4,5};
        mat[0] = arr1;
        mat[1] = new int[]{8,7,6};
        mat[2] = new int[]{10};

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

```

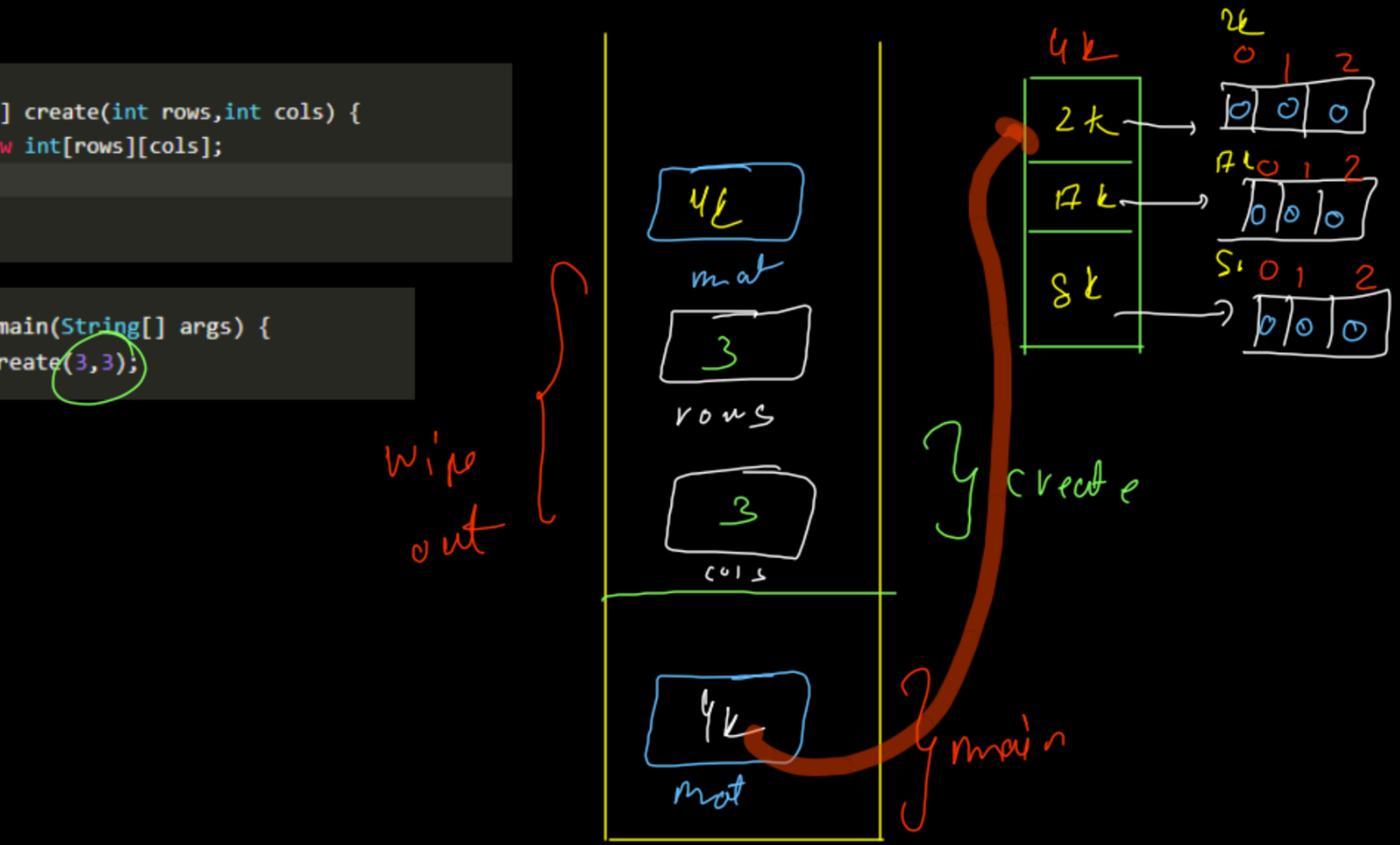
2-D Array as a Method Parameter

```java
public static void fill(int[][] mat) {

    int count = 1;

    for(int i=0;i<mat.length;i++) {
        for(int j=0;j<mat[0].length;j++) {
            mat[i][j] = count++;
        }
    }
}

public static void main(String[] args) {
    int[][] mat = new int[3][3];

    fill(mat);

    for(int i=0;i<mat.length;i++) {
        for(int j=0;j<mat[0].length;j++) {
            System.out.print(mat[i][j] + " ");
        }
    }
}
```



```java
public static int[][] create(int rows,int cols) {
    int[][] mat = new int[rows][cols];
    return mat;
}
```

```java
public static void main(String[] args) {
    int[][] mat = create(3,3);
```



Matrix Multiplication

Easy

Prev Next

1. You are given a number n1, representing the number of rows of 1st matrix.
2. You are given a number m1, representing the number of columns of 1st matrix.
3. You are given n1*m1 numbers, representing elements of 2d array a1.
4. You are given a number n2, representing the number of rows of 2nd matrix.
5. You are given a number m2, representing the number of columns of 2nd matrix.
6. You are given n2*m2 numbers, representing elements of 2d array a2.
7. If the two arrays representing two matrices of dimensions n1 * m1 and n2 * m2 can be multiplied, display the contents of prd array as specified in output Format.
8. If the two arrays can't be multiplied, print "Invalid input".

$$\begin{array}{c} \\ 0 \\ 1 \end{array}\begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 1 \end{bmatrix} \end{array}_{2\times 3} \qquad \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 0 \end{bmatrix} \end{array}_{3\times 3}$$

Matrix multiplication is not commutative

$$A * B \neq B * A$$

$$\begin{array}{c} \\ 0 \\ 1 \end{array}\begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 1 \end{bmatrix} \end{array}_{2\times 3} \qquad \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 0 \end{bmatrix} \end{array}_{3\times 3}$$

r_1 c_1 : $2 \times \textcircled{3}$

r_2 c_2 : $\textcircled{3} \times 3$

for matrix multiplication, it is a rule that $c_1 = r_2$.

$$\begin{matrix} & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 \\ 1 & 2 & 0 & 1 \end{matrix}_{2\times 3} \qquad \begin{matrix} & 0 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{matrix}_{3\times 3} = \begin{matrix} & 0 & 1 & 2 \\ 0 & 2 & 4 & 5 \\ 1 & 3 & 2 & 2 \end{matrix}_{2\times 3}$$

$$\begin{matrix} & 0 & 1 & 2 \\ 0 & a_{00} & a_{01} & a_{02} \\ 1 & a_{10} & a_{11} & a_{12} \end{matrix}_{2\times 3} \quad A \qquad \times \qquad \begin{matrix} & 0 & 1 & 2 \\ 0 & b_{00} & b_{01} & b_{02} \\ 1 & b_{10} & b_{11} & b_{12} \\ 2 & b_{20} & b_{21} & b_{22} \end{matrix}_{3\times 3} \quad B \qquad = \qquad \begin{matrix} & 0 & 1 & 2 \\ 0 & c_{00} & c_{01} & c_{02} \\ 1 & c_{10} & c_{11} & c_{12} \end{matrix}_{2\times 3} \quad r_1 c_2$$

$$c_{01} = a_{00} * b_{01} + a_{01} * b_{11} + a_{02} * b_{21}$$

$$c_{ij} = \sum a_{ik} * b_{kj}$$

```java
if(c1 != r2) {
    System.out.println("Invalid input");
    return;
}

int[][] c = new int[r1][c2];

int commonDimension = c1; //or r2

for(int i=0;i<c.length;i++) {
    for(int j=0;j<c[0].length;j++) {
        for(int k=0;k<commonDimension;k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}

for(int i=0;i<c.length;i++) {
    for(int j=0;j<c[0].length;j++) {
        System.out.print(c[i][j] + " ");
    }
    System.out.println();
}
```

```
class Grid

    public static void multiply(int A[][], int B[][], int C[][], int N)
      {

          int commonDimension = A[0].length;

         //add code here.
          for(int i=0;i<C.length;i++) {
            for(int j=0;j<C[0].length;j++) {
                for(int k=0;k<commonDimension;k++) {
                    C[i][j] += A[i][k] * B[k][j];
                }
             }
   }

      }
```

$O(N^3)$

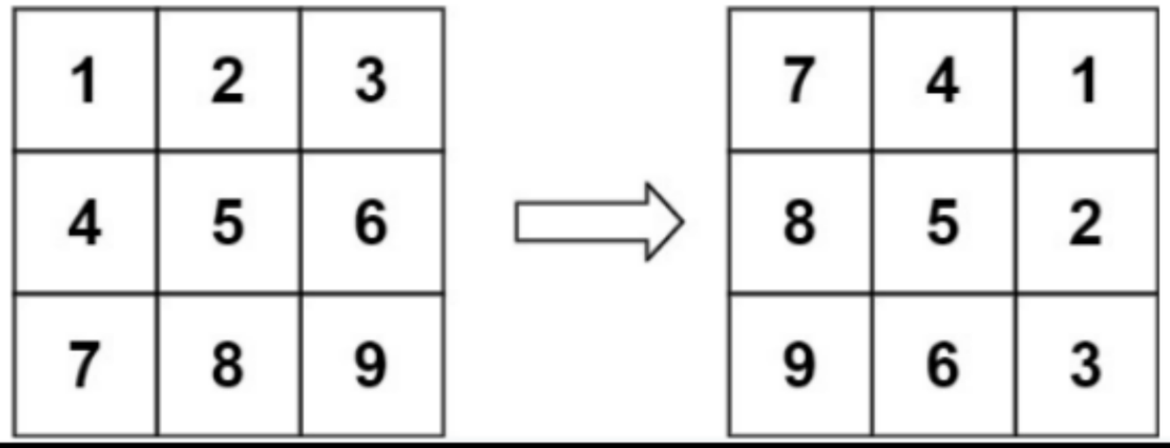
48. Rotate Image

Medium 12097 575 Add to List Share

You are given an `n x n` 2D `matrix` representing an image, rotate the image by **90** degrees (clockwise).

You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

Example 1:



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

rotate →

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

transpose →

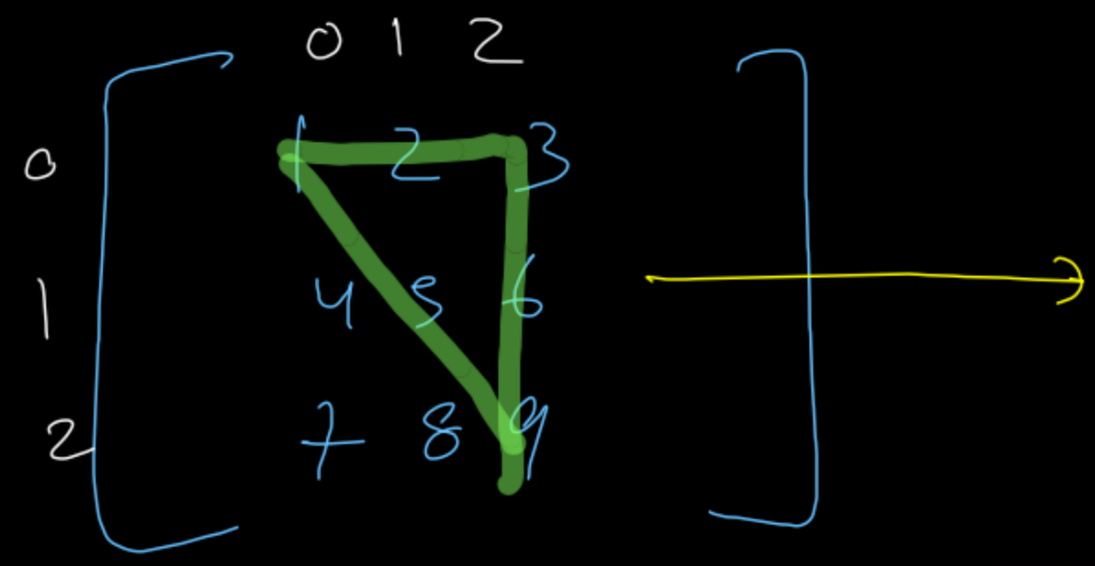
	0	1	2
0	1	4	7
1	2	5	8
2	3	6	9

① reverse each row ②

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

Transpose

↳ swapping of arr[i][j] & arr[j][i]



col = row ; col < n

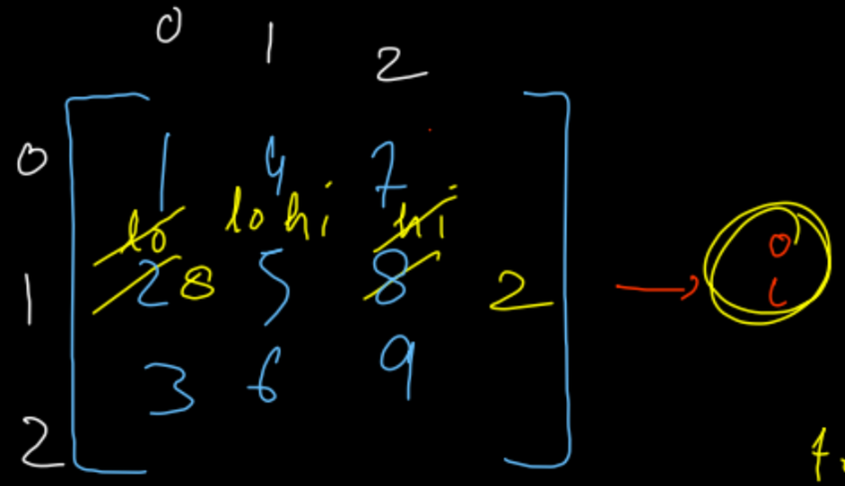
Traverse only in Upper Triangular

col >= row

Half

#Reverse

reverse(mat, i)




```java
public void transpose(int[][] mat) {
    //traverse only in upper triangle
    for(int i=0;i<mat.length;i++) {
        for(int j=i;j<mat[0].length;j++) {
            int temp = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = temp;
        }
    }
}
```

```java
public void rotate(int[][] matrix) {
    transpose(matrix);

    for(int i=0;i<matrix.length;i++) {
        reverseRow(matrix,i);
    }
}
```

```java
public void reverseRow(int[][] matrix,int i) {
    int lo = 0;
    int hi = matrix[0].length -1;

    while(lo < hi) {
        int temp = matrix[i][lo];
        matrix[i][lo] = matrix[i][hi];
        matrix[i][hi] = temp;
        lo++;
        hi--;
    }
}
```

Lecture - 14

Exit Point

Spiral Matrix

Saddle Point

Search in a 2-D Matrix -1

Search in a 2-D Matrix -2

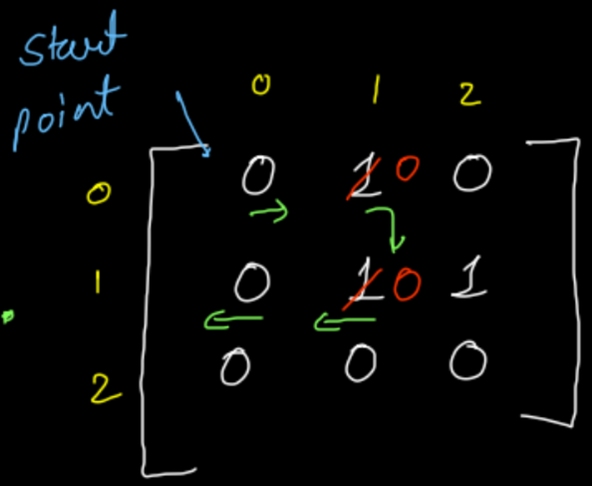
Ring Rotate

Exit Point of a Matrix

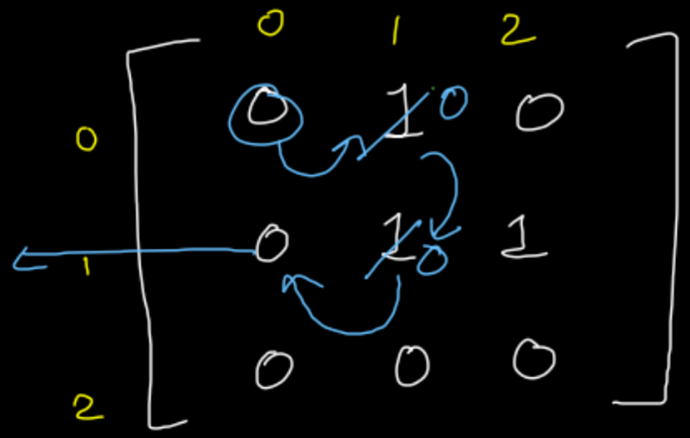
Exit Point in a Matrix

Easy Accuracy: **42.96%** Submissions: **1976** Points: **2**

Given a matrix of size n x m with 0's and 1's , you enter the matrix at cell (0,0) in left to right direction. whenever you encounter a 0 you retain in same direction , if you encounter a 1's you have to change direction to right of current direction and change that 1 value to 0, you have to find out from which index you will leave the matrix at the end.



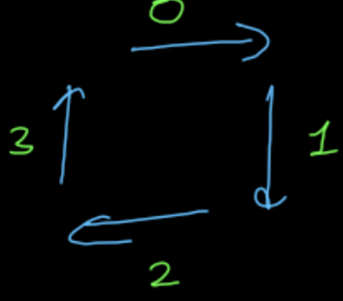
Starting direction: →



Initially dir = 0

i → row

j → col



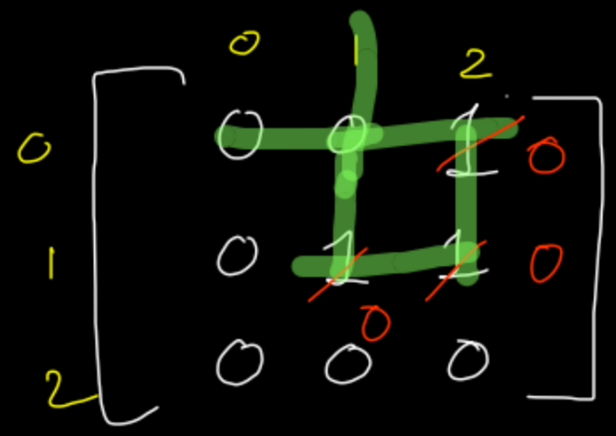
dir = 0 → j++;

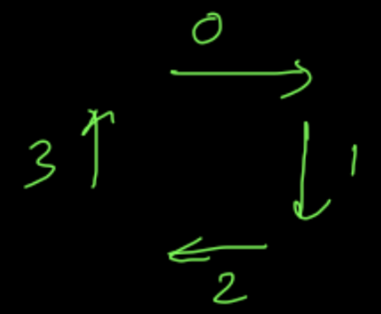
dir = 1 → i++;

dir = 2 → j--;

dir = 3 → i--;

dir = 0 & 2





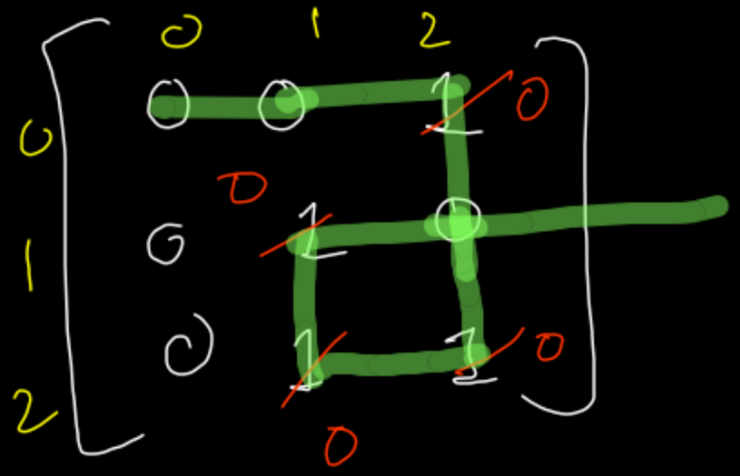
dir = (dir + 1) % 4 → 0, 1, 2, 3

dir = 0 → j++;

dir = 1 → i++;

dir = 2 → j--;

dir = 3 → i--;



dir = 0 → j++;

dir = 1 → i++;

dir = 2 → j--;

dir = 3 → i--;

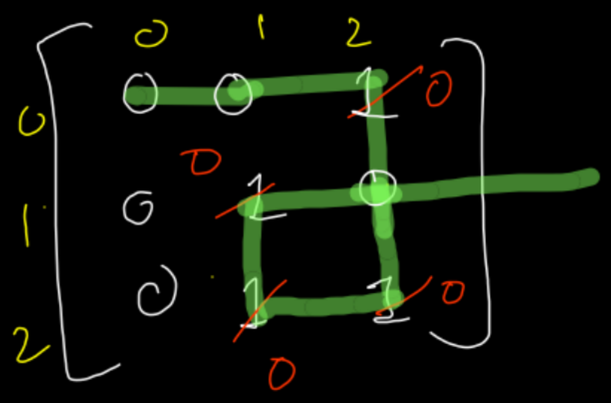
dir = ~~0~~ ~~1~~ ~~2~~ 3

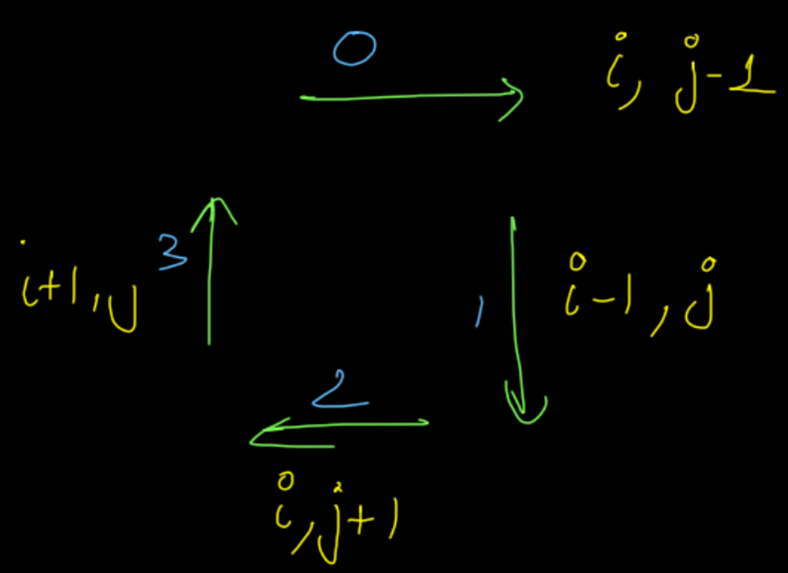
dir = (dir + 1) % 4

(1 + 1) % 4 = 2 % 4 = 2

dir = (2 + 1) % 4 = 3 % 4 = 3

dir = (3 + 1) % 4 = 4 % 4 = 0





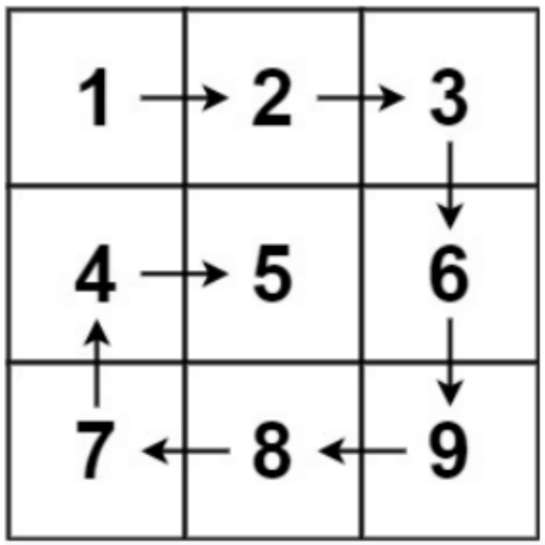
```java
public int[] FindExitPoint(int[][] matrix)
{
    // code here
    int[] res = new int[2];
    int i = 0;
    int j = 0;
    int dir = 0;

    while(i >= 0 && i < matrix.length && j >=0 && j < matrix[0].length) {
        if(matrix[i][j] == 1) {
            dir = (dir + 1) % 4;
            matrix[i][j] = 0;
        }

        if(dir == 0) {
            j++;
        } else if(dir == 1) {
            i++;
        } else if(dir == 2) {
            j--;
        } else {
            i--;
        }
    }

        //you have exited the matrix
        if(dir == 0) {
            j--;
        } else if(dir == 1) {
            i--;
        } else if(dir == 2) {
            j++;
        } else {
            i++;
        }

        res[0] = i;
        res[1] = j;

        return res;
    }
```

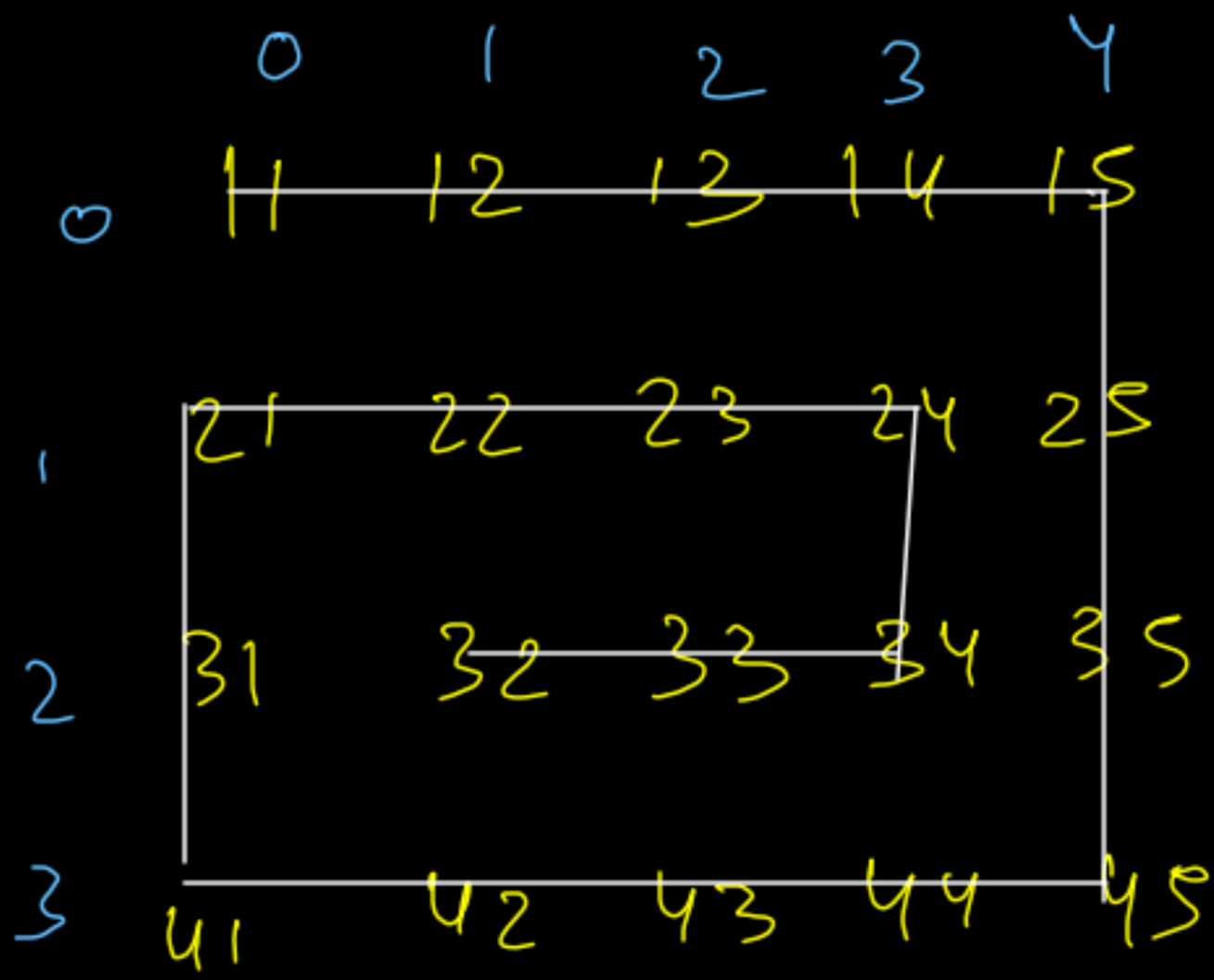
54. Spiral Matrix

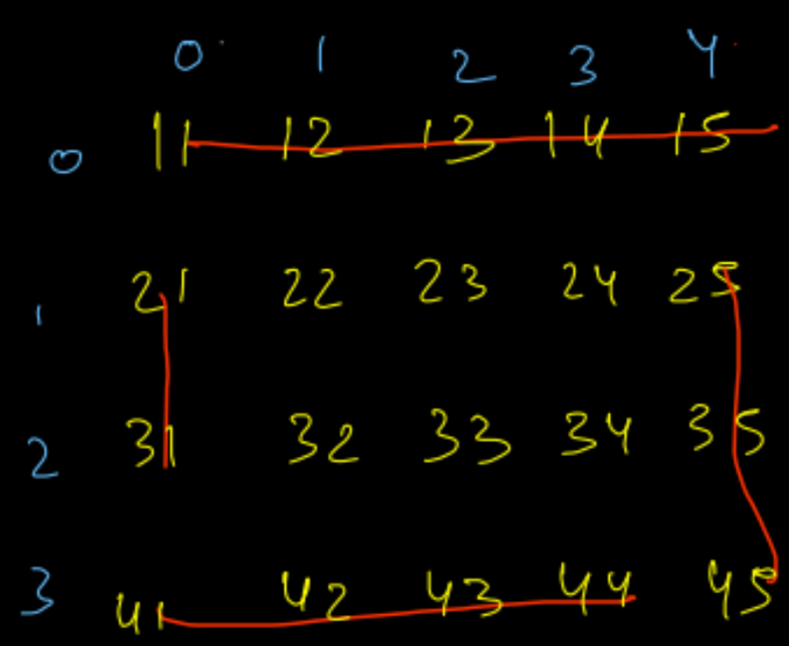
Medium 8885 914 Add to List Share

Given an `m x n` `matrix`, return *all elements of the* `matrix` *in spiral order.*

Example 1:




0 1 2 3 4
0
11 12 13 14 15
1
21 22 23 24 25
2
31 32 33 34 35
3
41 42 43 44 45



fr = 0 lr = mat.length - 1

fc = 0 lc = mat[0].length - 1

1 # first row

j = fc; j <= lc; j++

fr++;

2 # Last col

i = fr; i <= lr; i++

lc--;

3 # Last row

j = lc; j >= fc; j--

lr--;

4 # first col

i = lr; i >= fr; i--

fc++

```java
    int rows = matrix.length;
    int cols = matrix[0].length;
    int fr = 0, fc = 0, lr = rows-1, lc = cols -1;
    int count = 0;

    List<Integer> list = new ArrayList<>();
```

```java
    while(count != rows * cols) {
        //first row
        for(int j=fc;j<=lc;j++) {
            list.add(matrix[fr][j]);
            count++;
        }
        fr++;

        if(count == rows*cols) break;

        //last col
        for(int i=fr;i<=lr;i++) {
            list.add(matrix[i][lc]);
            count++;
        }
        lc--;

        if(count == rows*cols) break;

        //last row
        for(int j=lc;j>=fc;j--) {
            list.add(matrix[lr][j]);
            count++;
        }
        lr--;

        if(count == rows*cols) break;
```

```java
        if(count == rows*cols) break;

        //first col
        for(int i=lr;i>=fr;i--) {
            list.add(matrix[i][fc]);
            count++;
        }
        fc++;

        if(count == rows*cols) break;
    }

    return list;
```

240. Search a 2D Matrix II

Medium 9221 153 Add to List Share

Write an efficient algorithm that searches for a value `target` in an `m x n` integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

Search in a Row-wise & Column Wise Sorted Matrix

Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

Approach - 1
{Brute force}

Search in entire matrix

O(M*N)

Approach - 2

Apply Binary Search on each row/col.

O(M * log(N)) or
O(N log(M))

```java
public boolean binSearch(int[][] matrix,int i,int target) {
    int lo = 0;
    int hi = matrix[0].length -1;

    while(lo <= hi) {
        int mid = lo + (hi-lo)/2;

        if(matrix[i][mid] == target) return true;
        else if(matrix[i][mid] < target) lo = mid + 1;
        else hi = mid -1;
    }

    return false;
}
```

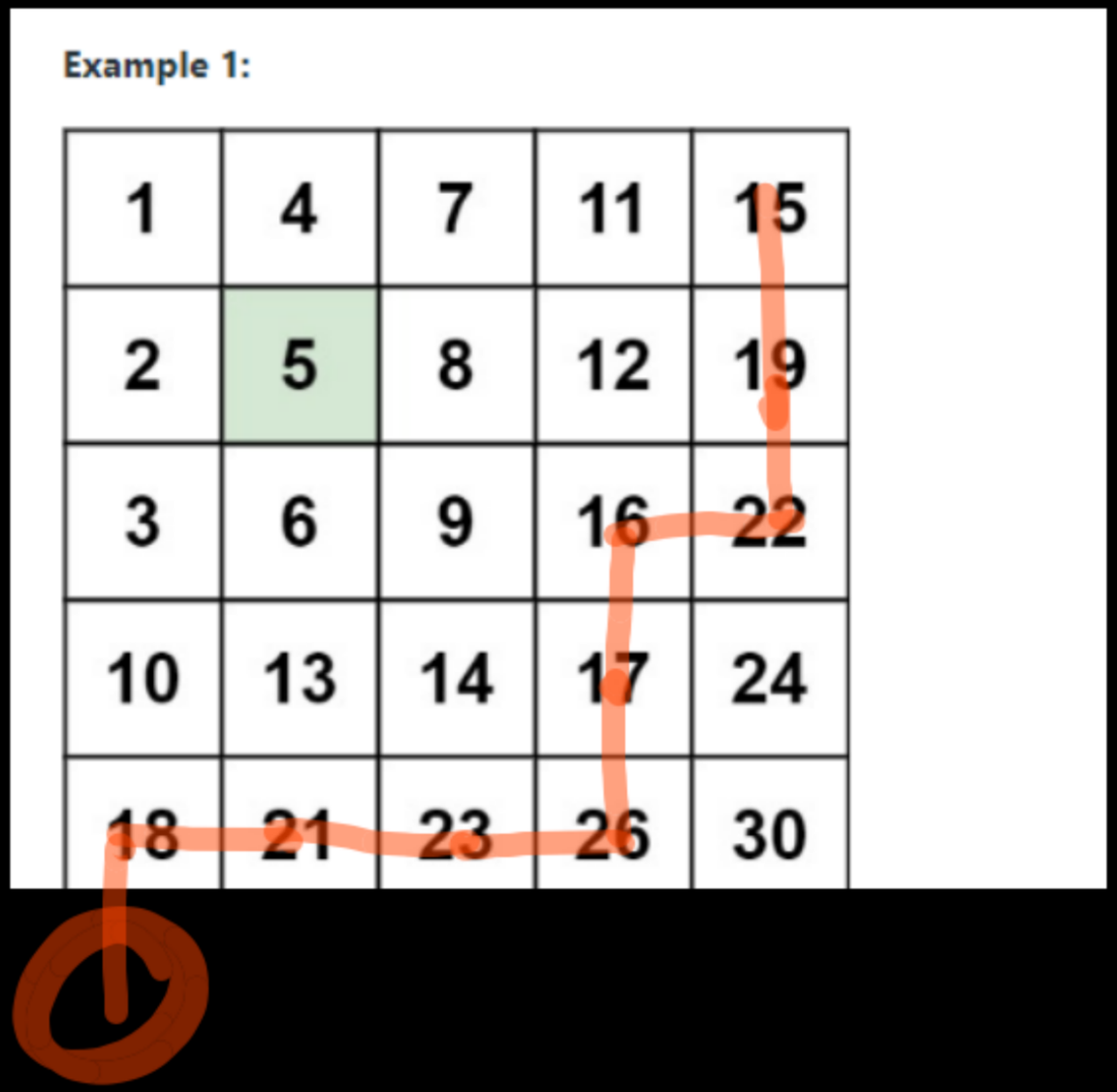
```java
    public boolean searchMatrix(int[][] matrix, int target) {

        for(int i=0;i<matrix.length;i++) {
            if(target >= matrix[i][0] && target <= matrix[i][matrix[0].length-1]) {
                boolean val = binSearch(matrix,i,target);
                if(val == true) return true;
            }
        }

        return false;
    }
```

#Approach-3 {Staircase Search}

O(N+M)

target = 20

while (i < matrix.length && j >= 0)



Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int i = 0;
        int j = matrix[0].length -1;

        while(i < matrix.length && j >= 0) {
            if(matrix[i][j] == target) return true;
            if(matrix[i][j] < target) i++;
            else j--;
        }

        return false;
    }
}
```

74. Search a 2D Matrix

Medium 9648 304 Add to List Share

Write an efficient algorithm that searches for a value `target` in an `m x n` integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Search in a Sorted Matrix

Example 1:

	0	1	2	3
0	1	3	5	7
1	10	11	16	20
2	23	30	34	60

target = 30

{1, 3, 5, 7, 10, 11, 16, 20, 23, 30, 34, 60}
0 1 2 3 4 5 6 7 8 9 10 11

lo lo mid hi mid hi

lo = 6

hi = 11

mid = (6+11)/2 = 8

rIdx = 1

cIdx = 1

rowIdx = idx / cols;

colIdx = idx % cols;

if(mat[rIdx][cIdx] == target)

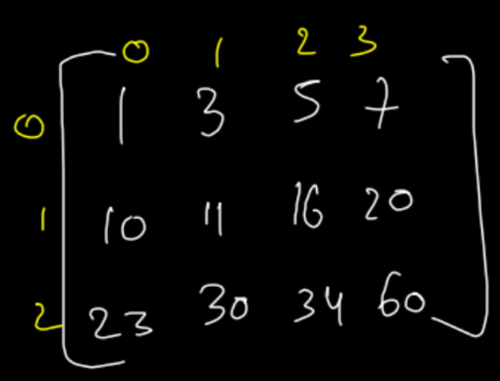
lo = mid + 1

1-D Array Idx = idx

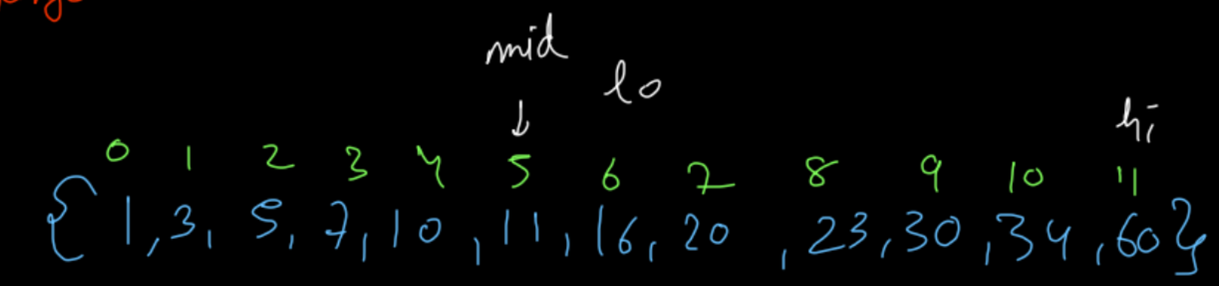
Row Idx = idx / cols;

Col Idx = idx % cols;

```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int lo = 0;
        int hi = matrix.length * matrix[0].length -1;

        while(lo <= hi) {
            int mid = lo + (hi-lo)/2;

            int rIdx = mid / matrix[0].length;
            int cIdx = mid % matrix[0].length;

            if(matrix[rIdx][cIdx] == target) return true;
            else if(matrix[rIdx][cIdx] < target) lo = mid + 1;
            else hi = mid -1;
        }

        return false;
    }
}
```



target = 30



arr[mid] == target

rowIdx = idx/cols;

colIdx = idx%cols;

5/4 → row

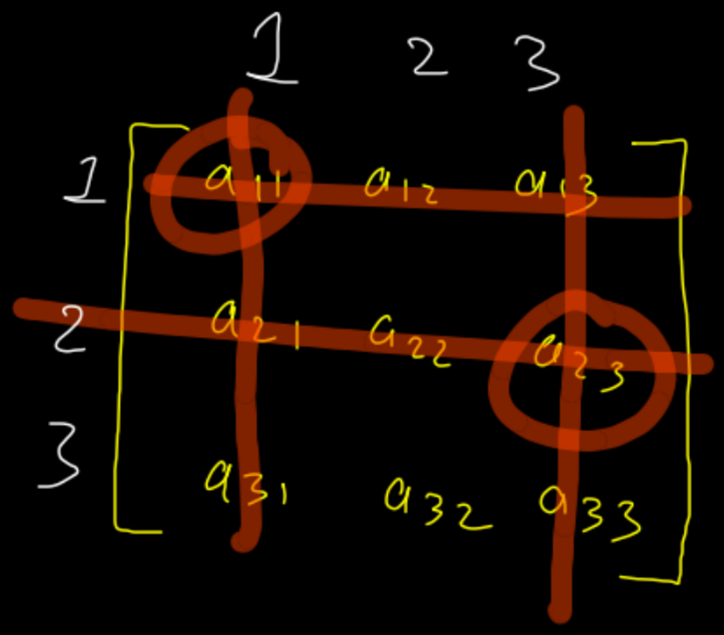
5%4 → 0 → col

Saddle Price

Easy

Prev Next

1. You are given a square matrix of size 'n'. You are given n*n elements of the square matrix.
2. You are required to find the saddle price of the given matrix and print the saddle price.
3. The saddle price is defined as the least price in the row but the maximum price in the column of the matrix.



a_{11} → min value in row 1

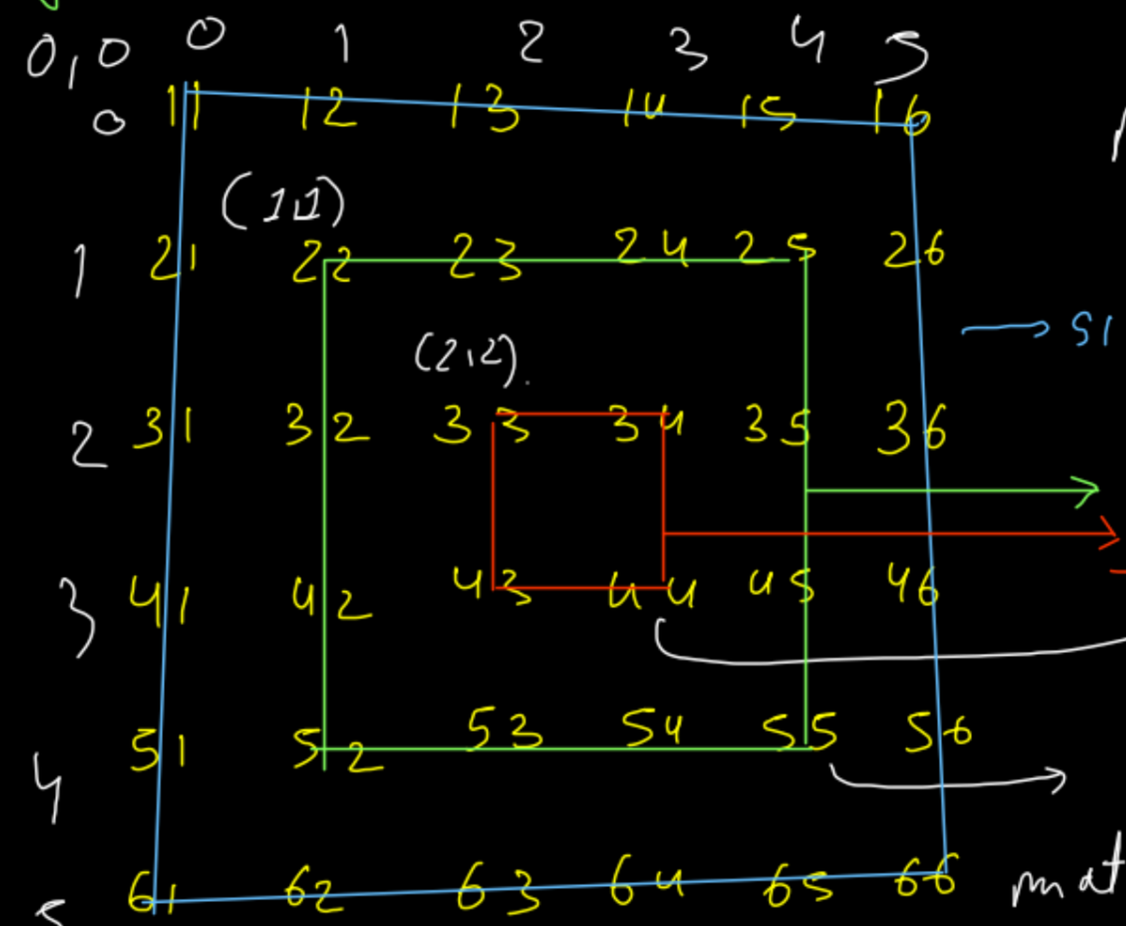
↳ max value in col 1

a_{23} → min value in row 2

↳ max value in col 3

H.W

Ring Rotate



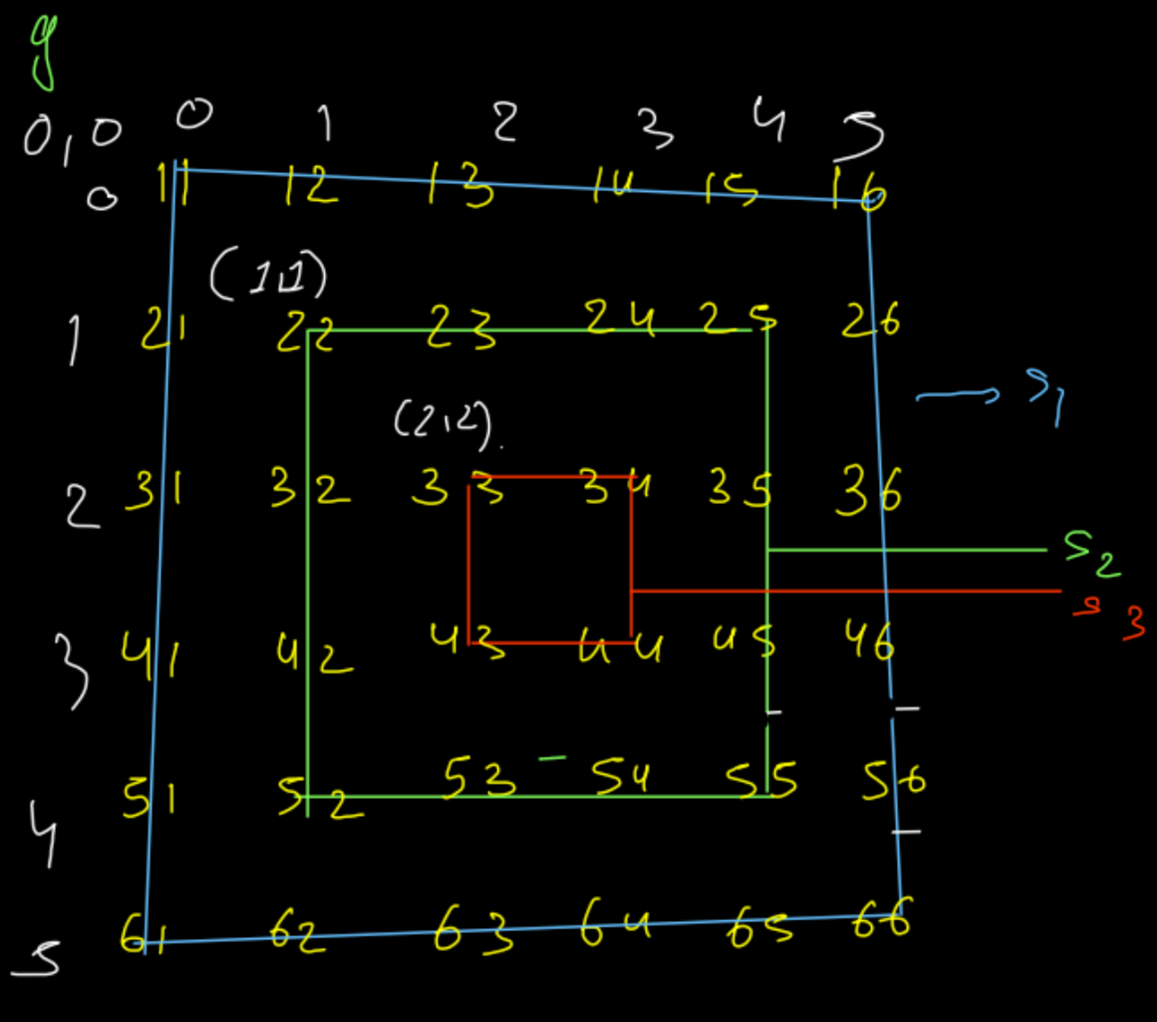
Top Left Corner = S-1, S-1

Bottom Right Corner = rows - S, cols - S

mat.length-3, mat[0].length-3

mat.length-2, mat[0].length-2

mat.length-1, mat[0].length-1



$$(lw) + (bw) + (rw) + (fw) - 4$$

$$2 * lw + 2 * bw - 4$$

$$2(l_r - f_r + 1) + 2(l_c - f_c + 1) - 4$$

$$2(l_r - f_r + 1) + 2(l_c - f_c + 1) - 4$$

$$\Rightarrow \quad 2l_r - 2f_r + 2 + 2l_c - 2f_c + 2 - 4$$

$$= 2(l_r - f_r + l_c - f_c)$$

```java
public static void main(String[] args) throws Exception {
    // write your code here
    Scanner scn = new Scanner(System.in);
    int rows = scn.nextInt();
    int cols = scn.nextInt();

    int[][] arr = new int[rows][cols];

    for(int i=0;i<rows;i++) {
        for(int j=0;j<cols;j++) {
            arr[i][j] = scn.nextInt();
        }
    }

    int s = scn.nextInt();
    int r = scn.nextInt();

    shellRotate(arr,s,r);
    display(arr);
}
```

```java
public static void shellRotate(int[][] arr,int s, int r) {
    int[] oneD = fillOneDFromShell(arr,s);
    rotate(oneD,r);
    fillShellFromOneD(arr,s,oneD);
}

public static void rotate(int[] arr, int k) {
    int n = arr.length;

    k = k % n;
    if(k < 0) k = k + n;

    reverse(arr,n-k,n-1);
    reverse(arr,0,n-k-1);
    reverse(arr,0,n-1);
}

public static void reverse(int[] arr,int lo, int hi) {
    while(lo < hi) {
        int temp = arr[lo];
        arr[lo] = arr[hi];
        arr[hi] = temp;
        lo++;
        hi--;
    }
}

public static void display(int[][] arr){
    for(int i = 0; i < arr.length; i++){
        for(int j = 0; j < arr[0].length; j++){
            System.out.print(arr[i][j] + " ");
        }
        System.out.println();
    }
}
}
```

```java
public static int[] fillOneDFromShell(int[][] arr, int s) {
    int fr = s-1;
    int fc = s-1;
    int lr = arr.length -s;
    int lc = arr[0].length -s;

    int size = 2 * (lr - fr + lc - fc);
    int[] oneD = new int[size];
    int idx = 0;

    //first col
    for(int i=fr;i<=lr;i++) {
        oneD[idx] = arr[i][fc];
        idx++;
    }
    fc++;

    //last row
    for(int j=fc;j<=lc;j++) {
        oneD[idx] = arr[lr][j];
        idx++;
    }
    lr--;

    //last col
    for(int i=lr;i>=fr;i--) {
        oneD[idx] = arr[i][lc];
        idx++;
    }
    lc--;

    //first row
    for(int j=lc;j>=fc;j--) {
        oneD[idx] = arr[fr][j];
        idx++;
    }

    fr++;

    return oneD;
}
```

```java
public static void fillShellFromOneD(int[][] arr, int s, int[] oneD) {
    int fr = s-1;
    int fc = s-1;
    int lr = arr.length -s;
    int lc = arr[0].length -s;

    int idx = 0;

    //first col
    for(int i=fr;i<=lr;i++) {
        arr[i][fc] = oneD[idx];
        idx++;
    }
    fc++;

    //last row
    for(int j=fc;j<=lc;j++) {
        arr[lr][j] = oneD[idx];
        idx++;
    }
    lr--;

    //last col
    for(int i=lr;i>=fr;i--) {
        arr[i][lc] = oneD[idx];
        idx++;
    }
    lc--;

    //first row
    for(int j=lc;j>=fc;j--) {
        arr[fr][j] = oneD[idx];
        idx++;
    }

    fr++;
}
```

Lecture - 15

Strings in Java

Interning & its need

Implications of String Interning

- immutability
- == vs equals

String Builders & String Buffers

Strings in Java

String : A string is a Class in Java. So, just like any other class, it will have some data members & member functions. However, in simple terms, we can say that a String is an array of characters.

- ↳ lowercase ('a' - 'z')
- ↳ uppercase ('A' - 'Z')
- ↳ digits ('0' - '9')
- ↳ special symbols ('#', '*', etc)
- ↳ null ('\0') character

Program to Create and Use Strings in Java.

```java
public static void main(String[] args) {
    String str = "DSA"; (1)
    String str1 = new String("Dev"); (2)
    System.out.println(str);
    System.out.println(str1);
}
```

```java
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = scn.next(); // after the space, no input will be taken
        System.out.println(str);
    }
}
```

stdin

```
Hello, my name is Khan
```

```
Finished in 131 ms
Hello,
```

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();
    System.out.println(str);
}
```

```
Finished in 130 ms
Hello, my name is Khan
```

stdin

```
Hello, my name is Khan
```

length() and charAt() Methods

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();

    for(int i=0;i<str.length();i++) {
        char ch = str.charAt(i);
        System.out.println(ch);
    }
}
```

"Hello" → { 'H', 'e', 'l', 'l', 'o' }
0 1 2 3 4

Each and every character can be represented by a numeric code.
This is called its ASCII value.

American Standard Code for Information Interchange.

** Java uses Unicode (code for many languages). Thus, the size of char datatype is 2bytes in Java.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

https://unicode-table.com/en/ → for unicode Table

'0' - '9'
48 57

'A' - 'Z'
65 90

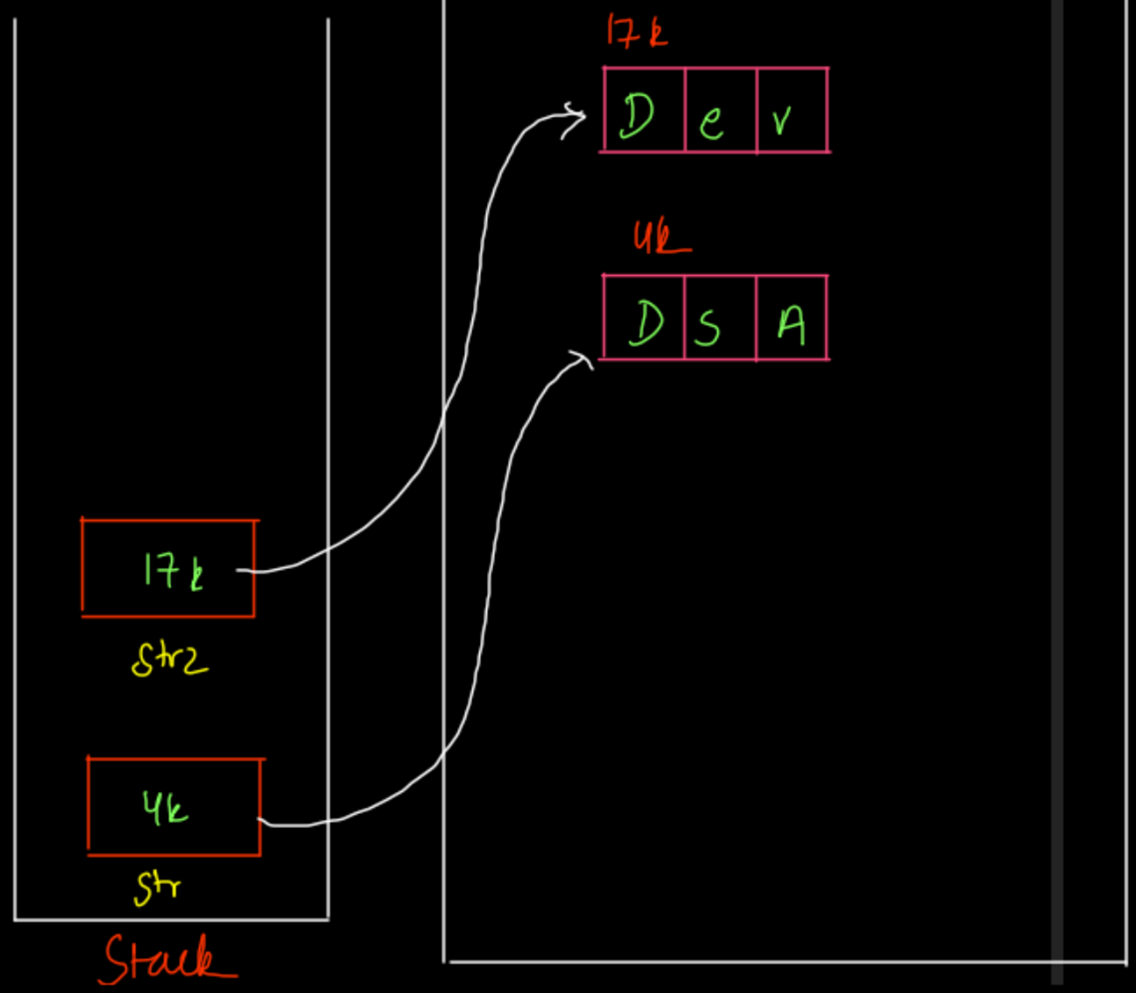
'a' - 'z'
97 122

+1 'a' - 'A' = 32

'A' - 'a' = -32

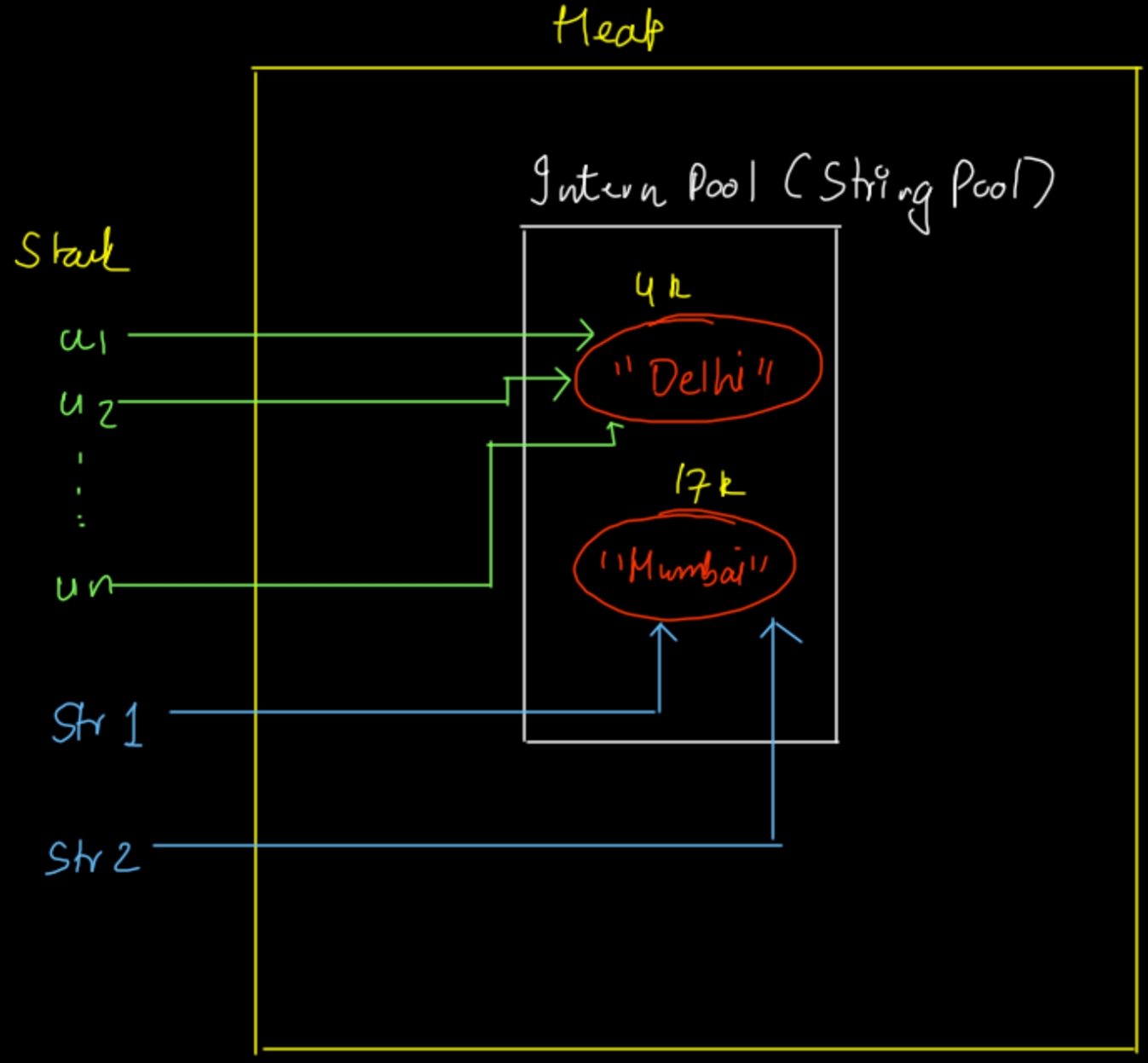
Strings Memory Mapping (Only Overview)

```
String str = "DSA";

String str2 = new String("Dev");
```



Strings Interning

```
String u1 = "Delhi";
String u2 = "Delhi";
.
.
.
String un = "Delhi";

String str1 = "Mumbai";
String str2 = "Mumbai";
```



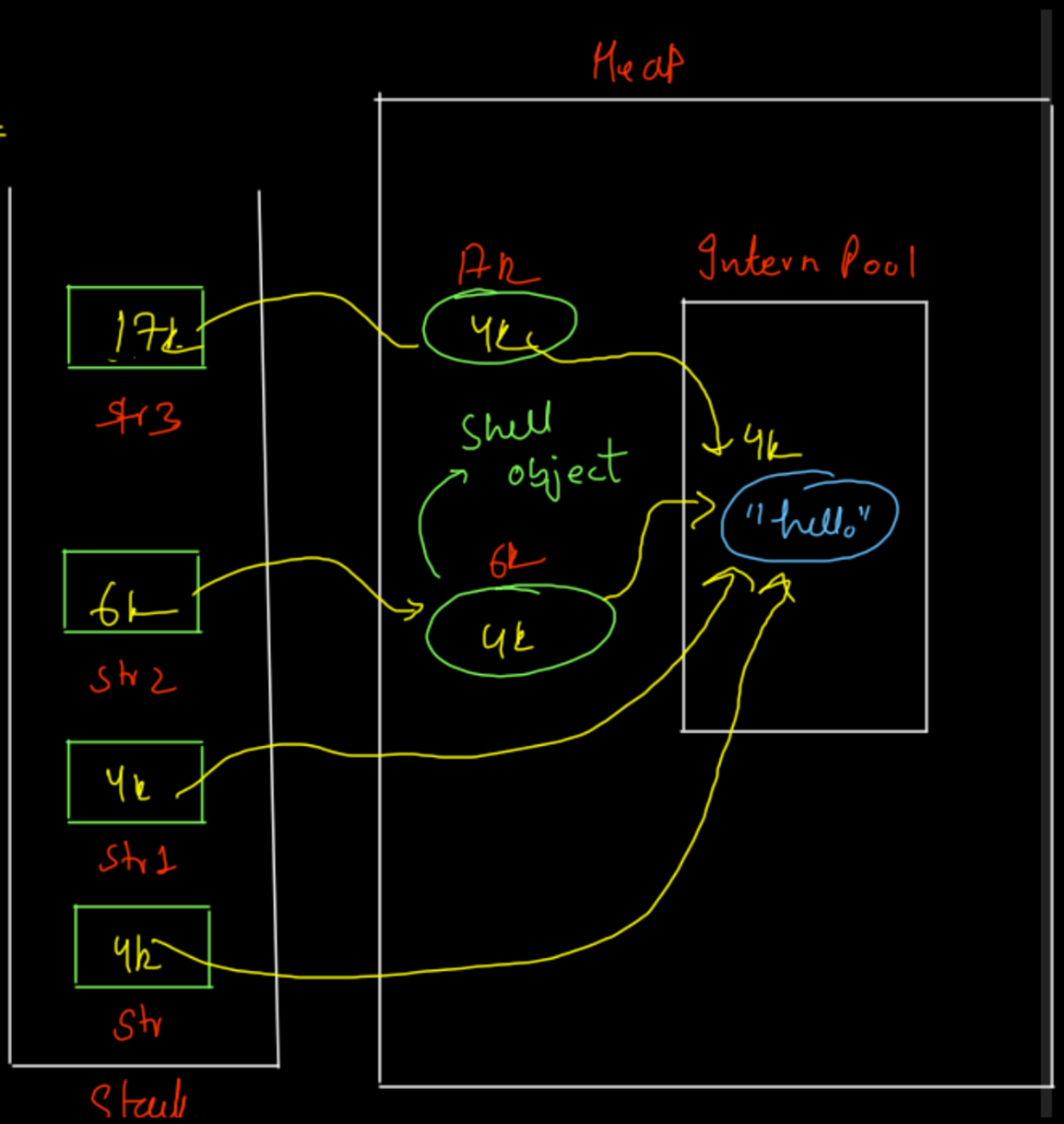
In Java, string literals are created inside a special memory section inside the heap memory called the intern pool or the string pool.

Interning is the concept of using the same string literal i.e. the same string literal is referenced by many references to save memory like u1, u2,... un in the previous example were all referencing to the same string literal "Delhi".

Interning in Java has 2 main implications
1. Java Strings are immutable
2. == vs equals in Strings

String Complete Memory Mapping

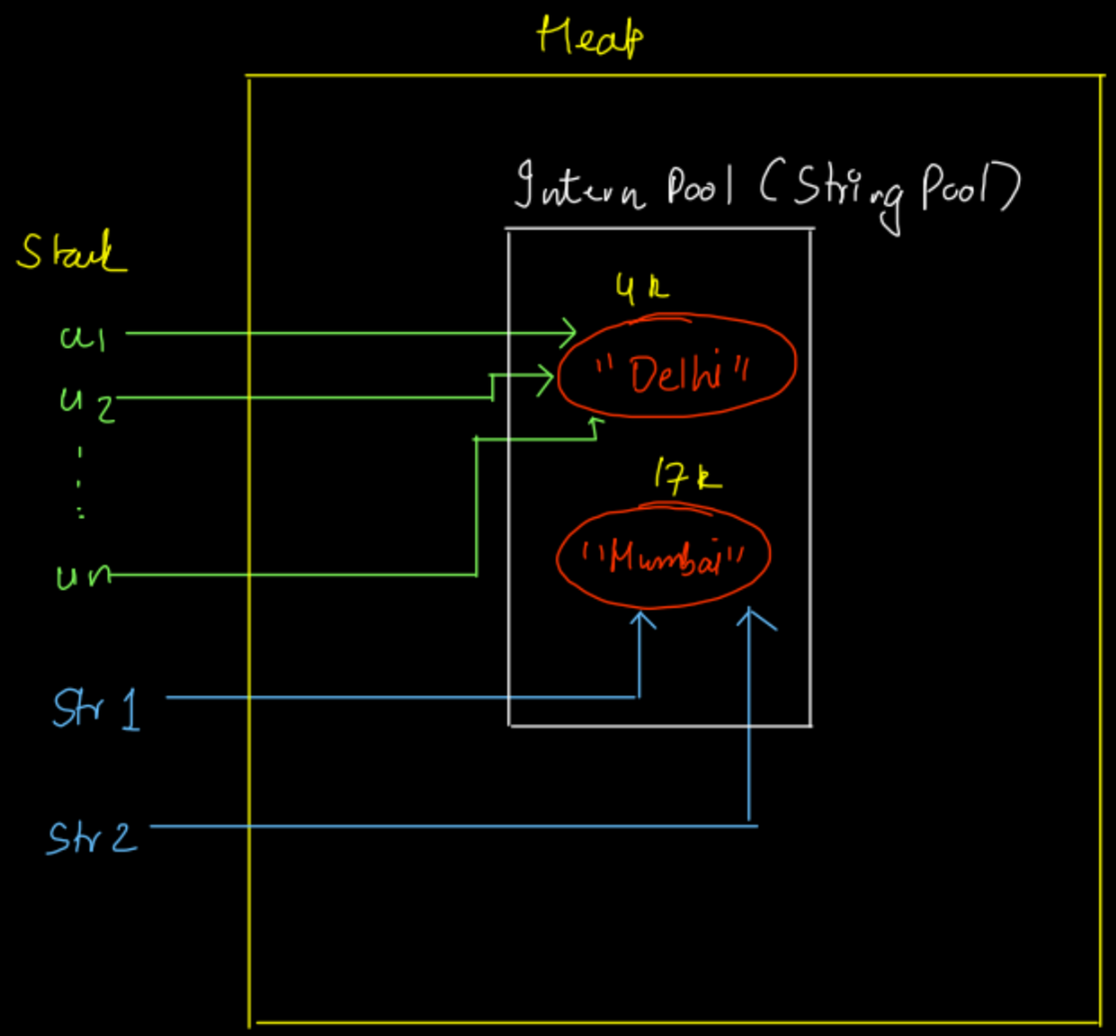
```
String str = "hello";
String str1 = "hello";
String str2 = new String("hello");
String str3 = new String("hello");
```



When we directly assign the String literal to a reference, if the literal already exists in the intern pool, the reference points to the already existing literal due to interning. Otherwise, a new literal is created in the intern pool and the reference stores it's address.

Instead of directly assigning a String literal, if the new keyword is used, a shell object is created which further stores the address of the String literal and our reference stores the address of this shell object. This shell object is created inside heap but not inside the intern pool. Whenever the new keyword is used, a new shell object will be created, however, the same string literal will be used if it exists due to interning.

Implications of Interning – Strings are immutable.

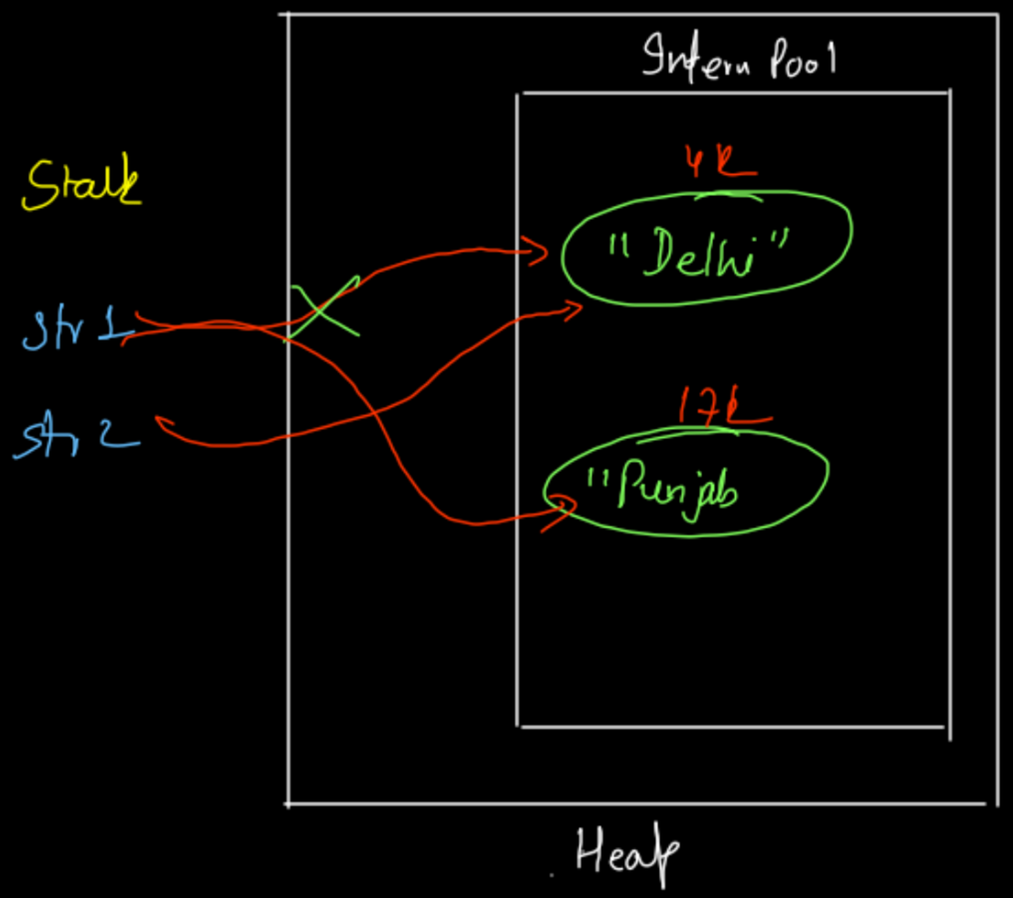


Need of immutability

```
a1 = "Punjab";
```

** References are mutable, instances are not.

```java
public class Main {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str1 = "Delhi";
        String str2 = "Delhi";

        System.out.println(str1);
        System.out.println(str2);

        str1 = "Punjab";

        System.out.println(str1);
        System.out.println(str2);

    }
}
```



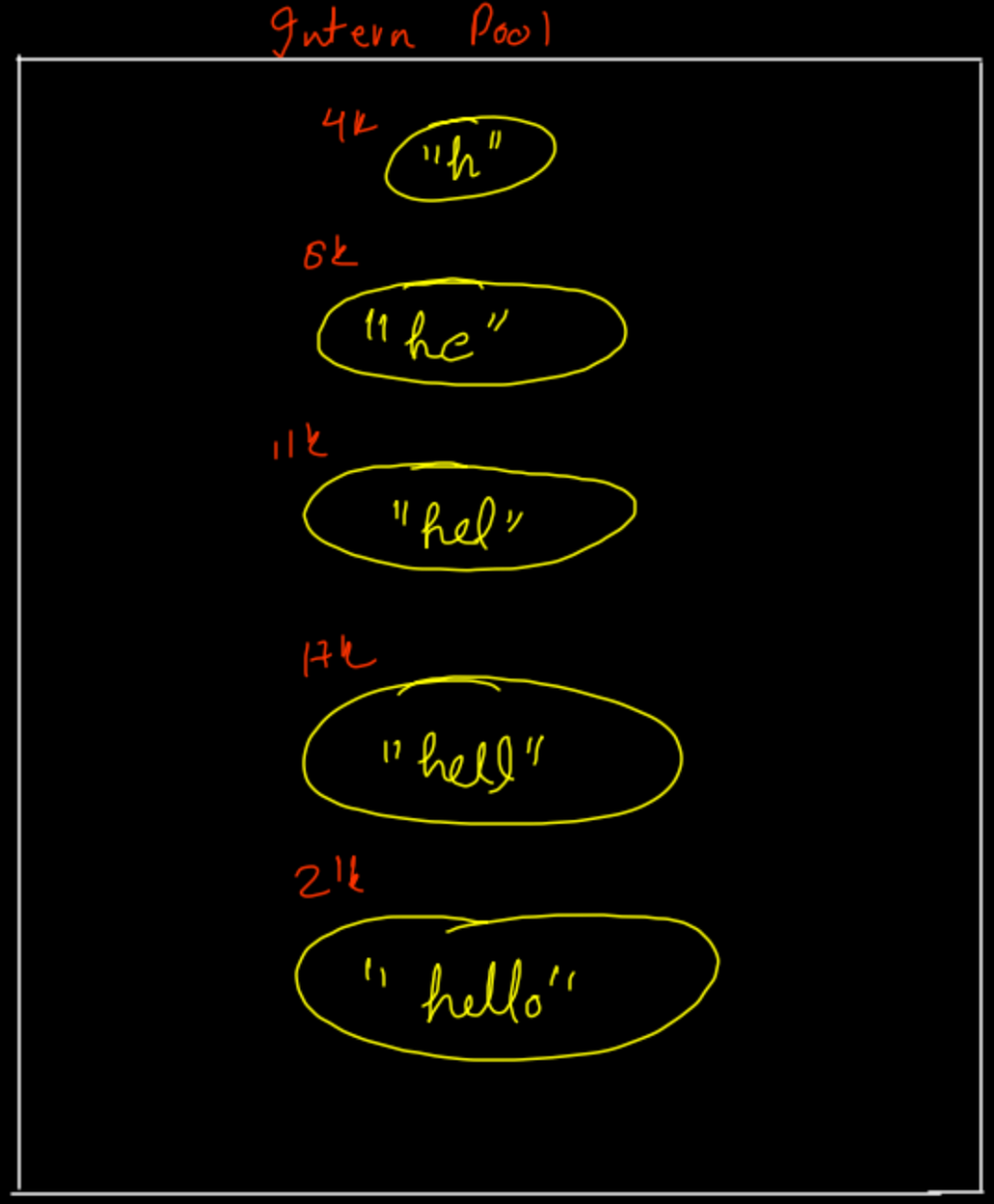

```java
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = "h";
        str = str + 'e';
        str  = str + 'l';
        str  = str + 'l';
        str = str + "o";
        System.out.println(str);
    }
}
```

```
str = "h";
str = str + "e";
str = str + "l";
str = str + "l";
str = str + "o";
```

O(N)

O(N²)

str = ~~4k~~ ~~8k~~ ~~11k~~ ~~17k~~ 21k



Hello → (5)

① (H)

② e → 1(H)

③ l → 2(He)

④ l → 3(Hel)

⑤ o → 4(Hell)

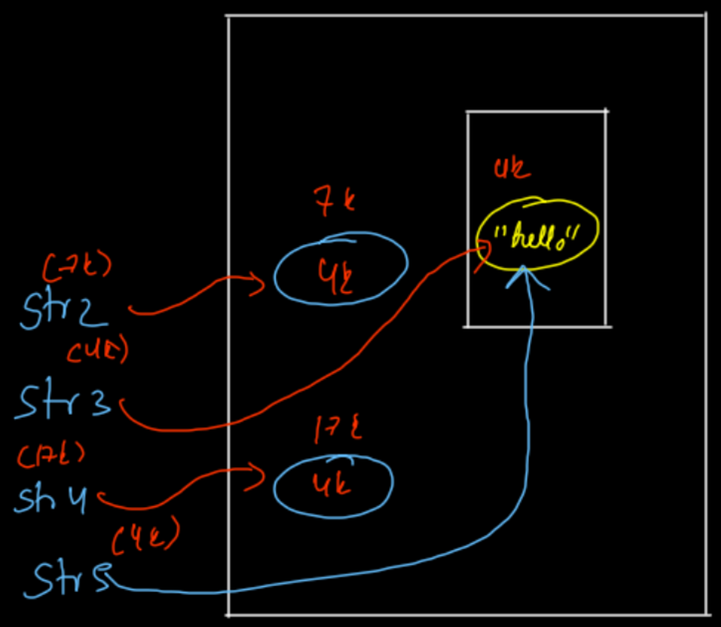
$\frac{N(N-1)}{2} = O(N^2)$

System.gc

Implications of Interning — equals vs ==

LHS == RHS

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // String str1 = "Guneet";
        String str2 = new String("hello");
        String str3 = "hello";
        String str4 = new String("hello");
        String str5 = "hello";

        System.out.println(str2 == str3); → false
        System.out.println(str3 == str5); → true
        System.out.println(str2 == str4); → false
        System.out.println(str2.equals(str3)); → true
        System.out.println(str2.equals(str4)); → true
        System.out.println(str2.equals(str5)); → true
    }
}
```



StringBuilders & String Buffers

You can say that they are just like Strings in C++. Here, references and instances, both are mutable.

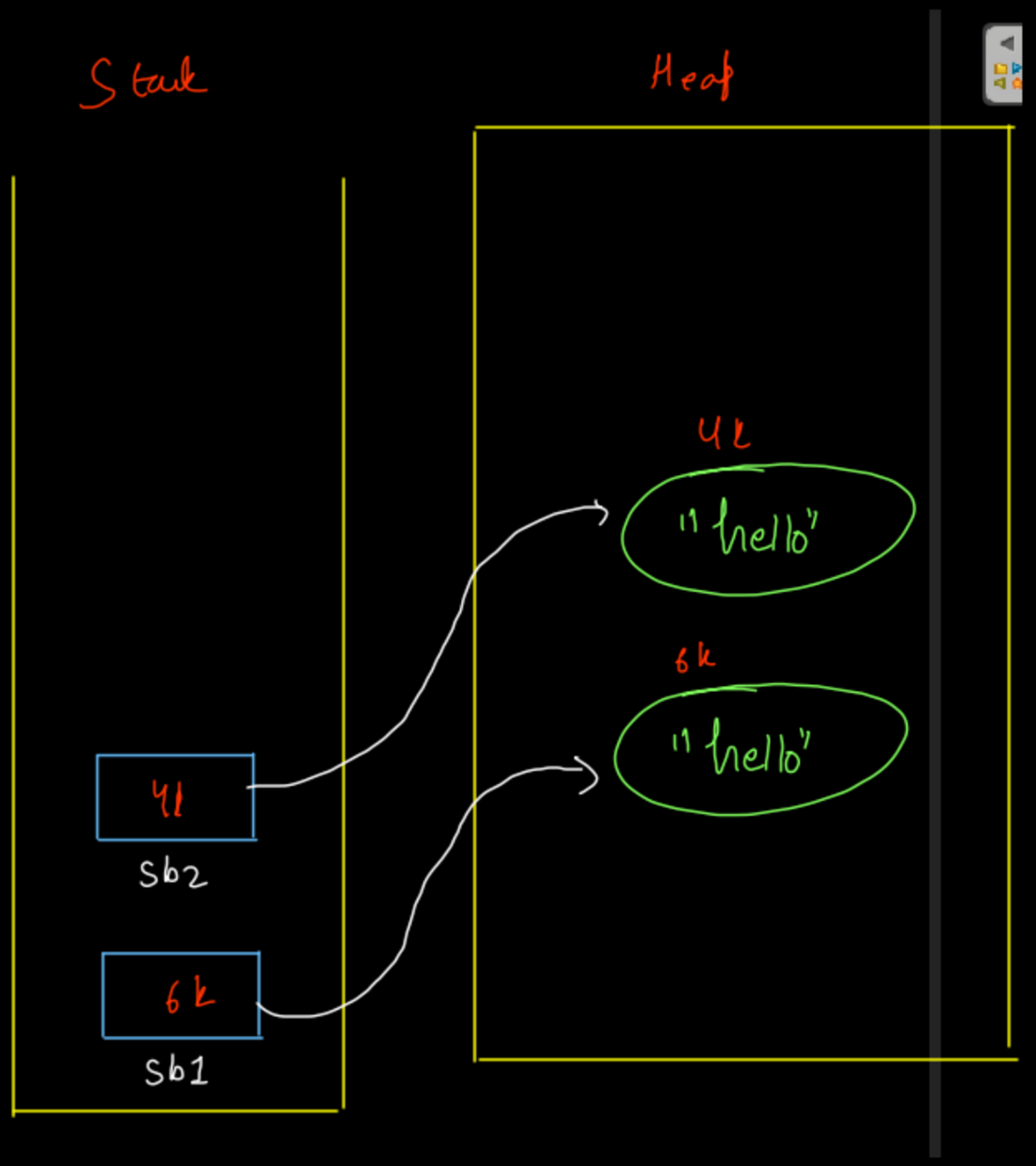
It is also possible to update (change) a character at a particular index.

There is no interning in StringBuilders and StringBuffers.

** Difference b/w a StringBuilder and StringBuffer is that the StringBuffer is a thread-safe class whereas StringBuilder is not.

StringBuilder (Memory Mapping)

```
StringBuilder sb1 = new StringBuilder("hello");
StringBuilder sb2 = new StringBuilder("hello");
```



StringBuilders (Usage & functions)

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    StringBuilder sb = new StringBuilder("Hello");
    System.out.println(sb);

    sb.setCharAt(0,'h');
    System.out.println(sb);
    sb.append('o');
    System.out.println(sb);

    sb.append(",there!");
    System.out.println(sb);
    sb.append(1);
    System.out.println(sb);

    String s = sb.toString();
    System.out.println(s);
}
```