Time Table { FJP - 2 }

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
9 PM class	9PM class	9PM class	off	9PM class	Weekly Test 9PM	Test + Doubts Discussion

Continuation of class

① Getting
② Patterns

Level ①

- Functions & Arrays
- 2D Arrays
- String & StringBuilder

DSA

- Recursion & Backtracking
- Searching & Sorting
- Dynamic Programming

My Introduction

→ Pointers

→ Pepcoding Articles (30 - 40%)

→ Job-Switch program - 4 { 3-4 months }

→ leetcode

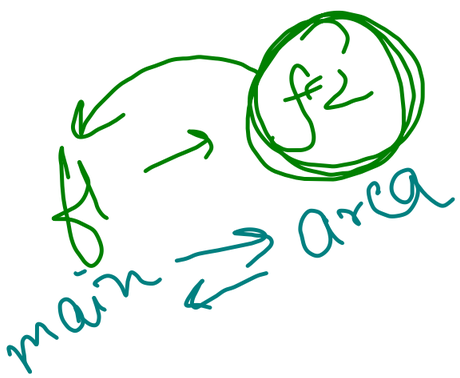
→ linkedIn

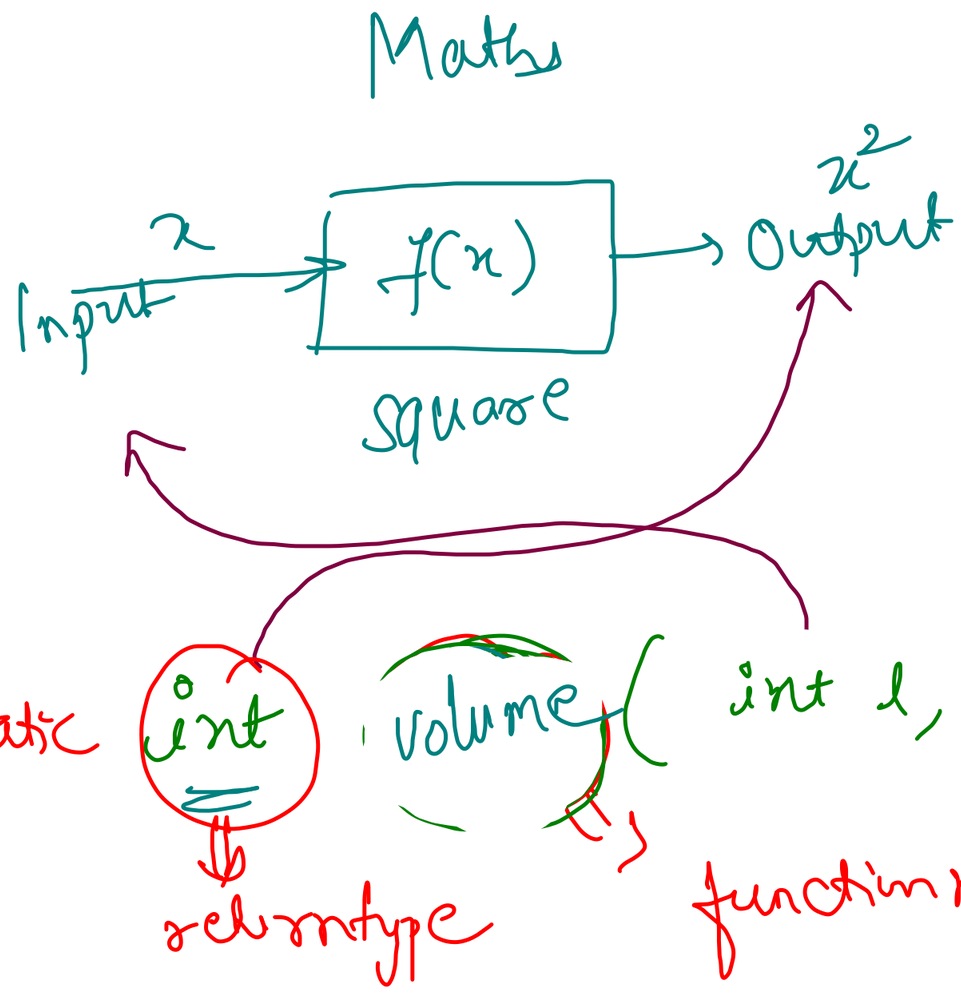
→ Competitive Programming

→ Product Based company

- Google (3)
- Salesforce (AMTS) (2)
- Sprinklr (1)

Functions

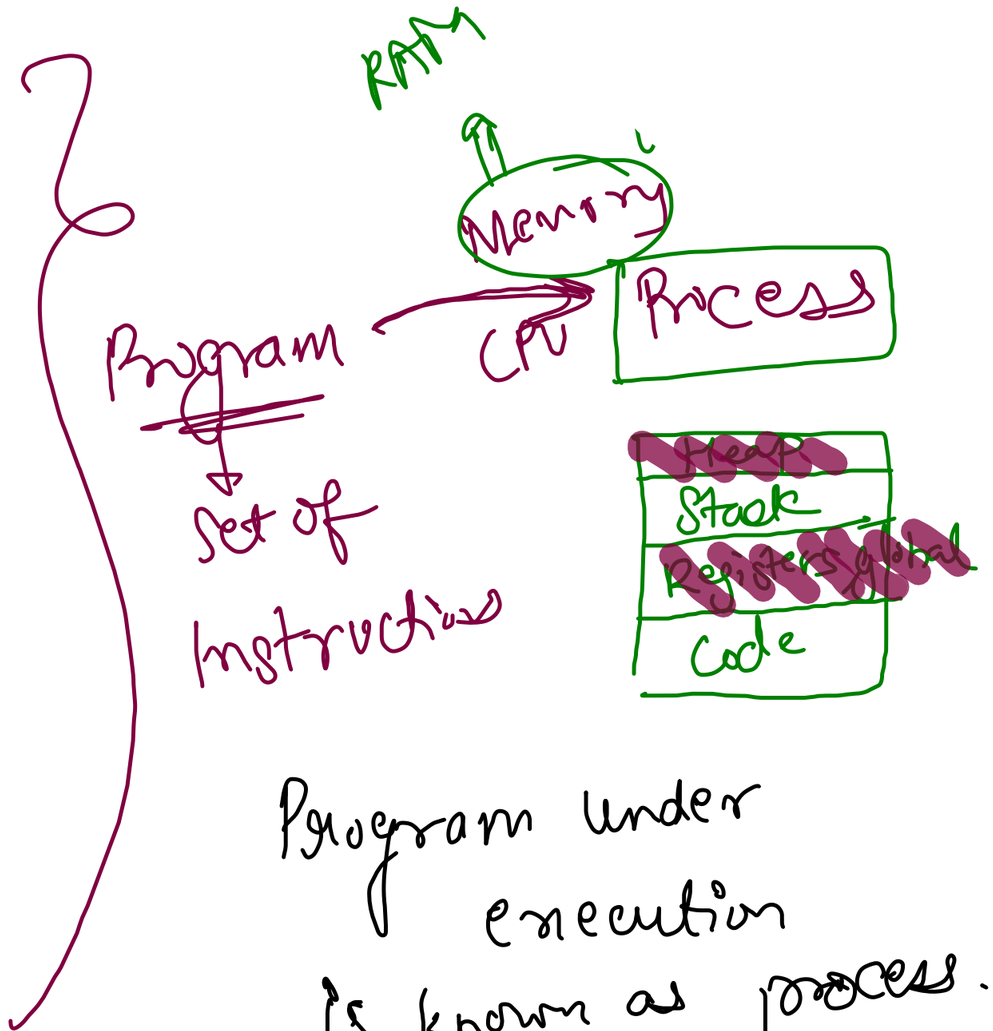





```
public static int volume( int l, int b, int h) {

}
```

returntype

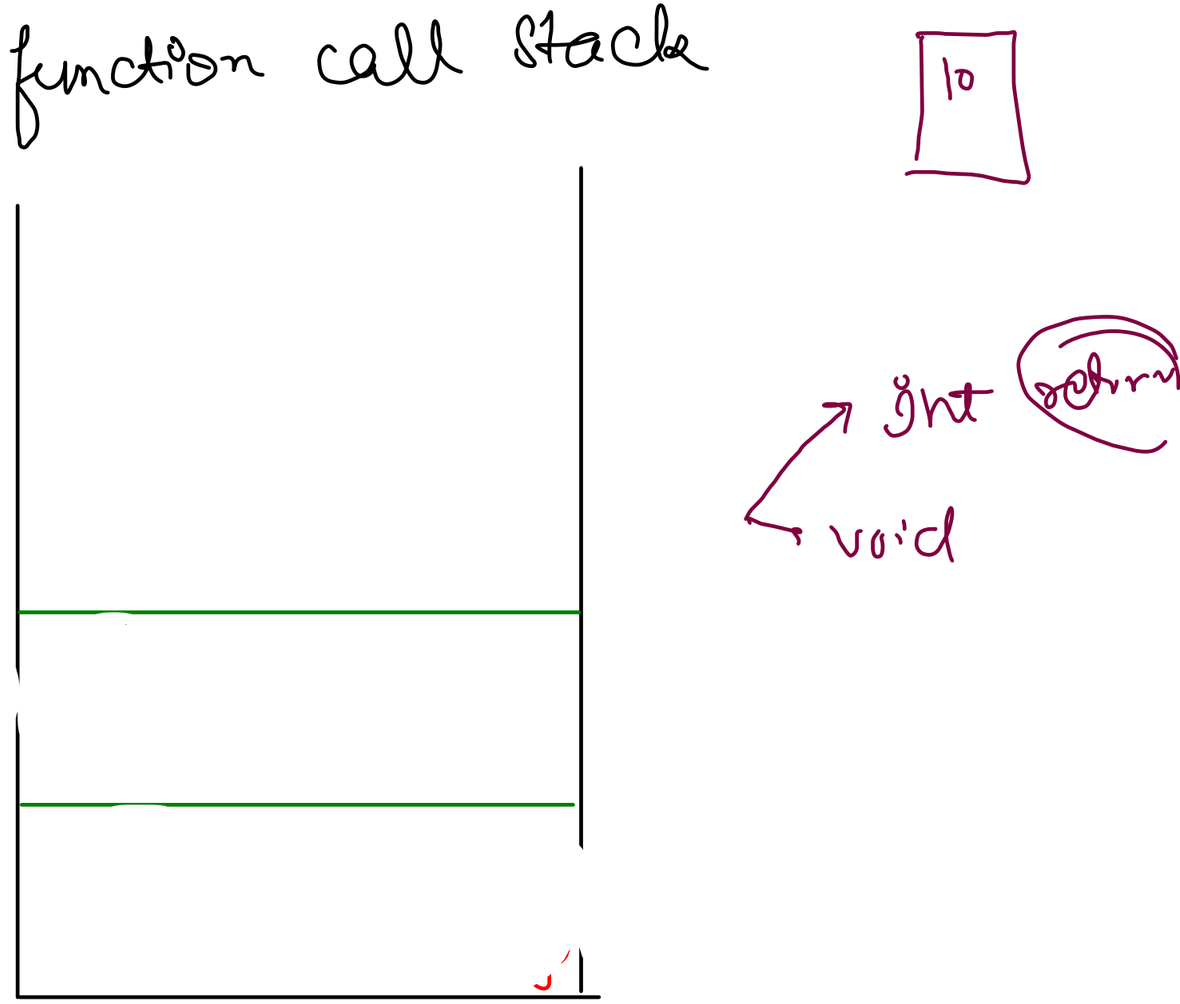
functionname



OS

Program under execution is known as process.

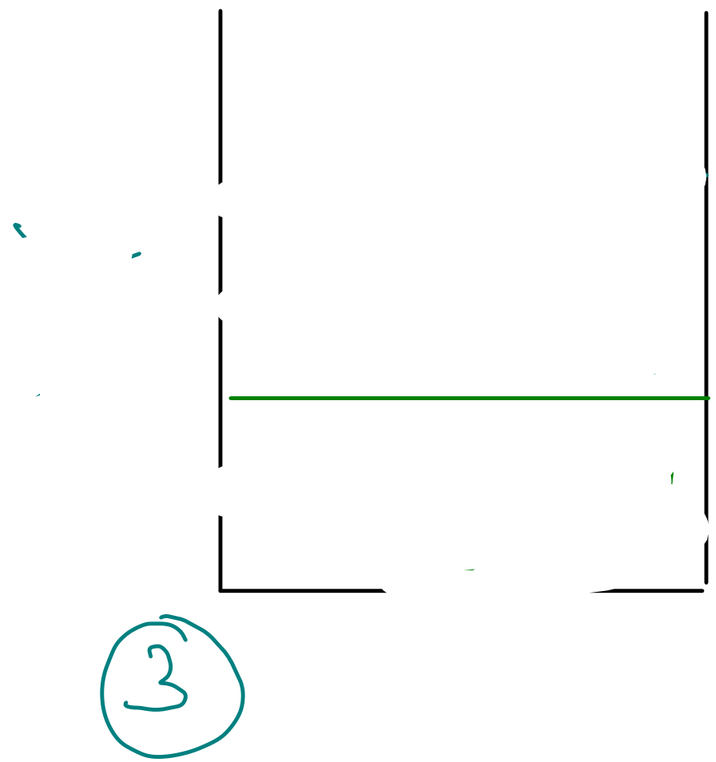
```java
public static int factorial(int n){
    int res = 1;

    for(int i=1; i<=n; i++){
        res = res * i;
    }

    return res;
}

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

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


    int nfact = factorial(n);
    int rfact = factorial(r);
    int nmrfact = factorial(n - r);


    int ncr = (nfact / (rfact * nmrfact));
    System.out.println(ncr);
}
```



n = 1292126429

d = 2

freq = 0 1 2 3



3

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int d = scn.nextInt();
    int f = getDigitFrequency(n, d);
    System.out.println(f);
}


public static int getDigitFrequency(int n, int d){

    int freq = 0;

    // For Extracting Digits
    while(n > 0){
        int digit = n % 10;

        if(digit == d){
            freq++;
        }

        n = n / 10;
    }

    return freq;
}
```

Number system

Decimal No

0-9

10 digits

$10 = 1*10^1 + 0*10$

100 → 99

$100 =$

25169_3

$= 6*10^2 + 9*10^1 + 3*10^0$

$+ 1*10^3 + 5*10^4$

$+ 2*10^5$

0 1 2 3 4 9 — 10 11 12 ---- 20 — 21 22 ---- 30 — 99 — 100

0-9 0-9 0-9 0-5

- min no 0000
- max no 9999
- count of nos 10^4

Binary No

2

0 1

00
01
10
11

0
1

000
001
010
011
1100
101
110
111

2^2 2^1 2^0

$1101011 = 1*2^0 + 1*2^1 + 0*2^2 + 1*2^3 + 0*2^4 + ---$

$2*2*2*2$

0/1 0/1 0/1 0/1

min 0000

max 1111

Count 2^4

Octal Nb System

↳ 0, 1, 2, 3, 4, 5, 6, 7

$(7\ 5\ 3\ 4\ 6\ 1\ 2)_8$

$= 2*8^0 + 1*8^1 + 6*8^2 + 4*8^3 + 3*8^4 + 5*8^5 + 7*8^6$

0	10	20	30	40 50 60
1	11	21	37	
2	12	⋮	⋮	47 57 67
3	13	⋮		
4	14			
5	15	27	37	
6	16			
7	17			

70
77

100	777
101	1000
102	7777
107	
	10000

$7*8^2 + 7*8^1 + 7*8^0$

$(777)_8$ +

1000

$1*8^3$

$(999)_{10} = 1000$

$9*10^2 + 9*10^1 + 9*10^0$

$1*10^3$

8 * 8 * 8 * 8

min ⇒ 0000

max ⇒ 7777

total ⇒ 8^4

6 months + leetcode (L2) Octal

pepcoding

L1

L2 ⇔ gfg → 9m

service based

tier 3 ⇐

tier 1 (3rd, 4th)

DSA

GFG 200

2-3 hr

8 hr

{ DSA { 600+ leetcode }
Dev { 50 + Project } }

2-3 Dev

150

backlogs

DSA

80-90

40-50

Today's Target

- Decimal To Any Base ⇒ classwork
- Any Base To Decimal ⇒ classwork
- Any Base To Any Base ⇒ homework
- Any Base Addition ⇒ classwork
- Any Base Subtraction ⇒ classwork
- Any Base Multiplication ⇒ homework

dividend

Decimal to Any Base

$(634)_{10}$ = $(?)_8$

$\uparrow 61$ $\uparrow 62$

→ What ?
→ how ?
why ?

b	N	rem	multiplier
8	634		$10^0 = 1$
8	79	2	10^1
8	9	7	10^2
8	1	1	10^3
	0		

$79 = 9 * 8 + 7$

$\{0+2\}+\{70\}+100+1000$

$= 1172$

$(1172)_8$

```
public static int getValueInBase(int n, int b){
    int res = 0;
    int multiplier = 1;

    while(n > 0){
        int divisor = n / b;
        int remainder = n % b;

        n = divisor;
        res = res + remainder * multiplier;
        multiplier *= 10;
    }

    return res;
}
```

$(634)_{10}$

$6*10^2 + 3*10^1 + 4*10^0$

$=$

$(1172)_8$

$1*8^3 + 1*8^2 + 7*8^1 + 2*8^0$

10:20

$(634)_{10} = 8^1 * (-) + 2$

$= 8^4*0 + 8^3*1 + 8^2*1 + 8^1*7 + 8^0*2$

$79 = 8^1 * (8^2*0 + 8^1*1 + 8^0*1) + 7$

$= (8^3*0 + 8^2*1 + 8^1*1 + 8^0*7)$

$634 = 8^1 * 79 + 2$

$79 = 8^1 * 9 + 7$

$9 = 8^1 * 1 + 1$

$1 = 8^1 * 0 + 1$

$9 = 8^1 * (8^1*0 + 1) + 1$

$9 = 8^2*0 + 8^1*1 + 8^0*1$

$(173)_{10} = (?)_8$

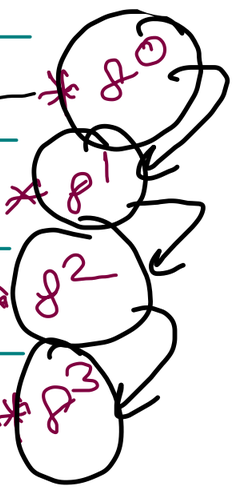
8	173
8	21 − 5 * 10^0
8	2 − 5 * 10^1
	0 − 2 * 10^2

$= 5 + 50 + 200 = (255)_8$

Any Base to Decimal

$(1172)_8 = (?)_{10}$

10	1172
10	117 — 2
10	11 — 7
10	1 — 1
	0 — 1



$= 2*8^0 + 7*8^1 + 1*8^2 + 1*8^3$

$= (634)_{10}$

```java
public static int getValueIndecimal(int n, int b){
    int res = 0;
    int multiplier = 1;

    while(n > 0){

        int divisor = n / 10;
        int remainder = n % 10;

        n = divisor;

        res = res + remainder * multiplier;
        multiplier *= b;
    }

    return res;
}
```

$(110010)_2 = (?)_{10}$

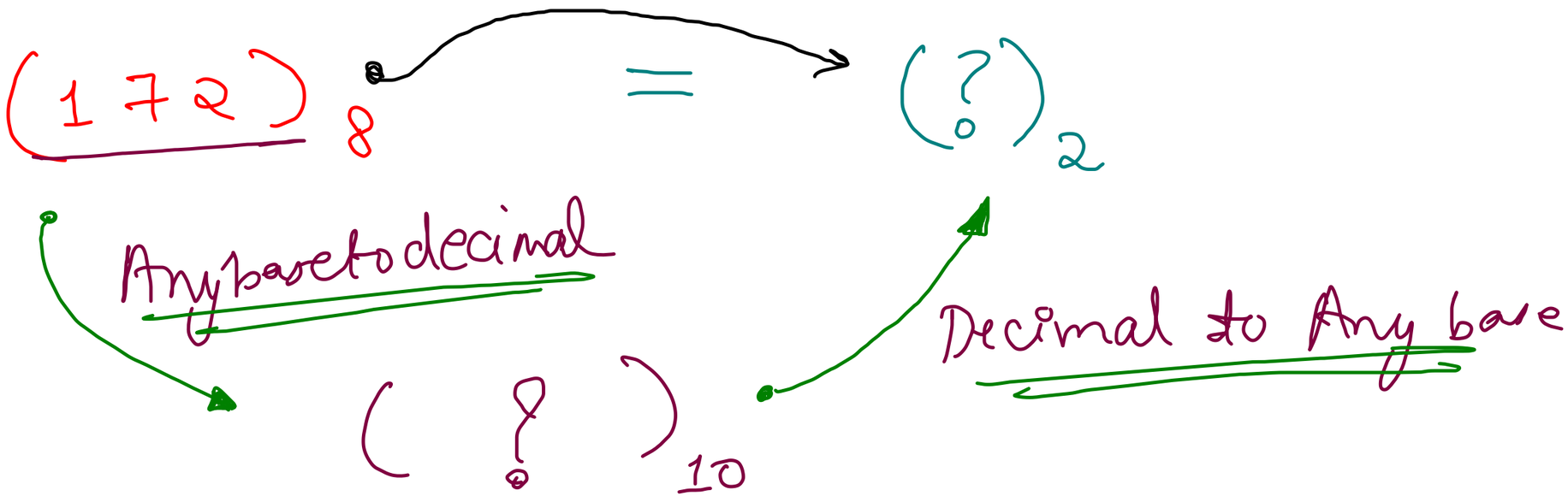
10	110010	
10	11001	$-0 * 2^0$
10	1100	$-1 * 2^1$
10	110	$-0 * 2^2$
10	11	$-0 * 2^3$
10	1	$-1 * 2^4$
10	0	$-1 * 2^5$

$= 0 + 2 + 0 + 0 + 16 + 32$

$= 18 + 32$

$= (50)_{10}$

Any base to any base



Any Base addition

carry 1 0 1 0

$(4\ 3\ 6)_{10}$

$+\ (7\ 5\ 4)_{10}$

$(1\ 1\ 9\ 0)$

temp = (0 + 6 + 4)

$\frac{10}{10}$ → 1 {quot}, 0 {rem}

1 + 3 + 5 = $\frac{9}{10}$ → 0, 9

0 + 4 + 7 = $\frac{11}{10}$ → 1, 1

Carry 1 1 1 0

0 (2 3 6)8

0 (7 3 4)8

---

1 2 1 2

$\{2*10^0 + 1*(0^1 + 2*10^2 + 1*10\}$

1212 = 1000 + 200 + 10 + 2

236 → 23

236/10 = 23

① st

c d1 d2

6 + 6 + 4

= 16 → 1 (quot), 6 (rem)

8

② nd

1 + 3 + 5

= 9 → 1, 1

8

③ rd

1 + 2 + 7 = 10 → 1, 2

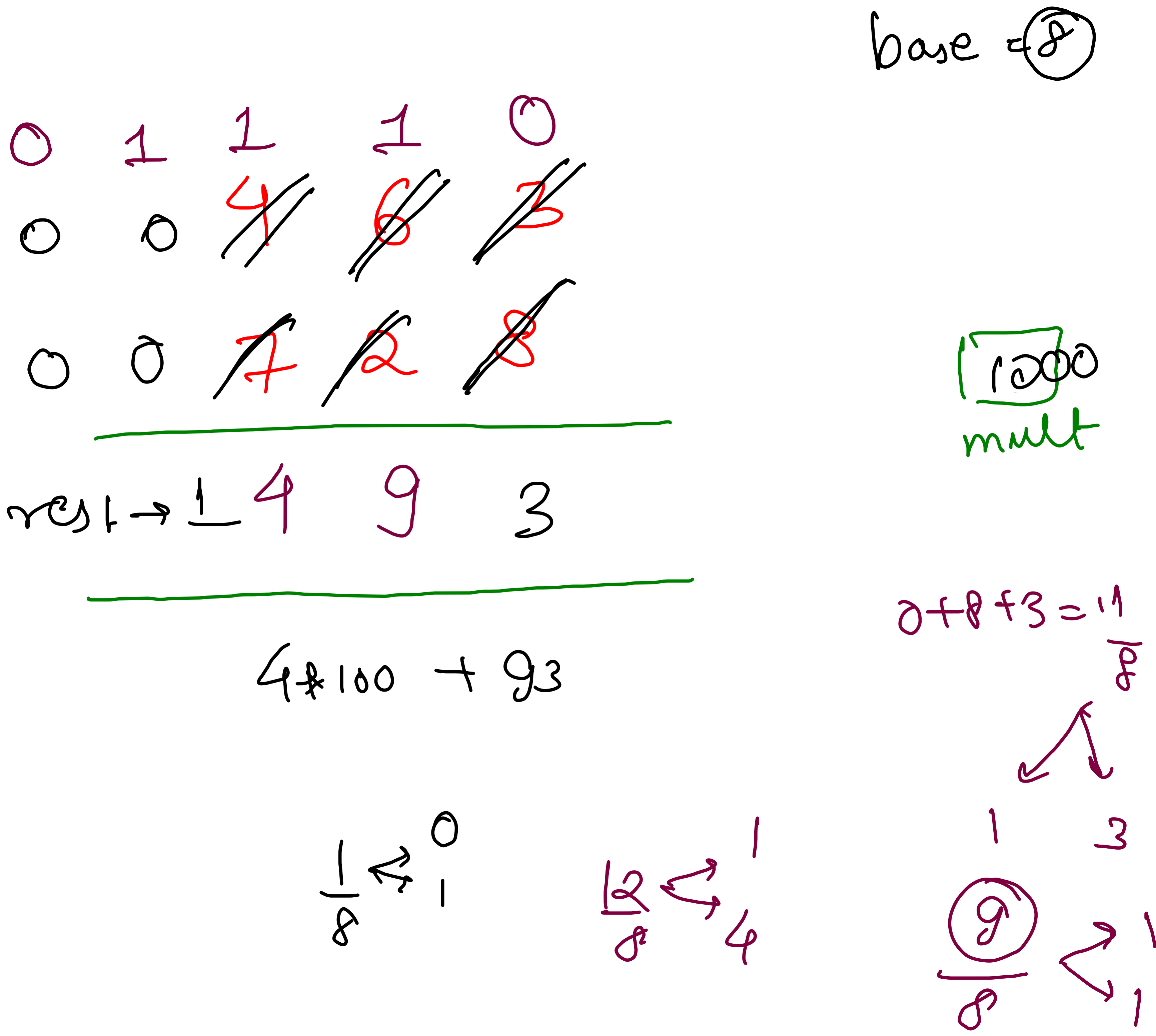
8

④ th

1 + 0 + 0 = 1 → 0, 1

8

```
public static int getSum(int b, int n1, int n2){
    int res = 0;
    int multiplier = 1;
    int carry = 0;

    while(carry > 0 || n1 > 0 || n2 > 0){

        int d1 = n1 % 10;
        int d2 = n2 % 10;

        int temp = carry + d1 + d2;
        int quot = temp / b;
        int rem = temp % b;

        res = res + (rem * multiplier);
        carry = quot;

        multiplier *= 10;
        n1 /= 10;
        n2 /= 10;
    }

    return res;
}
```

Pseudo Code

```
carry = 0
res = 0
multiplier = 1

while(carry || n1 > 0 || n2 > 0) {
    d1 = n1 % 10
    d2 = n2 % 10
    temp = carry + d1 + d2
    quot = temp / base
    rem = temp % base

    carry = quot
    res += rem * multiplier

    multiplier *= 10
    n1 /= 10
    n2 /= 10
}
```

quotient will become next carry

&

rem will be Contributing in result.

Any base substraction

borrow

$$\begin{array}{cccc} -1 & -1 & -1 & \\ (1 & 2 & 1 & 2)_{10} \\ - & (2 & 5 & 6)_{10} \\ \hline 0 & 9 & 5 & 6 \end{array}$$

$d1 - d2 + \text{borrow}$

$2 - 6 + 0 = -4$

$2-6 = -4+10$

$= 6$

gfl

$2-2+(-1)+10$

$= -1+10 = 9$

$1-5+(-1)$

$= -4-1 = -5+10$

$= 5$

Any base subtraction

borrow

$$\begin{array}{cccc} -1 & -1 & -1 & 0 \\ (1 & 2 & 1 & 2)_{10} \\ - & (2 & 5 & 6)_{10} \\ \hline 0 & 9 & 5 & 6 \end{array}$$

$-4+0$

$2-6 = -4+10$

$= 6$

Q − d1 + borrow

$2 - 6 + 0 = -4 + 10$

$= 6$

gf1

$2-2+(-1)+10$

$= -1+10 = 9$

$1 - 5 + (-1)$

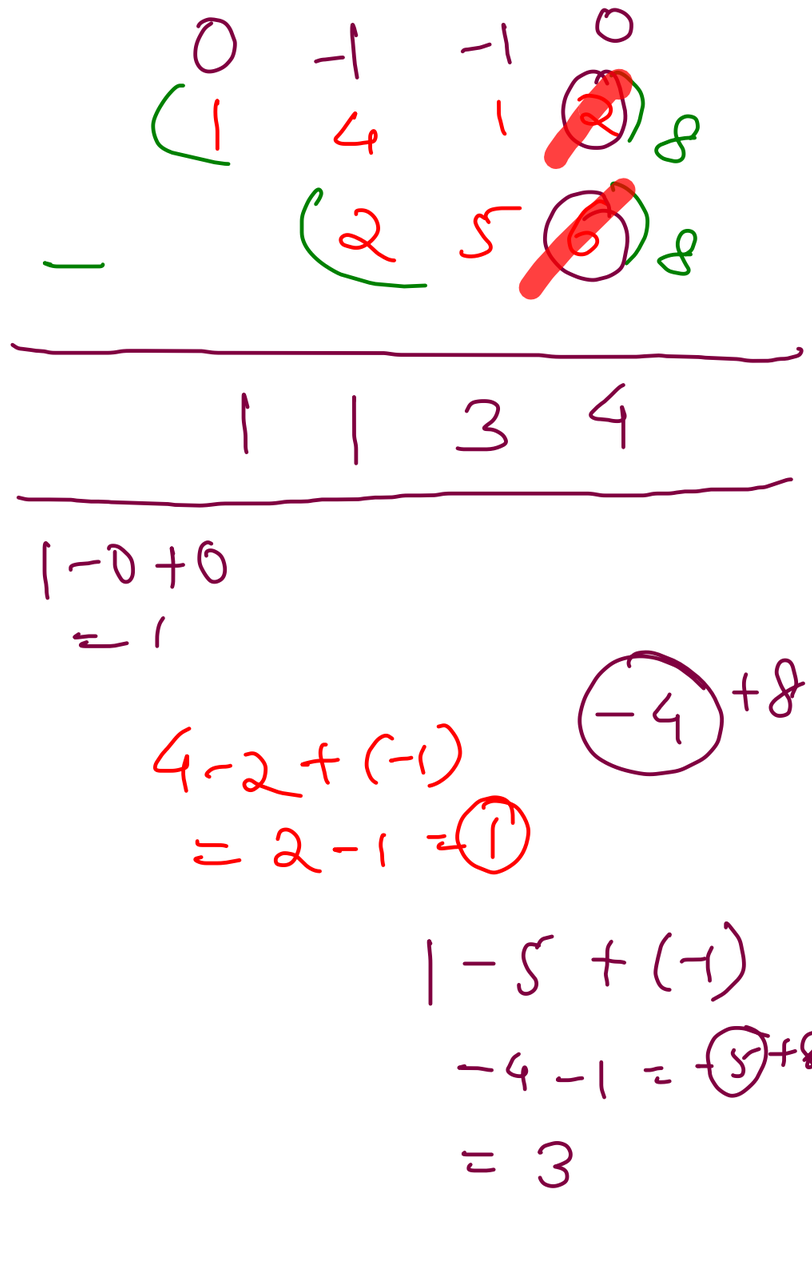
$= -4 - 1 = -5 + 10$

$= 5$

Pseudo code

$n2 > n1$

```
borrow = 0      res = 0      multiplier = 1

while( n2 > 0) {

    d1 = n1%10 , d2 = n2%10

    temp = d2 - d1 + borrow

    if(temp < 0) {
        res += (temp + base) * multiplier;
        borrow = -1;
    } else {
        res += temp * multiplier;
        borrow = 0;
    }

    multiplier *= 10;
    n1 /= 10;   n2 /= 10;
}
```



Any base Multiplication

$$(2\ 1\ 5\ 6)_{10} \times (7\ 4)_{10}$$

1	1	3	4	0	
	(2	1	5	6)$_{10}$	
X			(7	4)$_{10}$	
21564	{ 1	8	6	2	4	7
215620	1	5	0	9	2	X
	1	5	9	5	4	4

Any base addition

$7*2+1 = 15_{10}$ → 1, 5

$7*6 = 42_{10}$ → 4, 2

$\frac{24}{10}$ → 2, 4

$\frac{39}{10}$ → 3, 9

$5*4+2 = \frac{22}{10}$ → 2, 2

$4*1+2 = \frac{6}{10}$ → 0, 6

$\frac{8}{10}$ → 0, 8

res

Algorithm

while{ n1->0 }{

① Extract digits of n1 : d1

② multiply d1 with n2 : temp → getProductwithDigit( )

③ res = anybaseAddition(res, temp, multiplier)

multiplier *= 10

}

pencil eraser

$$
\begin{array}{cccccc}
 & 1 & 1 & 5 & 5 & 0 \\
 & (2 & 1 & 5 & 6)_8 & \\
\times & & & (7 & 4)_8 & \\
\hline
1 & (1 & 0 & 6 & 7 & 0)_8 \\
(1 & 7 & 4 & 0 & 2 & 0)_8 \\
\hline
2 & 0 & 4 & 7 & 1 & 0
\end{array}
$$

2156 * 74

= 2156 * 4 (+) 2156 * 7 * 8

octal

$\frac{24}{8}$ → 3, 0

20 + 3 = $\frac{23}{8}$ → 2, 7

$\frac{42}{8}$ → 5, 2

$\frac{6}{8}$ → 0, 6

$\frac{40}{8}$ → 5, 0

$\frac{8}{8}$ → 1, 0

$\frac{12}{8}$ → 1, 4

$\frac{1}{8}$ → 0, 1

$\frac{15}{8}$ → 1, 7

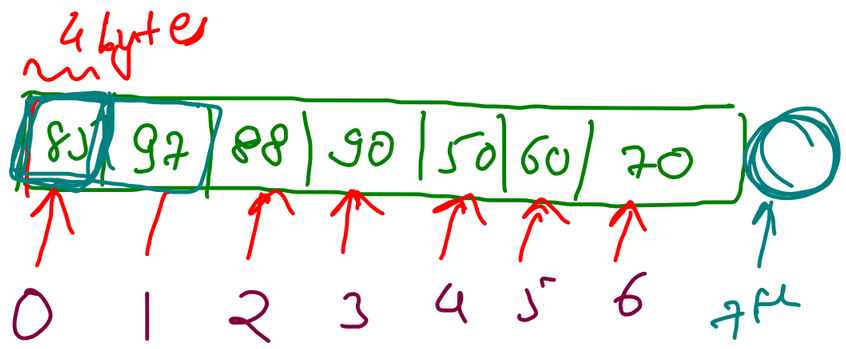
Data structure

Arrays

int m1 = 95;

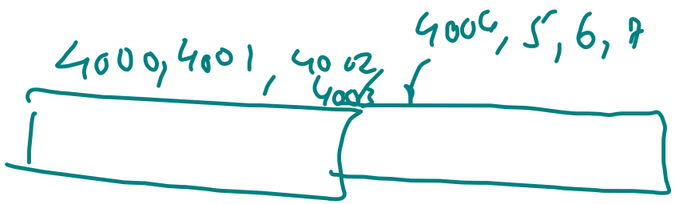
int m2 = 97;

int m3 = 88;



contiguous memory allocation

7 * 4 = 28 byte



7 size

0 - 6

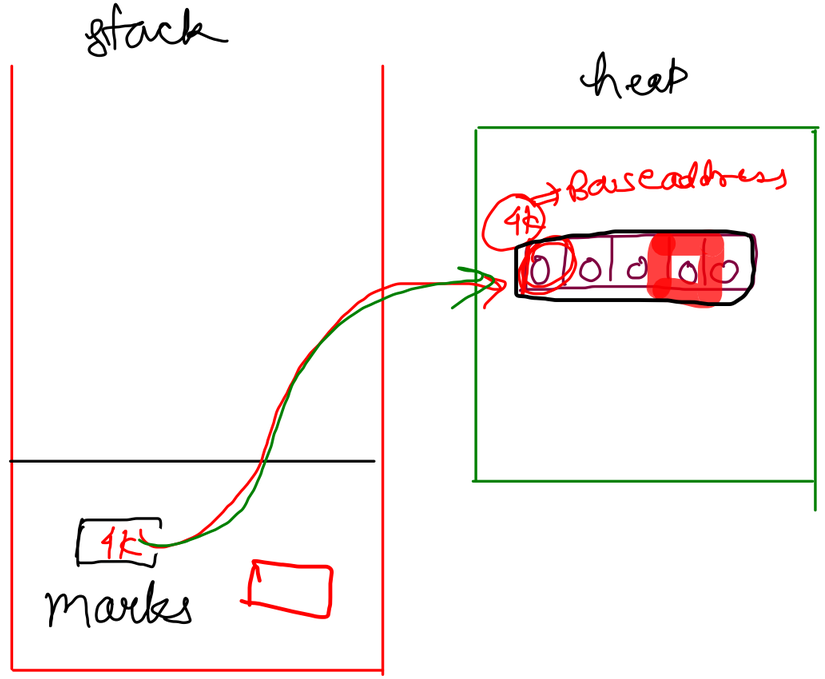
Memory mapping of arrays

```
int[] marks = new int[n];
```

{Static array}

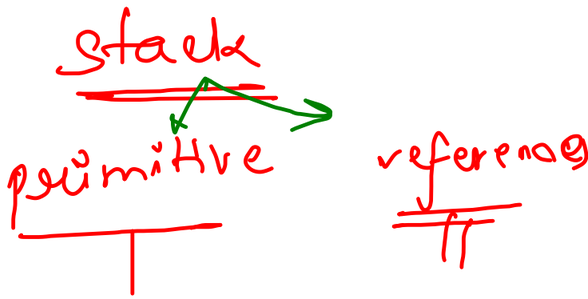
marks [i]

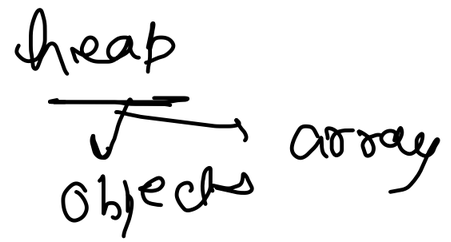
base address of array + ith block * 4 bytes



Random access {constant operation}

:=> BA + index * size of one block





marks [0]

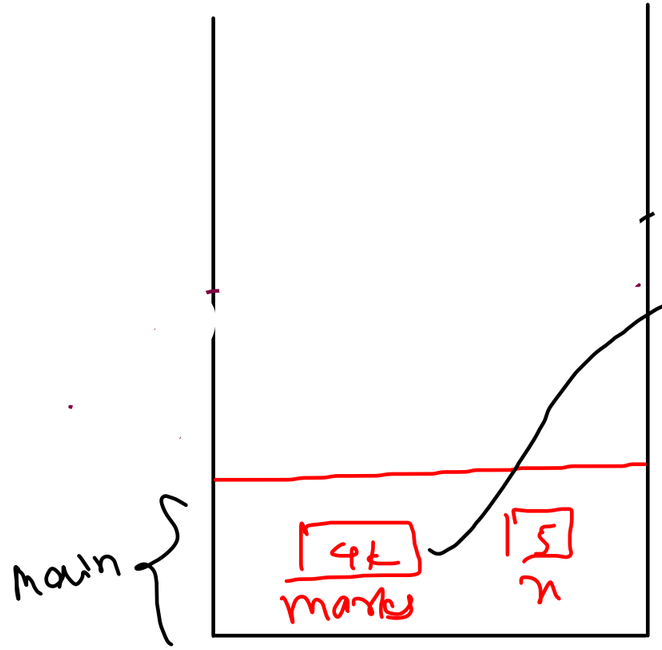
4k + 0*4 = 4k

marks[3]

4K + 3*4 = 4k12

Passing array to function

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

}

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

    int n = scn.nextInt();

    // Declaration & Initialization
    int[] marks = new int[n];

    // Random Access using Indexing
    for(int i=0; i<n; i++){
        marks[i] = scn.nextInt();
    }

    // Array Traversal
    display(marks);
}
```

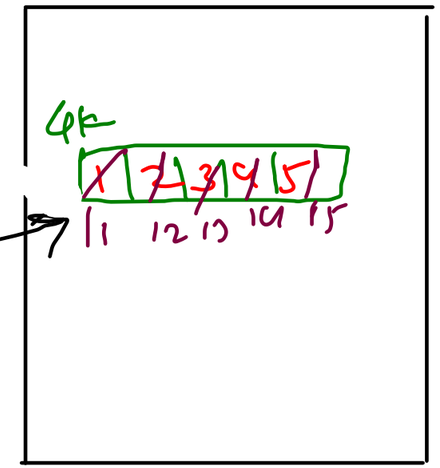


"Changes in the heap always persist"

fcnt call stack

Shallow copy

permanent



"Java is always pass by value"

Shallow copy

Java

8 primitive

4byte int

4byte float

8byte double

1byte byte

1byte boolean

2byte character

8byte short

8byte long

(-∞) Integer.MIN_VALUE

int

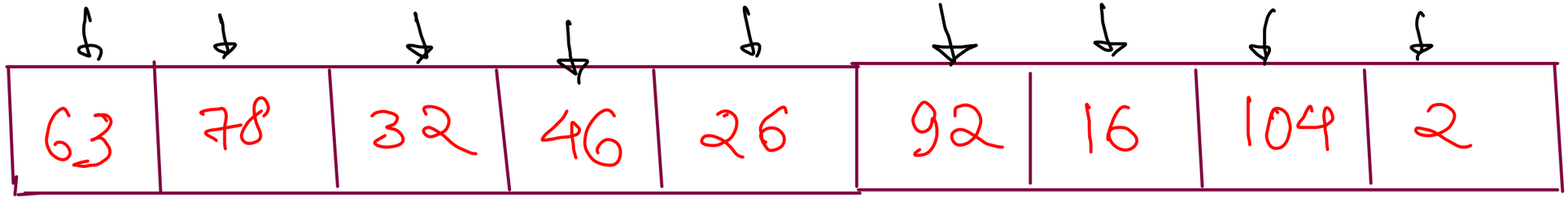
-2^{31} to $+2^{31}-1$

Integer.MAX_VALUE (+∞)

$\approx 10^9$

long

-2^{63} to $2^{63}-1$



$span = max^m - min^m$

$max^m =$ ~~$-\infty$~~ ~~63~~ ~~78~~ ~~92~~ 104

$min^m =$ ~~$+\infty$~~ ~~63~~ ~~32~~ ~~26~~ ~~16~~ 2

```java
int min = Integer.MAX_VALUE;
int max = Integer.MIN_VALUE;

for(int i=0; i<n; i++){

    // is current element less than min of previous
    if(arr[i] < min){
        min = arr[i];
    }

    // is current ele greater than max
    if(arr[i] > max){
        max = arr[i];
    }
}

int span = max - min;
System.out.println(span);
```

identities

$x + 0 = x$

$x * 1 = x$

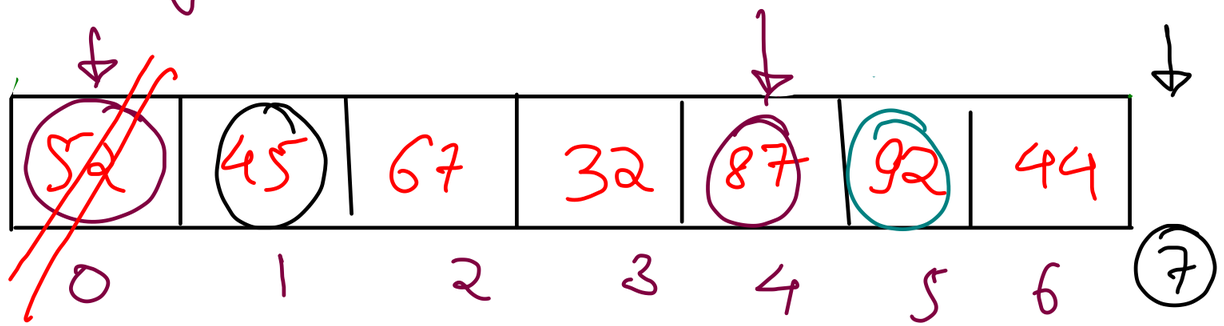
$max(x, -\infty) = x$

$min(x, +\infty) = x$

Today's questions

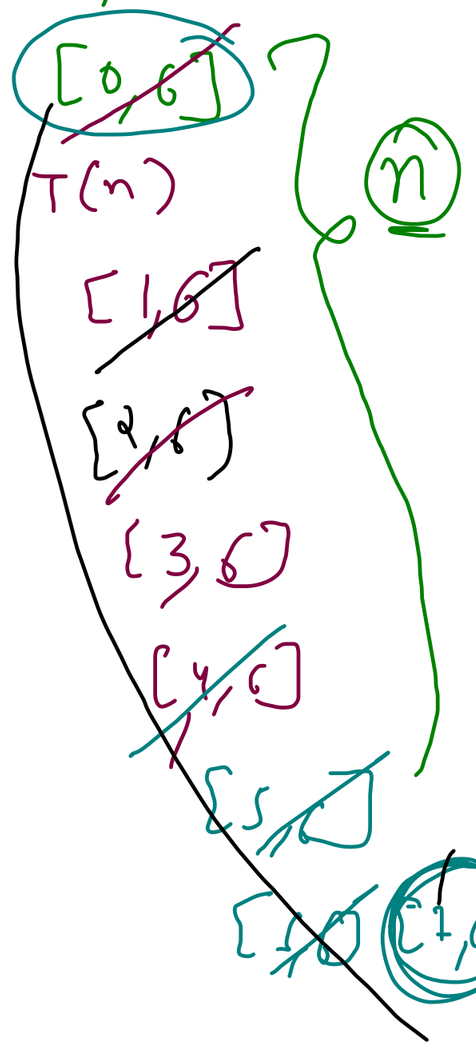
- </> Find Element In An Array → Linear Search Algorithm
- </> Bar Chart
- </> Sum Of Two Arrays
- </> Difference Of Two Arrays
- </> Reverse An Array
- </> Rotate An Array

Linear Search Algorithm



ele = 87 92 5?

search space



Execution Time / Run Time

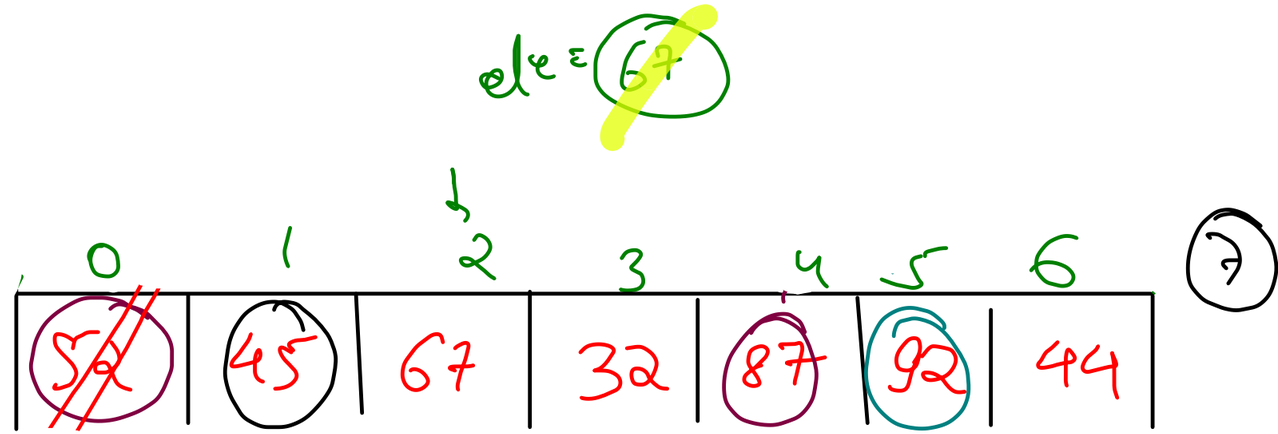
- 1 comp ⇒ best case
- n comp ⇒ worst case
- avg ; $\frac{n}{2}$ ⇒ avg case

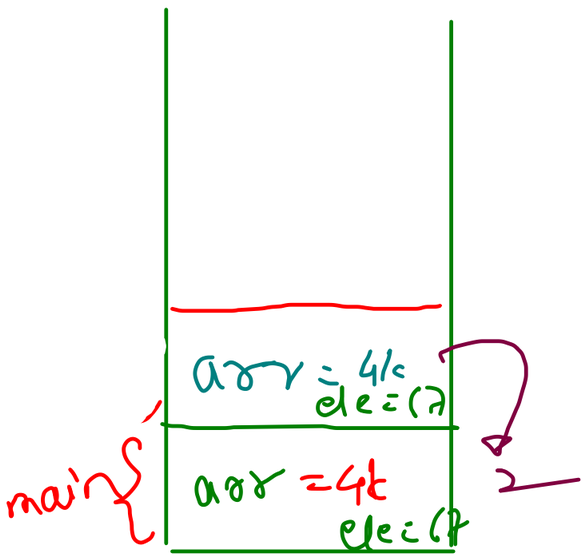
```
for(idx=0; idx < arr.length; idx++)
    if(arr[idx] == ele){
        return idx;
    }
}
return -1;
```

successful

unsuccessful search

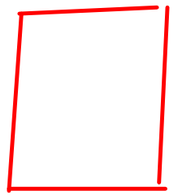
```java
public static int linearSearch(int[] arr, int ele) {

  for (int idx = 0; idx < arr.length; idx++) {

    if (arr[idx] == ele) {
      // successful search
      return idx;
    }

  }

  return -1; // unsuccessful search (element not found)
}
```





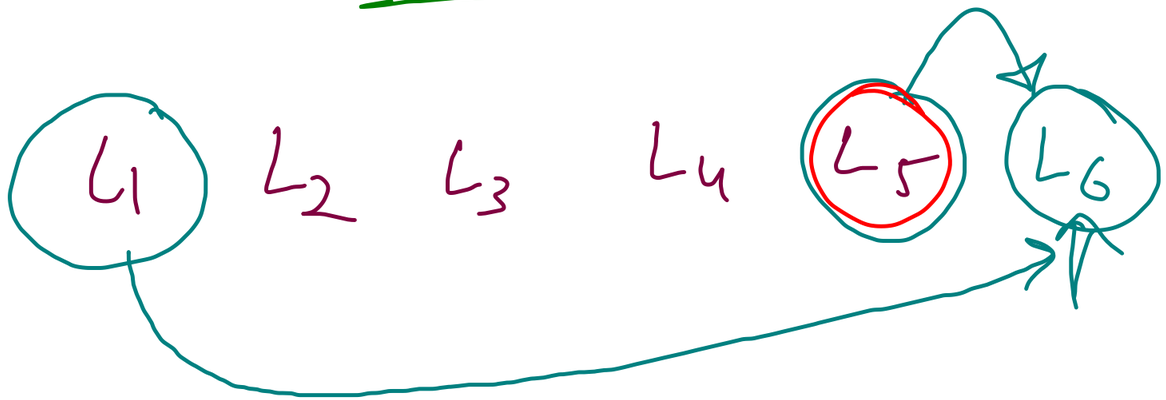

```java
public static void main(String[] args) throws Exception {
  Scanner scn = new Scanner(System.in);
  int n = scn.nextInt();

  int[] arr = new int[n];

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

  int ele = scn.nextInt();

  int ans = linearSearch(arr, ele);
  System.out.println(ans);
}
```

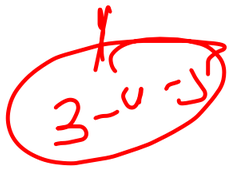
Notes

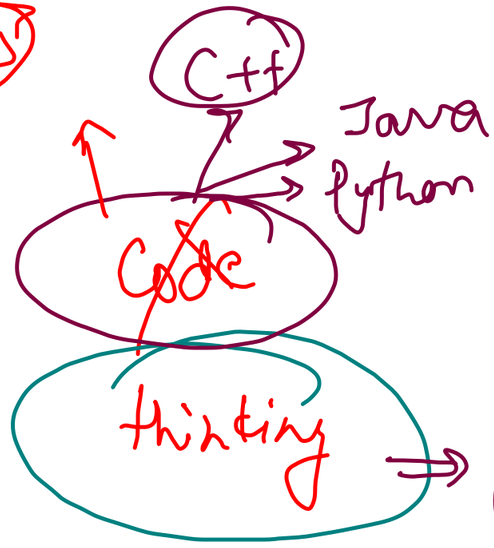


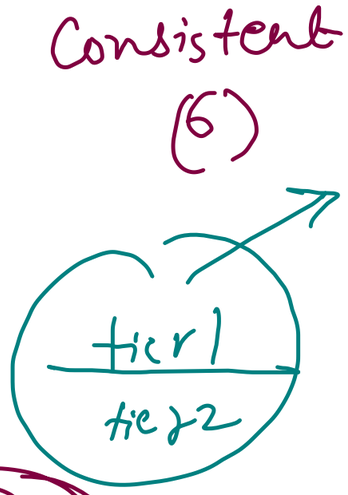
Question name
1 liner logic
Important points

Revision









DSA + Dev + Core

SB-C. (3-6)	Small Startups (6-12)	big startups (12-30)	PBC	tech giant {FAANG}
Aptitude Level 1	Level 1	Level 1 + 2	1 + 2	1 + 2
20-30%	Dev → So-Co	Dev → 30-40%	Dev	Dev
		Core	Core	Core

70-80

Prepading
leetcode
Level 1 2
MS
Virtual contest

AWS/CS
1
80-90%
Core
500
kickstart
CP 3
1
Google
Directi
Media.net
Tower Research
Uber

10:30

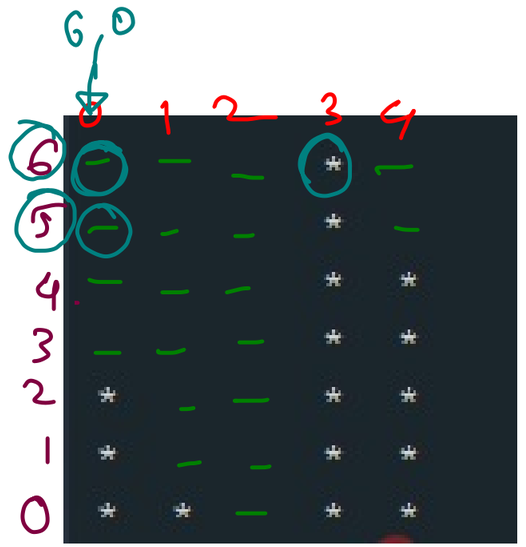
Codechef
Bit tricks DP/graphs

Tree, LL, Stack
leetcode

fin-tech
1
Aptitude
Maths + Level 1 2
Core

Bar Chart-

{ 3, 1, 0, 7, 5 }




```
int totalCols = n;
int totalRows = maxEle(arr);

for(int i=totalRows - 1; i>=0; i--){

    for(int j=0; j<totalCols; j++){

        if(/* ? */){
            System.out.print("*\t");
        } else {
            System.out.print("\t");
        }

    }
}
```

i < arr[j]

end of

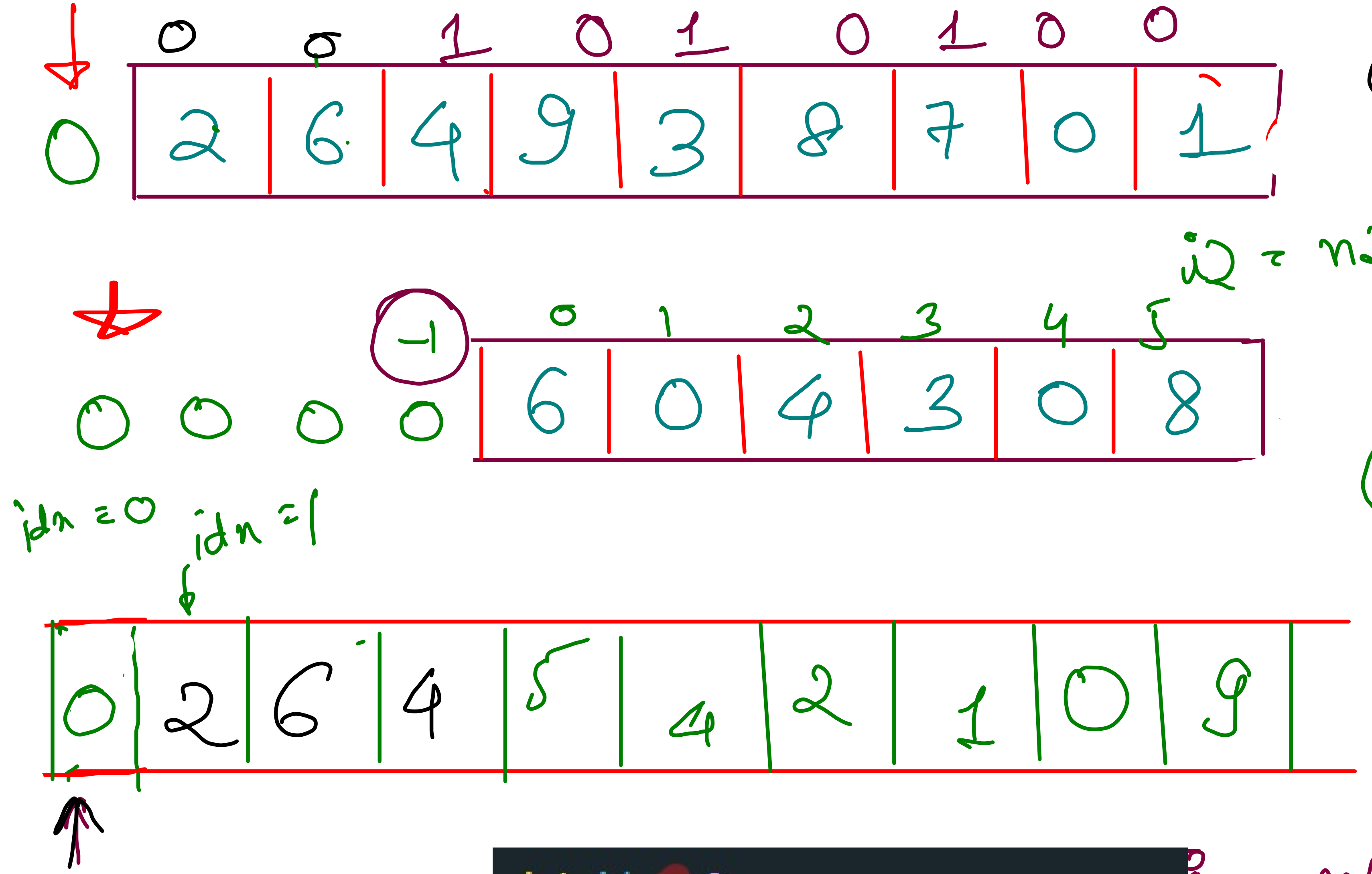
i >= arr[j] : space

else i < arr[j] : star

Sum of two arrays

i1 = n1-1

d1 = arr1[i1]



i2 = n2-1

carry = 0

d2 = arr2[i2]

arr2[-1] ← index out of bound exception

long {8 bytes} → $2^{63}-1$ ≈ 10^{18}

$>10^{18}$ → Arrays, BigInteger

factorial of "n"

n = 100!

i3 = n3-1

size = max(n1, n2)+1

```
int idx = 0;
while(res[idx] == 0){
    idx++;
}

for(int i=idx; i<res.length; i++){
    System.out.println(res[i]);
}
```

Pseudo Code

int [] res = new int [max(n1, n2) + 1]

carry = 0 i1 = n1-1 , i2 = n2-1 , i3 = n3-1

while (i3 >= 0) {

d1 = arr1[i1]
d2 = arr2[i2]

temp = carry + d1 + d2
quot = temp / 10
rem = temp % 10

carry = quot
res[i3] = rem;

i1--; i2--; i3--;

}

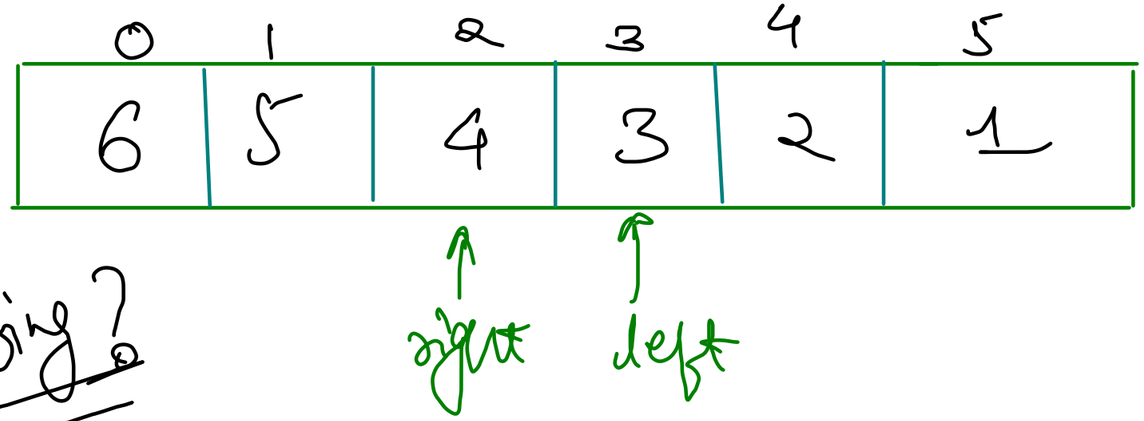
if (i1 < 0) d1 = 0
else d1 = arr1[i1]

if (i2 < 0) d2 = 0
else d2 = arr2[i2]

Today's Questions

- </> Reverse An Array
- </> Rotate An Array
- </> Inverse Of An Array
- </> Subarray Problem
- </> Subsets Of Array

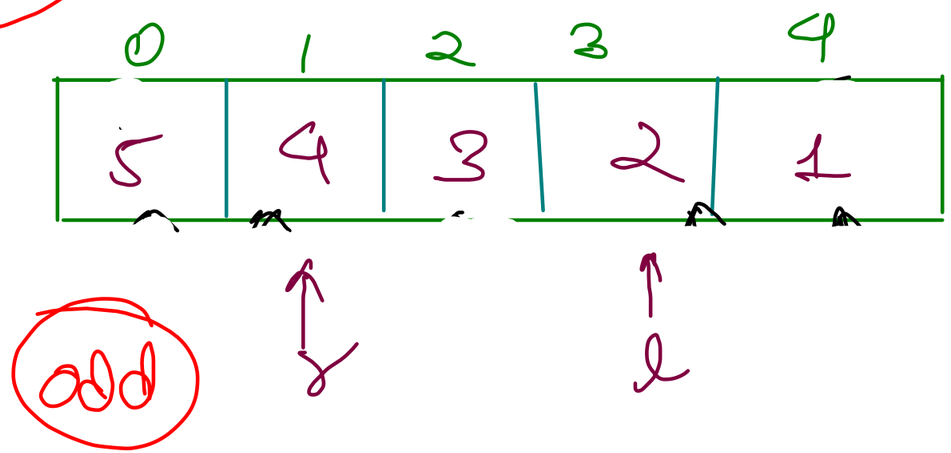
Reverse Array



even

swapping?

n/2



odd

l > r

l == r

swap(arr, left, right);

temp = arr[left]

arr[left] = arr[r]

arr[r] = temp

}

```java
public static void swap(int[] arr, int left, int right){
    int temp = arr[left];
    arr[left] = arr[right];
    arr[right] = temp;
}

public static void reverse(int[] a){
      int left = 0, right = a.length - 1;

      while(left < right){
          swap(a, left, right);
          left++; right--;
      }
}
```

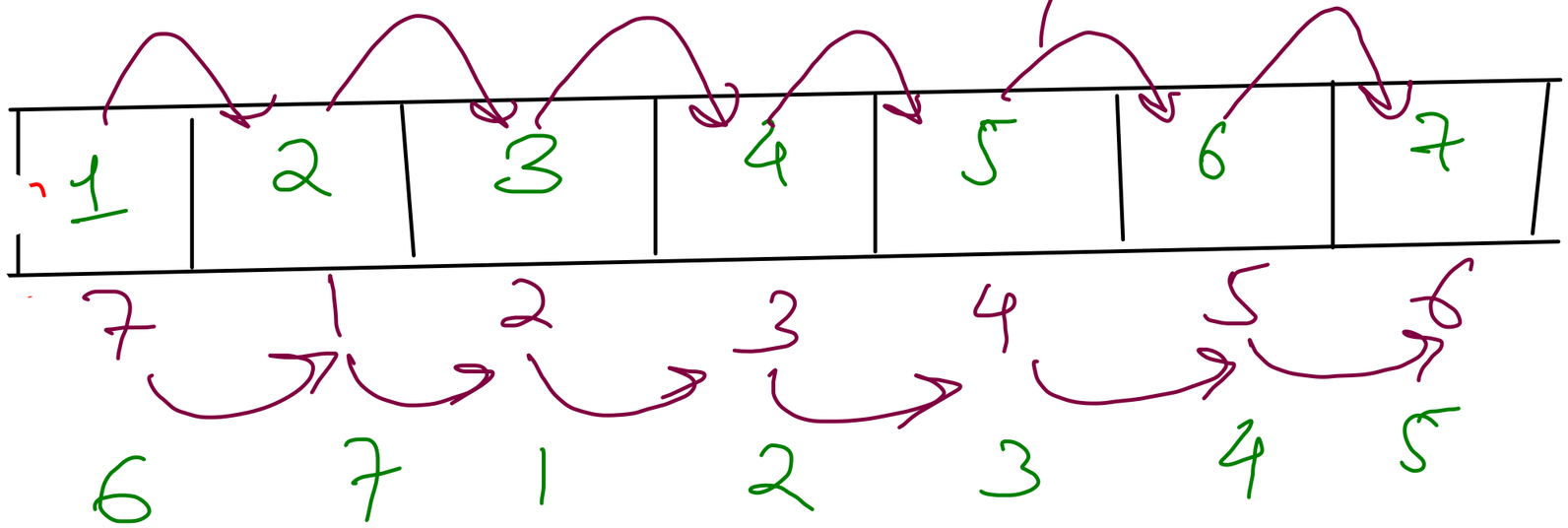
```java
public static void swap(int[] arr, int left, int right){
    int temp = arr[left];
    arr[left] = arr[right];
    arr[right] = temp;
}

public static void swap(int val1, int val2){
    int temp = val1;
    val1 = val2;
    val2 = temp;
}
```

① heap change

② stack change

Rotate an Array

Inplace → without extra space



k = 18

[0, 7-1]

k = 1 . n

if (k < 0)
k = k + n

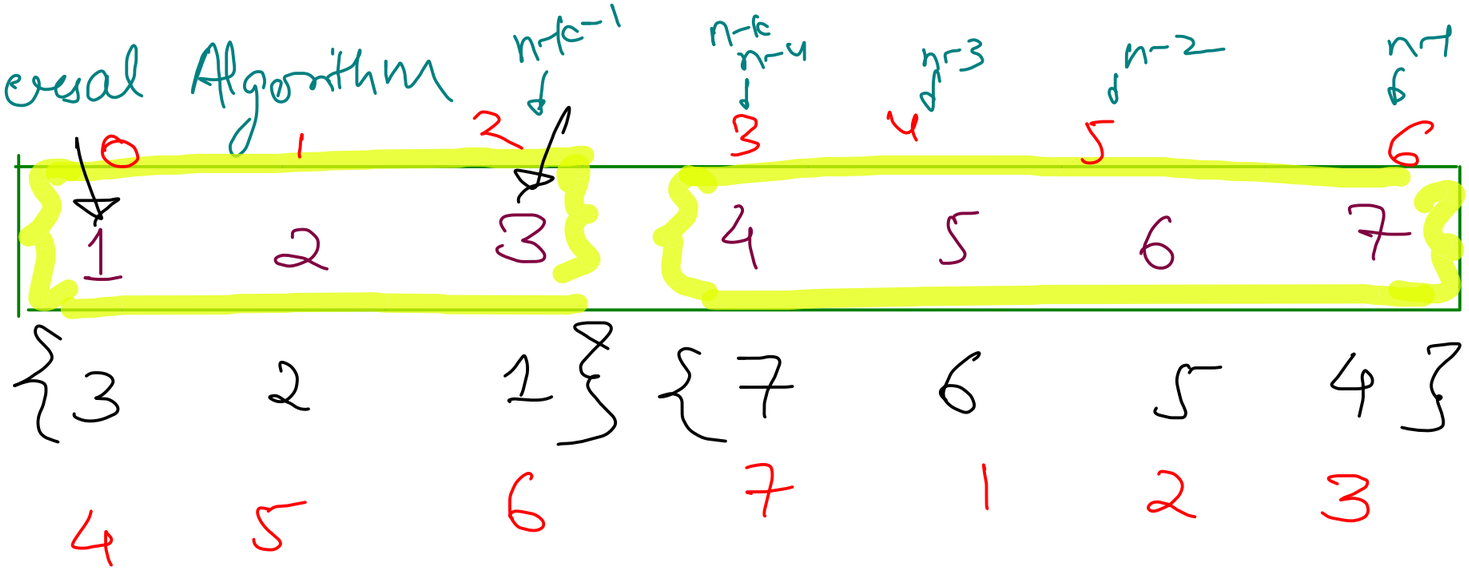
(4)

(1) Brute force

{Rotate Array k times} by 1 place

↳ Naive Approach → O(k * n)

(2) Reversal Algorithm



$O(n)$ {Swaps in worst Case}

(1) Reverse (arr, 0, n-k-1) $\frac{(n-k)}{2}$

(2) Reverse (arr, n-k, n-1) $\frac{k}{2}$

(3) Reverse (arr, 0, n-1) $\frac{n}{2}$

$\frac{n-k}{2} + \frac{k}{2} + \frac{n}{2}$

```java
public static void reverse(int[] a, int left, int right){
        while(left < right){
            swap(a, left, right);
            left++; right--;
        }
}

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

        // Handle Larger K Values
        k = k % n;

        // Handle Negative K Values
        if(k < 0){
            k = k + n;
        }

        // 1. Reverse First Block
        reverse(a, 0, n - k - 1);

        // 2. Reverse Second Block
        reverse(a, n - k, n - 1);

        // 3. Reverse Entire Array
        reverse(a, 0, n - 1);
}
```

6 participants raised hand

Why ?

$A = A_1 + A_2$

$Res = A_2 + A_1$

$(A_1' + A_2')'$

$321 + 7654$

$A_2 + A_1$

Inverse of a No

Mock interview {for Inplace Algo}

	0	1	2	3	4	5	6
a[i]	2	4	1	3	6	0	5

a[i] ⇒ i

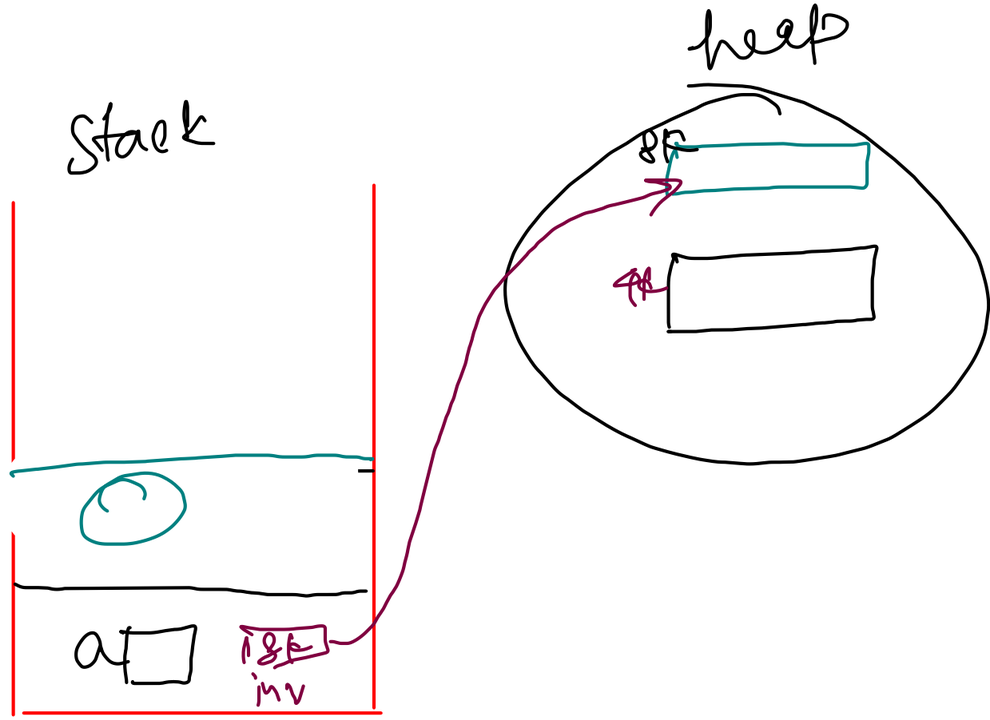
	0	1	2	3	4	5	6
b[j]	5	2	0	3	1	6	4

b[j] ⇒ i

Inplace {✓/✗}

✓new resultant array

```java
public static int[] inverse(int[] a){
  int n = a.length;
  int[] b = new int[n];

  for(int i=0; i<n; i++){
      // a[i] -> i
      int j = a[i];
      b[j] = i;
  }

  return b;
}
```



Subset / subsequence

each element
2 choices → ✓/✗

1	2	3	4	5

{1} {2} {3} {4} {5}

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

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

{1,4} {2,5}

{1,5}

total subset

2^n

Subarray / Substring

→ order must be same
→ Contiguous subpart of array

{1} {2} {3} {4} {5} ⑤

{12} {23} {34} {45} ④

{123} {234} {345} ③

{1234} {2345} ②

{12345} ①

total subarrays

$= n*n+1$

Today's Questions

1. subarrays } code
2. Sub sets
3. 2D array demo

- Declaration Initialization
- Input & Output
- memory mapping

10 20 30 40

10 20 30 40

20

20 30

20 30 40

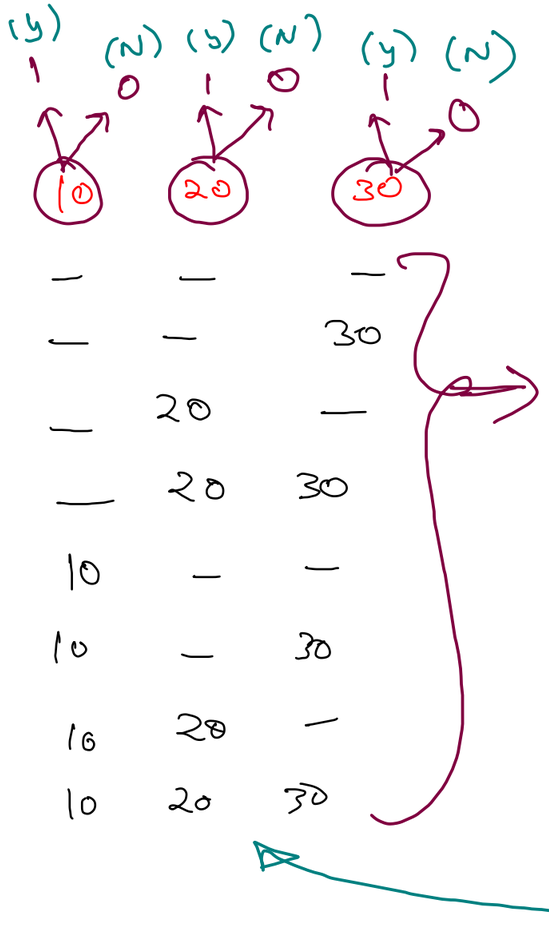
30

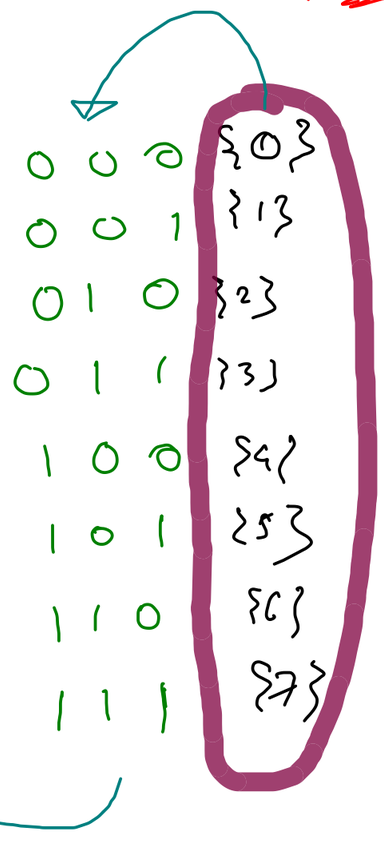
30 40

40

```java
for(int st=0; st<n; st++){
    for(int end=st; end<n; end++){
        for(int i=st; i<=end; i++){
            System.out.print(arr[i] + "\t");
        }
        System.out.println();
    }
}
```

Subsets





Recursion for loops

Steps

1. find total Subsets
   ts = 2^n
2. Run a loop on decimal no from 0 to ts-1
3. Convert decimal to its binary equivalent

011010

(binary / power) % 10

(011010 / 10^5) = 0 % 10 = 0

(011010 / 10^4) = 01 % 10 = 1

(011010 / 10^3) = 011 % 10 = 1

(011010 / 10^2) = 0110 % 10 = 0

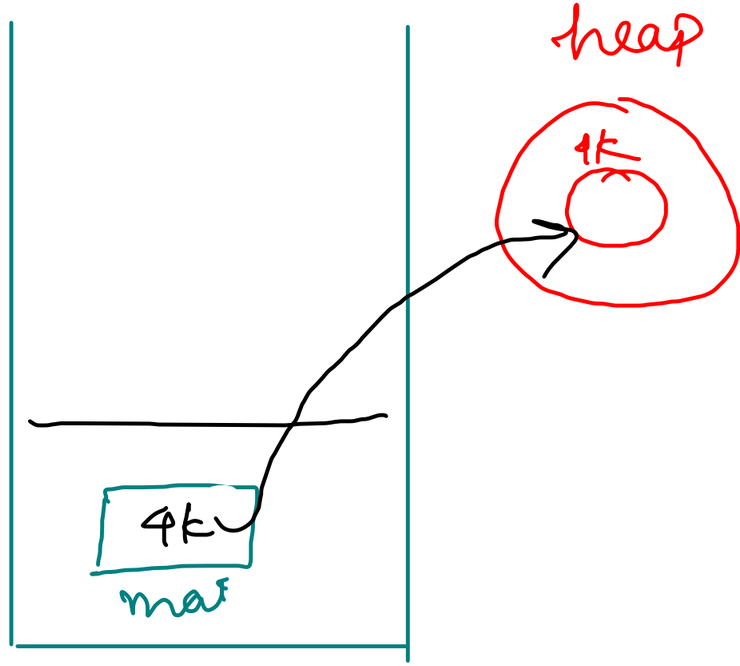
```java
int totalSubsets = (int)Math.pow(2, n);

for(int dec = 0; dec < totalSubsets; dec++){

    int binaryNo = decimalToBinary(dec);

    int power = (int)Math.pow(10, n - 1);
    // convert binaryNo to equivalent subset
    for(int i=0; i<n; i++){

        int bit = (binaryNo / power) % 10;

        if(bit == 1){
            System.out.print(arr[i] + "\t");
        } else {
            System.out.print("-\t");
        }

        power /= 10;
    }
    System.out.println();
}
```

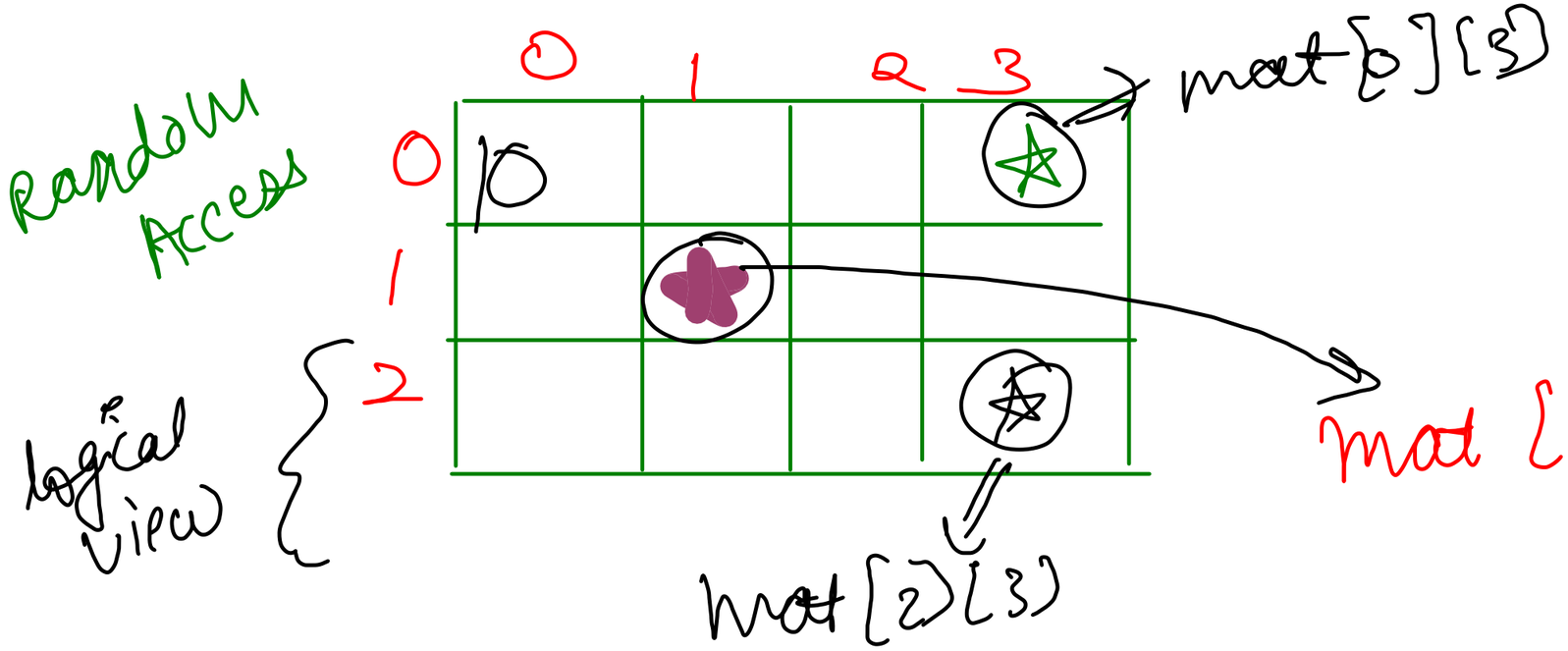
		10	20	30
0	000	—	—	—
1	001	—	—	30
2	010	—	20	—
3	011	—	20	30
4	100	10	—	—
5	101	10	—	30
6	110	10	20	—
7	111	10	20	30

2D arrays / matrix

row column

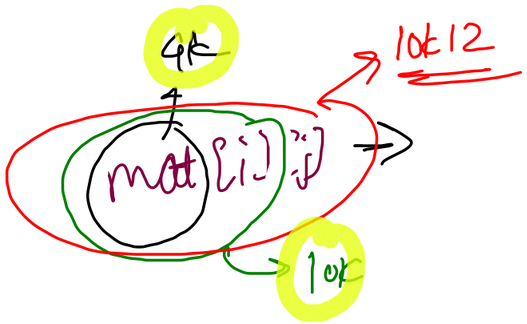
```
int [][] mat = new int [3][4];
```



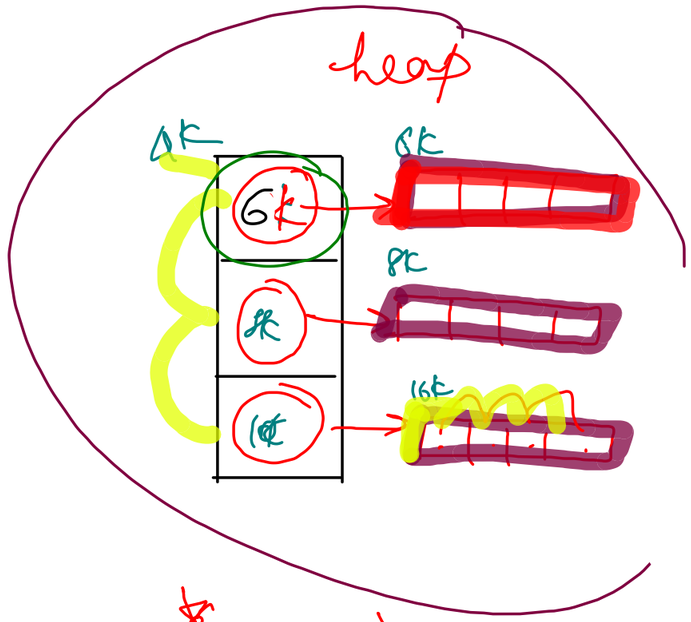


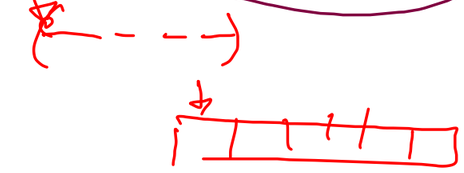
Memory Allocation

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

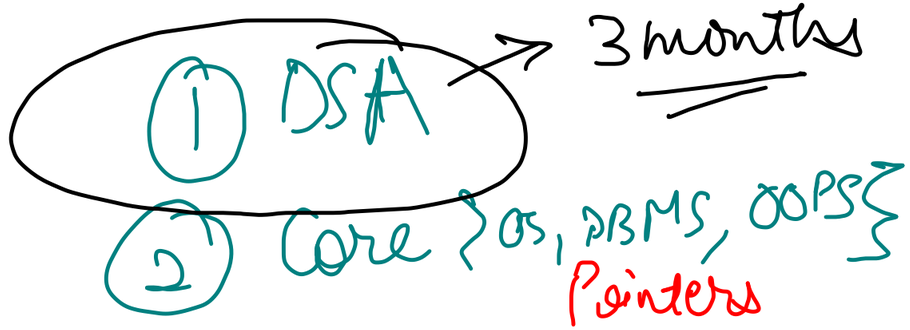



```java
int rows = scn.nextInt();
int col = scn.nextInt();

int[][] mat = new int[rows][col];

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

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

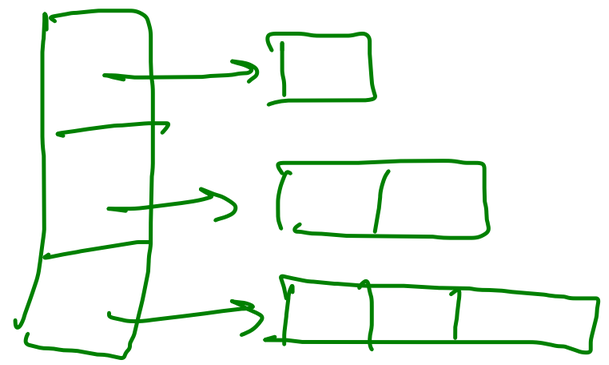




Stack & Queue



```
int [][] mat = new int[3];
for(int i=0; i<3; i++){
    mat[i] = new int[i+1];
}
```



Today's Questions

1. State of Wakanda - I — Wave Traversal {Zig-zag Order}
2. State of Wakanda - II — {Diagonal Order}
3. Spiral Display
4. Exit Point

Tomorrow's Questions

1. Matrix Multiplicatn
2. Rotate by 90°
3. Ring Rotate
4. Saddle Rice

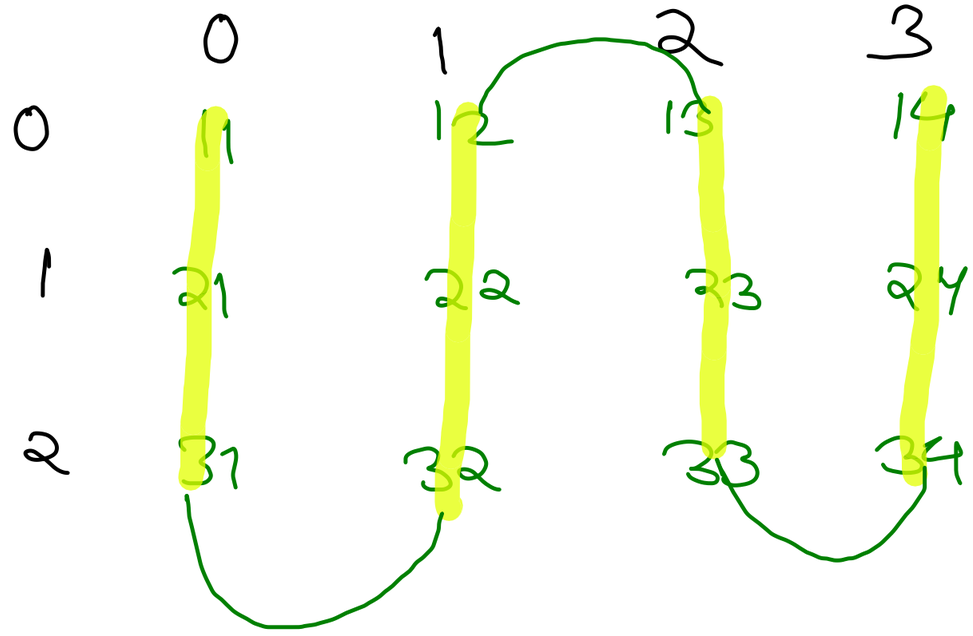
DSA

1. Level 1 :→ 3 months
2. Level 2 :→ 4 months
3. Level 3 :- 1 months

9 - 11:30

2.5 hr → M, T, W, F

State of Wakanda - 1

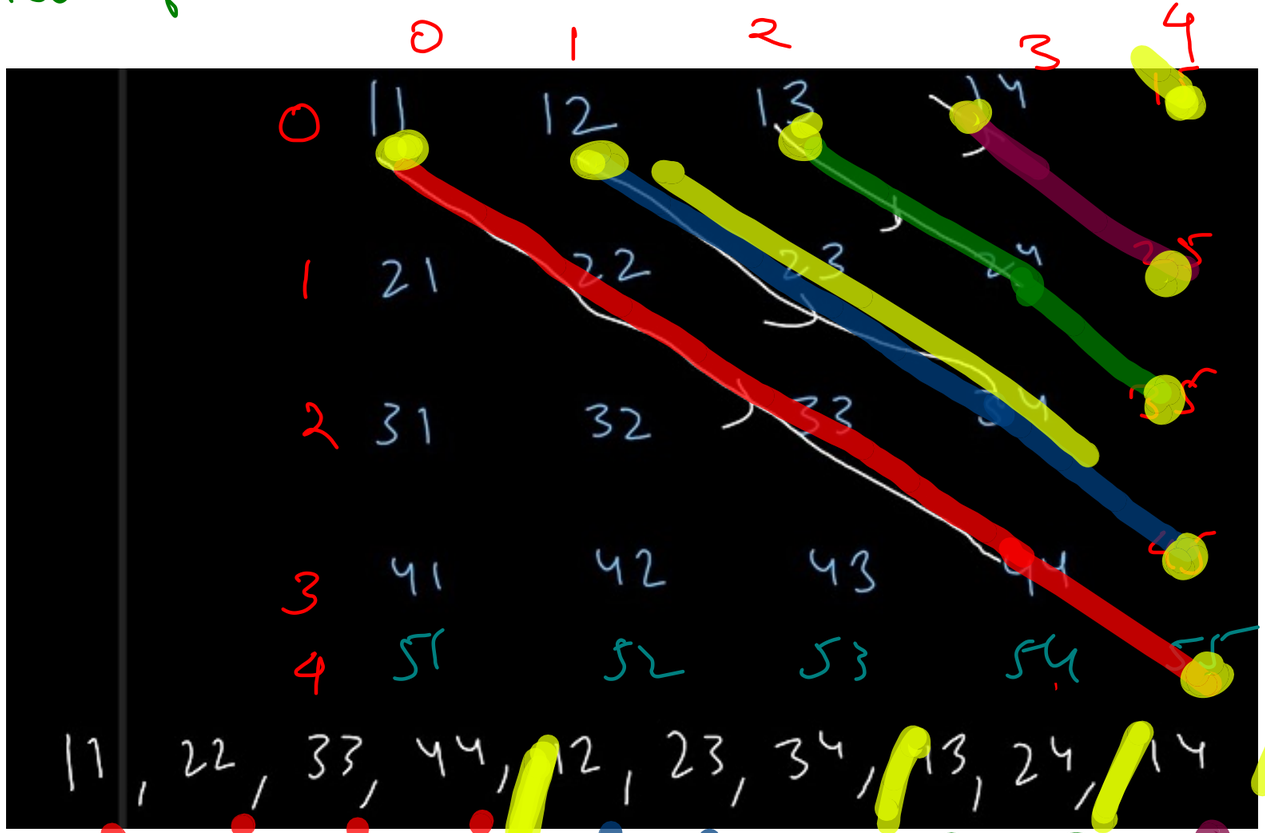


11 21 31 32 22 12

13 23 33 34 24 14

```java
public static void waveTraversal(int[][] mat){
    int n = mat.length; // rows
    int m = mat[0].length;

    // column by column print
    for(int j=0; j<m; j++){
        if(j % 2 == 0){
            for(int i=0; i<n; i++){
                System.out.println(mat[i][j]);
            }
        } else {
            for(int i=n-1; i>=0; i--){
                System.out.println(mat[i][j]);
            }
        }
    }
}
```

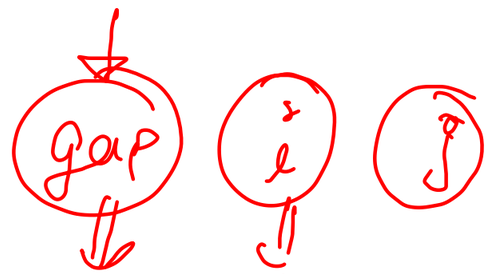
State of Wakanda - (11)



diagonals → for(gap=0; gap < m; gap++){ ← cols.

for(inner)
↳ i, i, i++j

5




```java
// outer loop -> diagonals -> gap
for(int gap = 0; gap < mat[0].length; gap++){

    // traversing the diagonal
    // gap -> j - i
    for(int i = 0, j = gap; j < mat[0].length; i++, j++){
        System.out.println(mat[i][j]);
    }
}
```

gap strategy =

Phase - I

Algo ✗

Code ✗

Phase - II

Algo ✓

Code ✗

Phase - III

Algo ✗

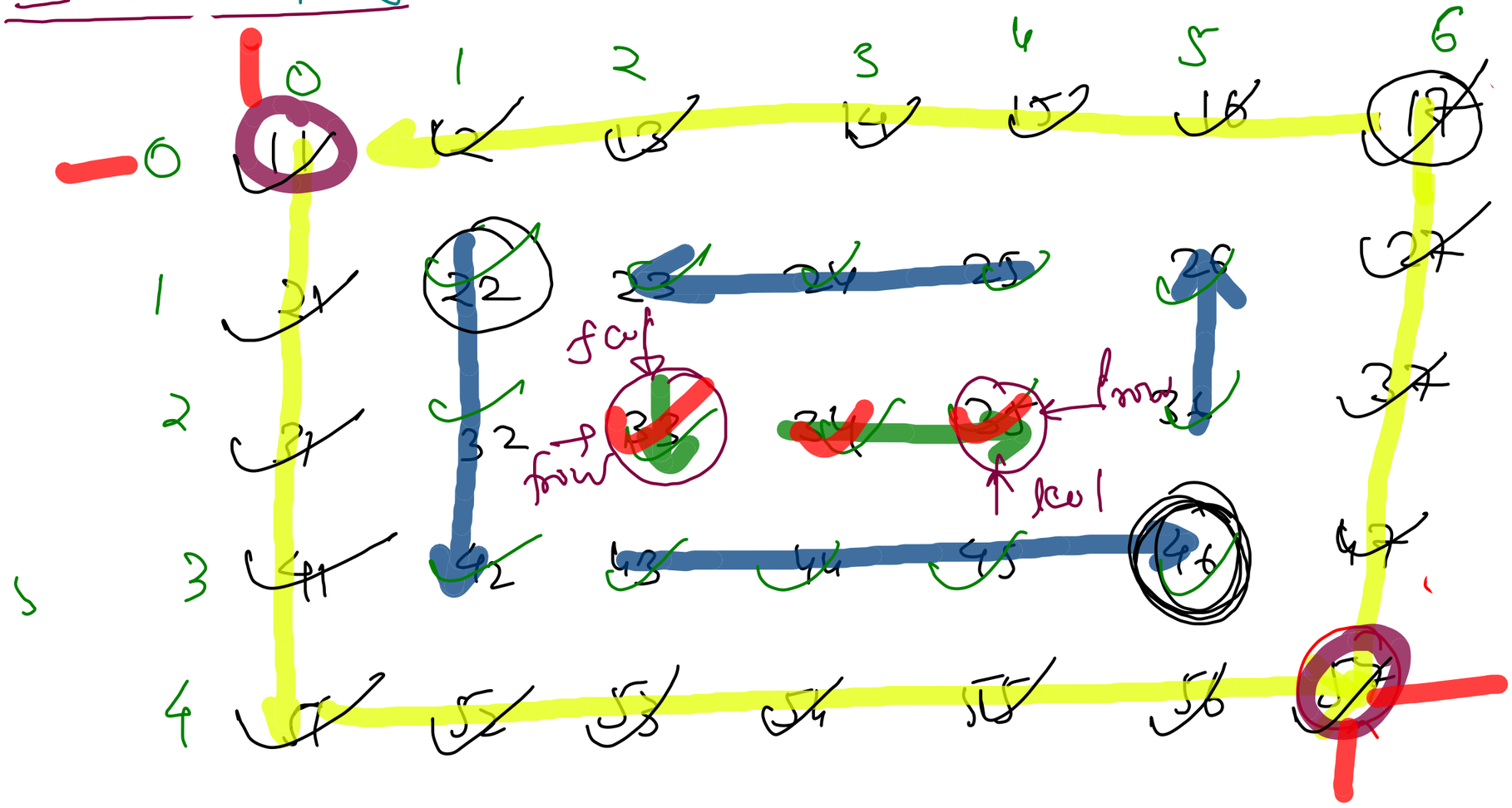
Code ✓

Phase IV

Algo - ✓

Code ✓

Spiral Display



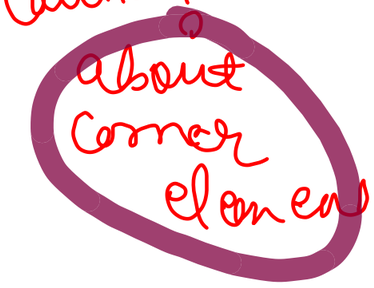
left wall → T to B

bottom wall → L to R

right wall → B to T

top wall → R to L

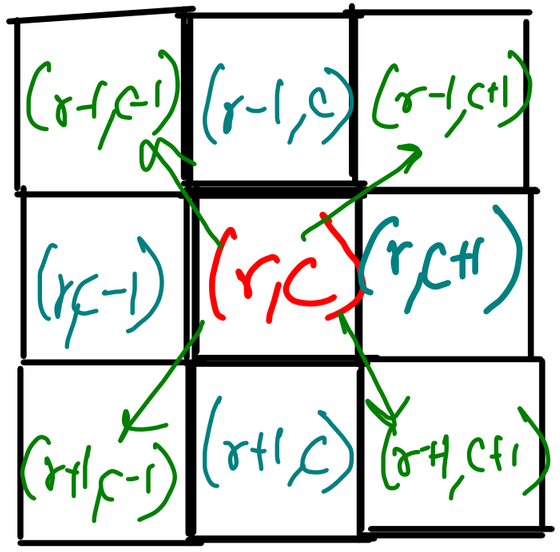
Caution!



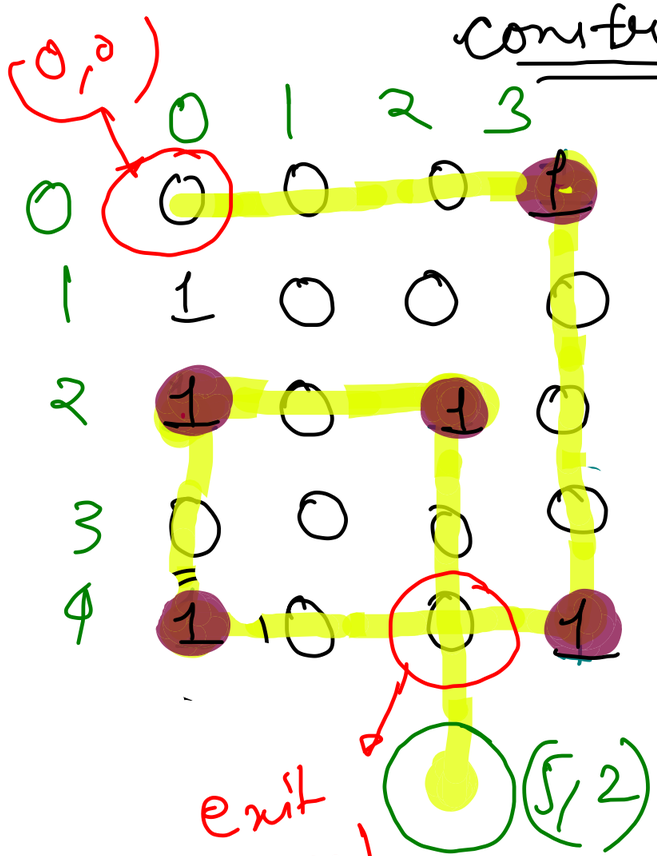
Exit Point

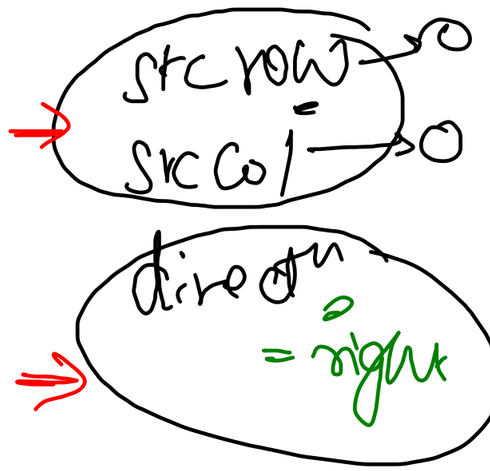
{ binary matrix }

there is always a exit Point

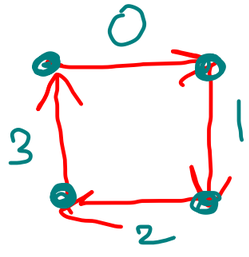


constructive algo





Samp direction



1 direction change { 90° clockwise }

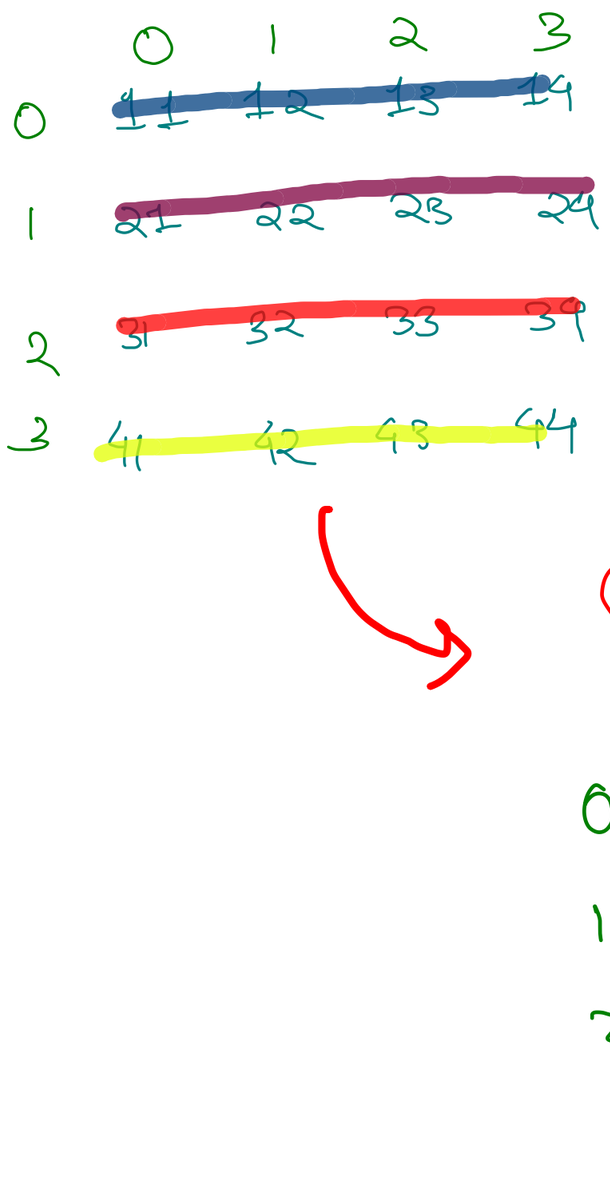
```java
int currRow = 0, currCol = 0;
int prevRow = 0, prevCol = 0;

// directions
// right -> 0, down -> 1, left -> 2, top -> 3
int direction = 0;

// left wall se exit, top wall se exit, bottom wall se exit, right wall se exit
while((currCol >= 0) && (currRow >= 0) && (currRow < n) && (currCol < m)) {

    if(mat[currRow][currCol] == 1){
        // change direction -> 90 degree clockwise
        direction = (direction + 1) % 4;
    }

    prevRow = currRow;
    prevCol = currCol;

    if(direction == 0){
        // right
        currCol++;
    } else if(direction == 1){
        // down
        currRow++;
    } else if(direction == 2){
        // left
        currCol--;
    } else {
        // top
        currRow--;
    }
}

System.out.println(prevRow);
System.out.println(prevCol);
```

Today's Questions

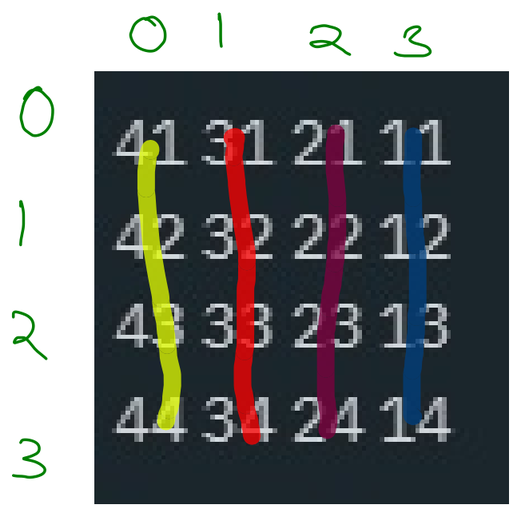
1. Matrix Multiplicatn
2. Rotate by 90°
3. Ring Rotate
4. Saddle Rice

Rotate by 90°

In-place



90° clockwise

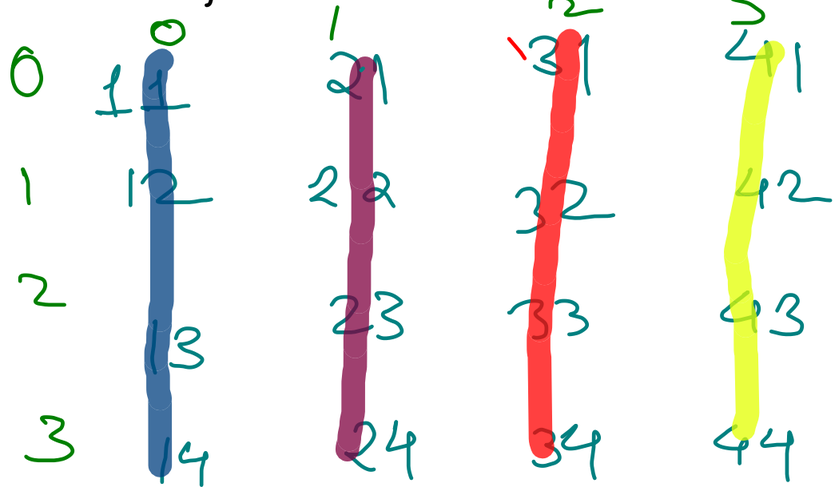



```java
public static void swapColumns(int[][] arr){

    int leftCol = 0, rightCol = arr[0].length - 1;
    while(leftCol < rightCol){

        for(int i=0; i<arr.length; i++){
            swap(arr, i, leftCol, rightCol);
        }
        leftCol++; rightCol--;
    }
}
```

↑ swap columns OR Reverse rows

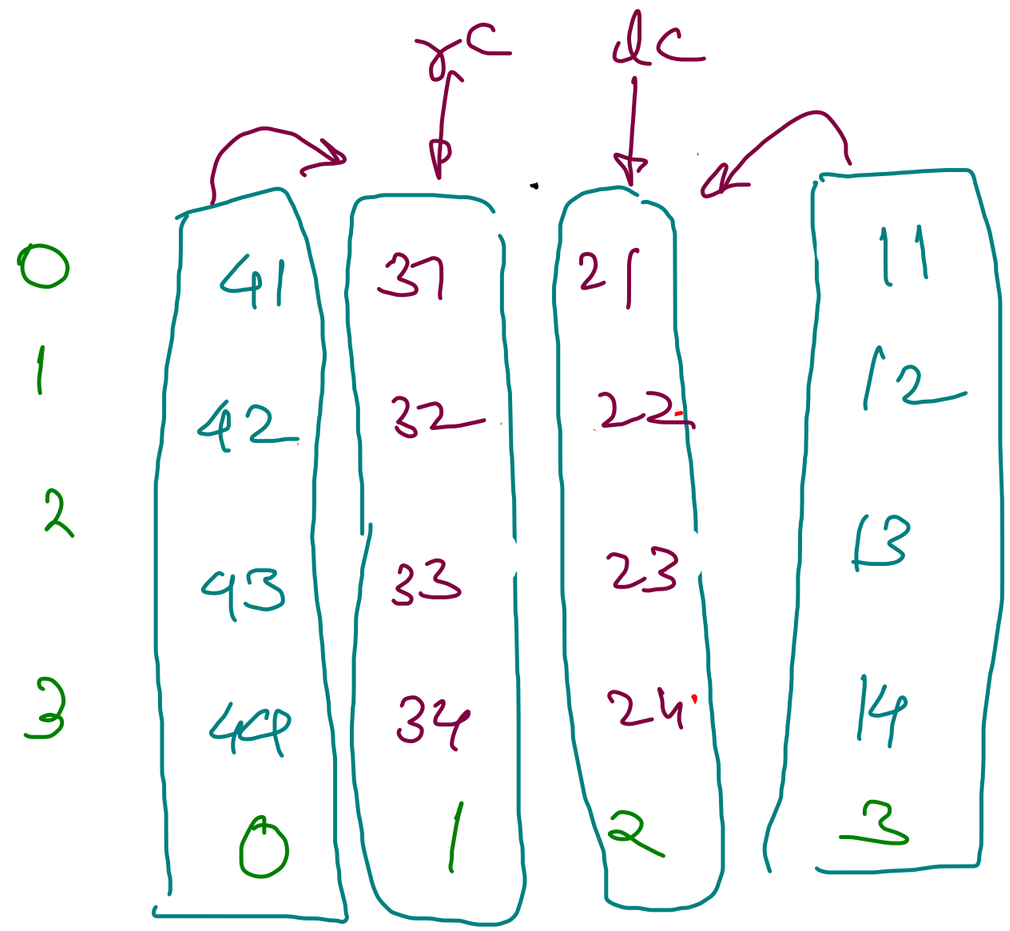
Transpose

a[i][j] <=> a[j][i]



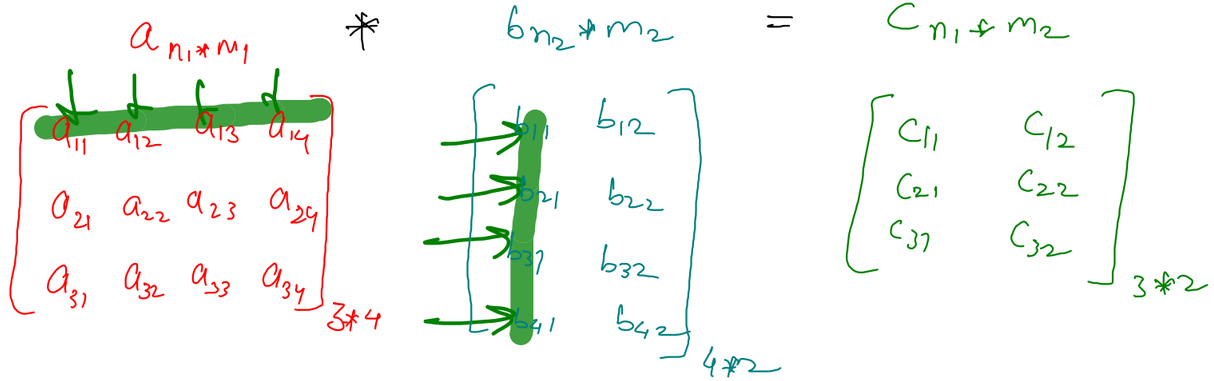

```java
public static void transpose(int[][] arr){
    for(int i=0; i<arr.length; i++){
        for(int j = i + 1; j < arr[0].length; j++){
            swap(arr, i, j);
        }
    }
}
```

```java
public static void swapColumns(int[][] arr){

    int leftCol = 0, rightCol = arr[0].length - 1;
    while(leftCol < rightCol){

        for(int i=0; i<arr.length; i++){
            swap(arr, i, leftCol, rightCol);
        }
        leftCol++; rightCol--;
    }
}
```



Matrix multiplication

$m_1 = n_2$




```java
int[][] res = new int[n1][m2];

for(int i=0; i<res.length; i++){

    for(int j=0; j<res[0].length; j++){
        // for calculating res[i][j]
        for(int k=0; k<m1; k++) { // m1 == n2
            res[i][j] += (mat1[i][k] * mat2[k][j]);
        }
    }
}
```

$$
\begin{cases}
c_{11} = a_{11} * b_{11} + a_{12} * b_{21} + a_{13} * b_{31} + a_{14} * b_{41} \\
c_{12} = a_{11} * b_{12} + a_{12} * b_{22} + a_{13} * b_{32} + a_{14} * b_{42} \\
c_{21} = a_{21} * b_{11} + a_{22} * b_{21} + a_{23} * b_{31} + a_{24} * b_{41} \\
c_{22} = a_{21} * b_{12} \quad a_{22} * b_{22} \quad a_{23} * b_{32} \quad a_{24} * b_{42}
\end{cases}
$$

$$c_{ij} = \sum_{k=0}^{m_1} a_{i,k} * b_{k,j}$$

```java
if(m1 != n2){
    System.out.println("Invalid input");
    return;
}
```

i=0 j=0 k=0/1/2/3

res[0][0]

$c_{11} = 0$

$c_{11} += a_{11} * b_{11}$

$+ a_{12} * b_{21}$

$+ a_{13} * b_{31}$

$+ a_{14} * b_{41}$

level (1)

breadth of knowledge

level (2)

depth of knowledge

level (3)

Problem solving

Saddle Price

	0	1	2 (minCol)	3
0	34	42	30	59
1 (maxRow)	50	85	44	97
2	42	16	25	36
3	15	21	05	50

All elements are unique/distinct

Saddle point-
- minimum in the row
- maximum in the column

44

(1) (0) saddle point (Y/N)

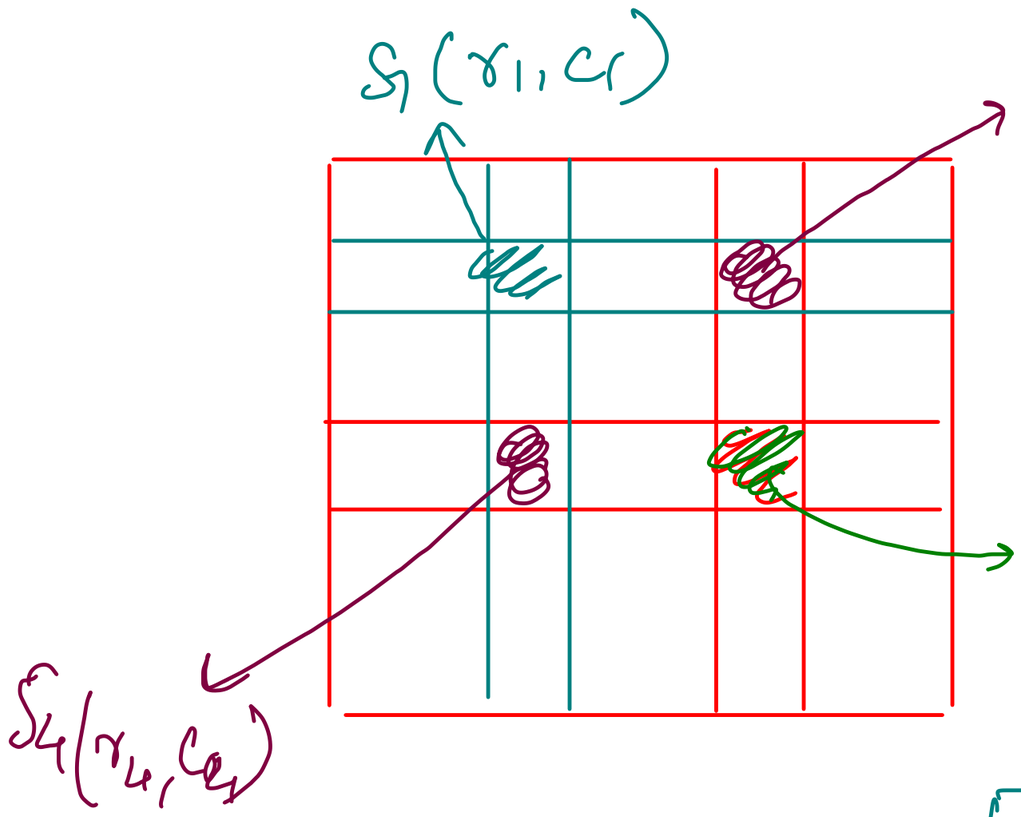
(2) >1 saddle point (Y/N)

Y ⇒ example

N ⇒ proof

```java
for(int i=0; i<mat.length; i++){

    // finding minimum of ith row
    int minCol = 0;
    for(int j=0; j<mat[0].length; j++){
        if(mat[i][j] < mat[i][minCol]){
            minCol = j;
        }
    }

    int maxRow = 0;
    for(int k=0; k<mat.length; k++){
        if(mat[k][minCol] > mat[maxRow][minCol]){
            maxRow = k;
        }
    }

    if(maxRow == i){ // min in Row -> maxinCol is also true
        System.out.println(mat[maxRow][minCol]);
        return;
    }
}
```

Assumption : ② saddle point S_1 S_2



① $S_1 < S_3$

② $S_1 > S_4$

} S_1

① $S_2 < S_4$

④ $S_2 > S_3$

} S_2

$S_1 < S_3 < S_2 \Rightarrow S_1 < S_2$

$S_1 > S_4 > S_2 \Rightarrow S_1 > S_2$

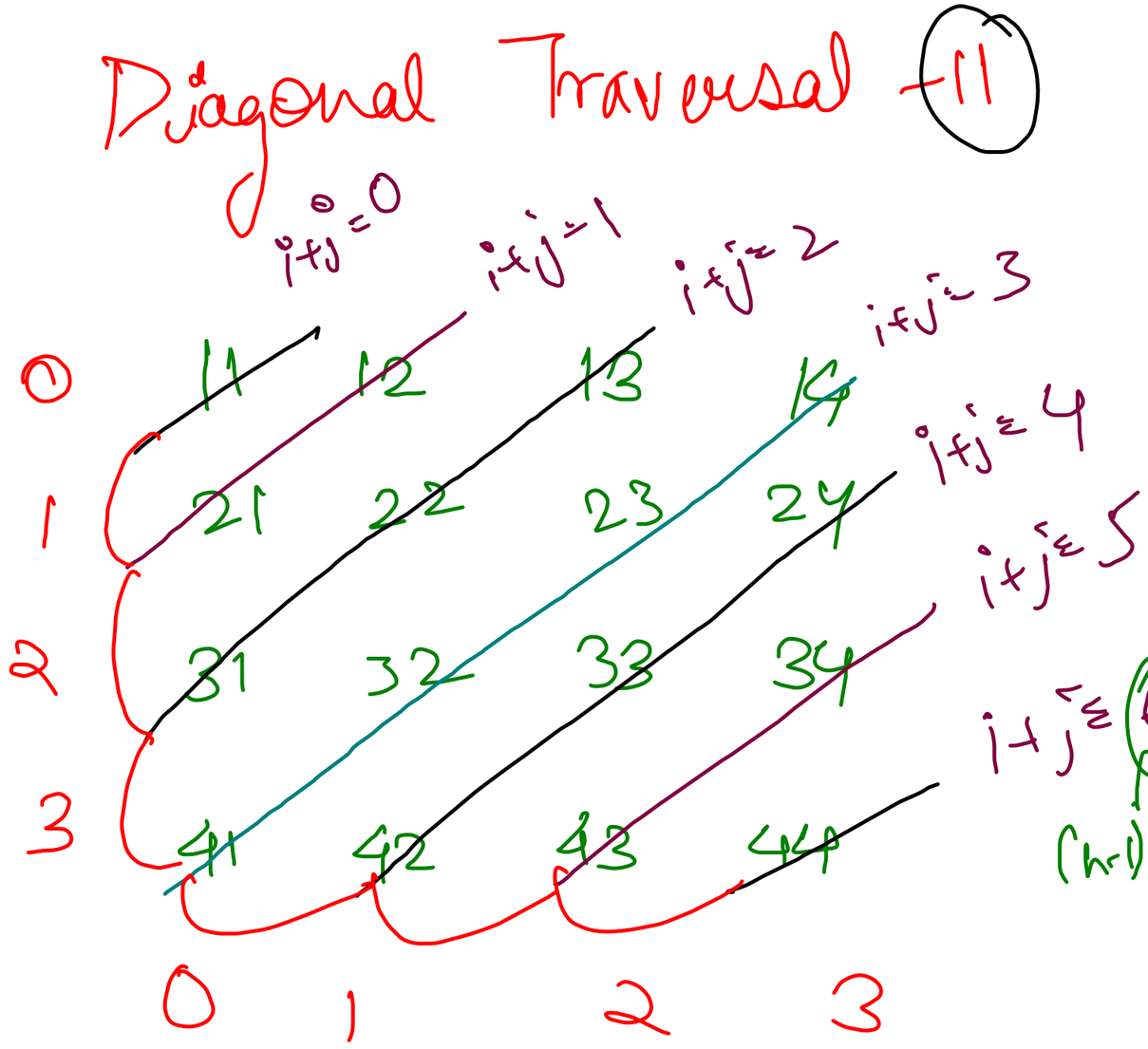
→ assumption wrong

There will be atmost 1 saddle point.

Maths Required for DSA

1. Permutations & Combinations
2. Sequence & Series → AP, GP
3. Number Theory → Prime Nos, GCD, LCM, factorial
4. logarithm
5. Sets, Relat & fun

Diagonal Traversal - II




```
for(int sum = 0; sum <= (n -1) + (m-1); sum++){
    int i = 0, j = 0;
    if(sum <= (n - 1 + m - 1) / 2){
        // lower triangle
        j = 0;
        i = sum - j;
    } else {
        i = n - 1;
        j = sum - i;
    }

    for(; i >= 0 && j < m; i--, j++){
        System.out.print(mat[i][j] + " ");
    }
    System.out.println();
}
```

Strings → Array of characters

- lowercase { 'a', 'b', 'c', 'd' ... }
- uppercase { 'A', 'B', 'C', ... }
- digits { '0', '1', '2', '3' ... }
- null character { '\0' };

Unicode ⇄ 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]

length()

{ 'h', 'e', 'l', 'l', 'o' }
0, 1, 2, 3, 4

"hello"

65 - 90 (A - Z)

97 - 122 (a - z)

48 - 57 (0 - 9)

A → a

a → A

b → B

B → b

'A' - 'a' = -32

'a' - 'A' = 32

pepCODinG

'p' - 'p' = 'a' - 'A'

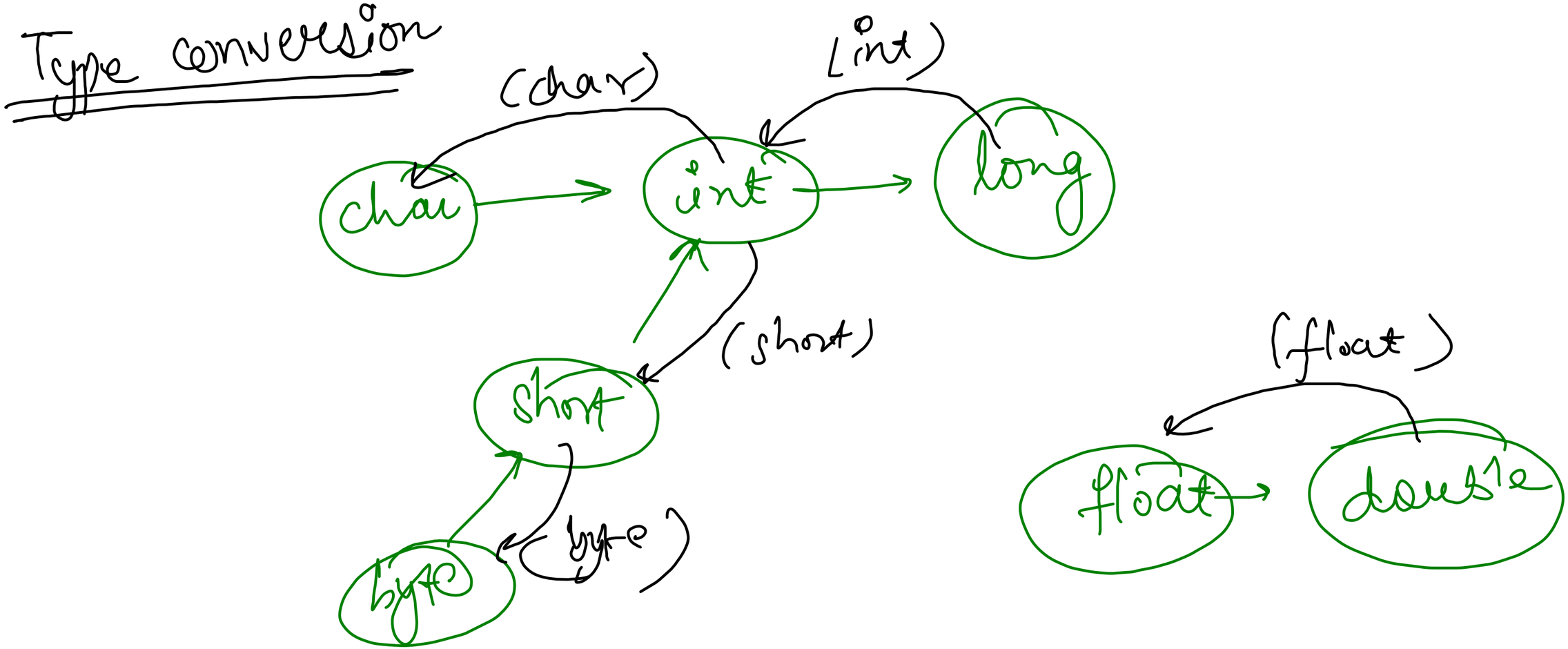
'p' = 'p' - 'a' + 'A'

'p' = 'p' - 97 + 65

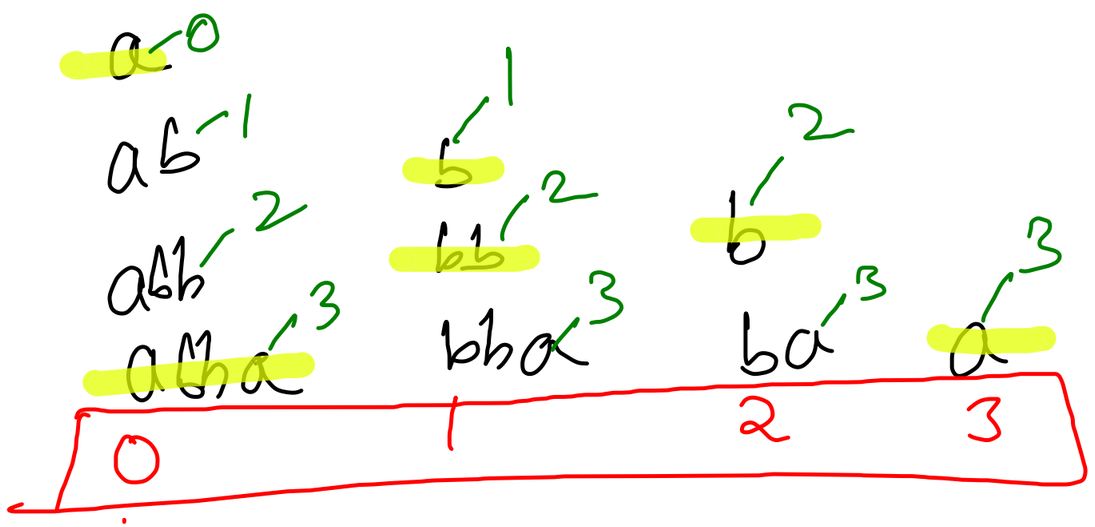
'p' = 'p' - 32

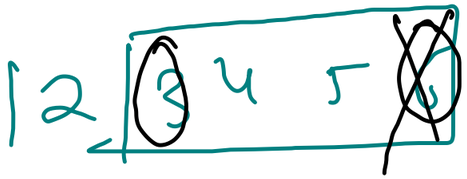
```
for(int i=0; i<?; i++){
  if(ifLowerCase)
    'a' <=> 'z'
    [97, 122]
    toUpperCase()
  else
    toLowerCase()
}
```

Type conversion



"abba"
0 1 2 3





[l, r)
r - l

Str.substring(startingIndex, endingIndex)

Strings

Memory Mapping

① String str = "peprodev"; ← string literal

② String str2 = new String("hello");

③ str3 = "hello"

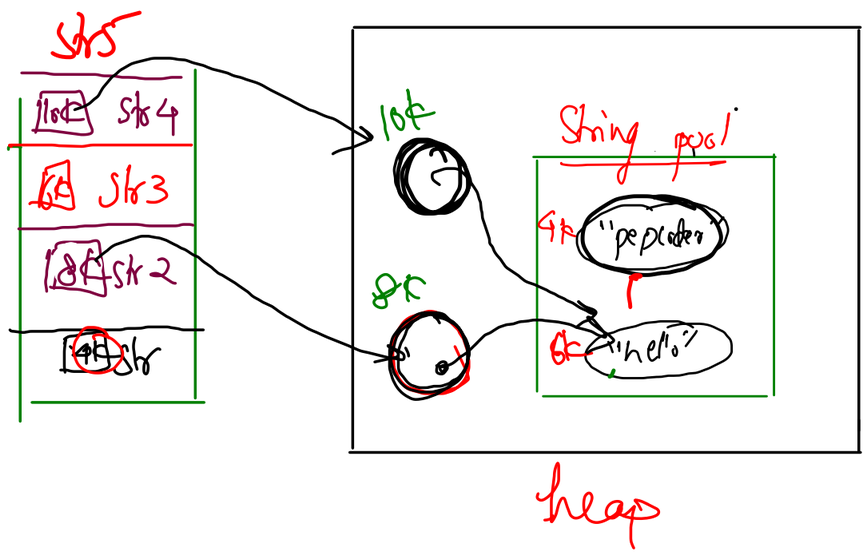
④ str4 = new String("hello");

⑤ str5 = "hello";

① length()

② substring()

③ charAt()



==
① str2 == str3 (F)
② str3 == str5 (T)
③ str2 == str4 (F)

④ str2.equals(str3) (T)
⑤ str3.equals(str5) (T)
⑥ str2.equals(str4) (T)

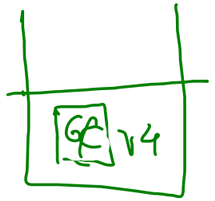
⑦ str == str2 (F)
⑧ str.equals(str2) (F)

Togglecase → ASCII code

String Interning → Advantage ⇒ Memory wastage
→ Implicit

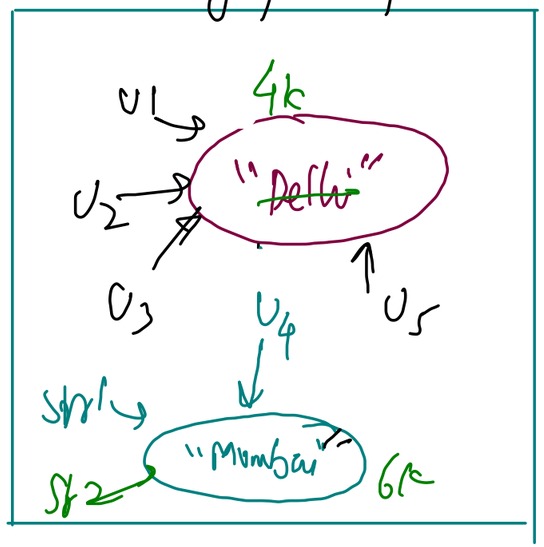
String str = "Mumbai";

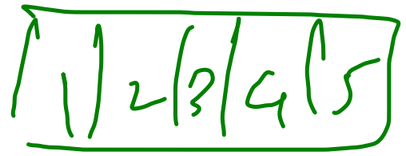
String str2 = "Mumbai";



① v4 = "Delhi"

v4 = "Mumbai"





arr[2] = 30

String.charAt(2) = 'A'

setchar ✗

Implications

address comparison

difference b/w (==) && equals()

int val = 10

if (val == 10)

(1) Address equal : true

(2) Content equal : true

(3) F,F : false

=

Immutability

"Java Strings are immutable"

"References are mutable

but instance/object are not

String S = "0";

for (int i = 1; i <= n; i++) {

S = S + i;

Concatenate

1 + 2 + 3 + ... 1000

$= \frac{1000 * 1001}{2} = \frac{n * n+1}{2}$

$O(n^2)$

"0123456 ... 1000"

995

1000 operations

01234

0123

012

012345

0123

0123

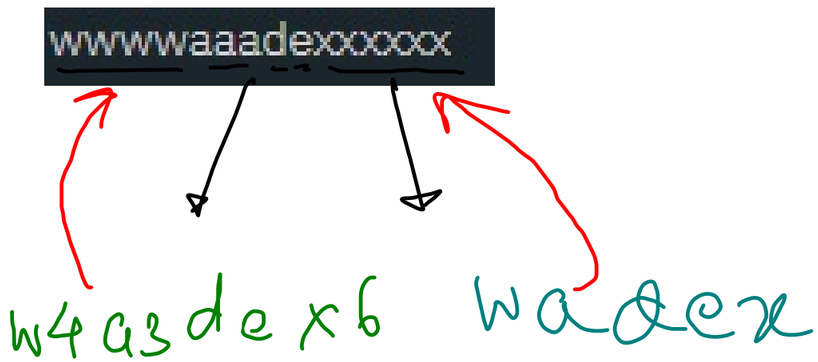
0123

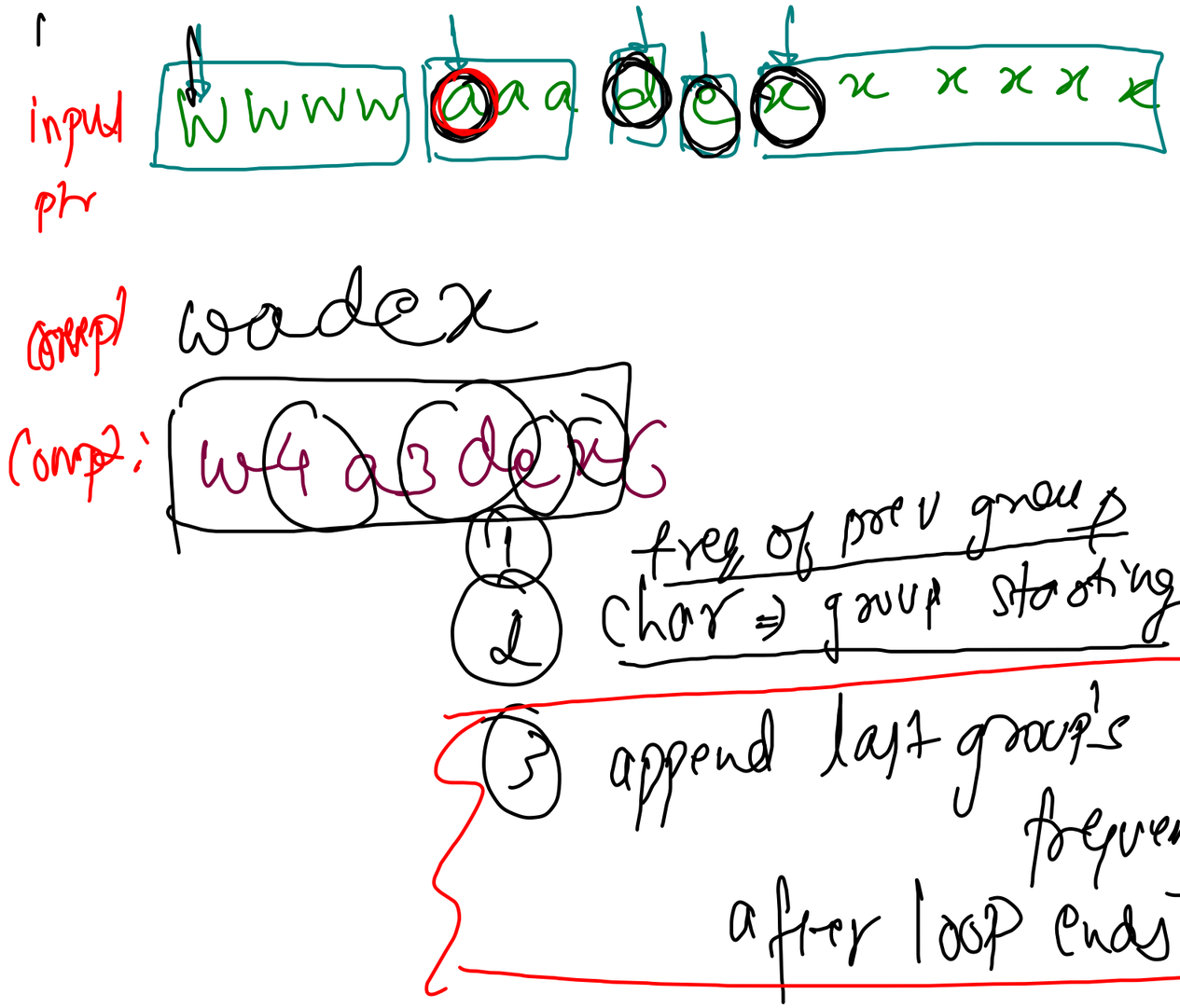
Distract → Social Media

Nodes

2 College

String Compression

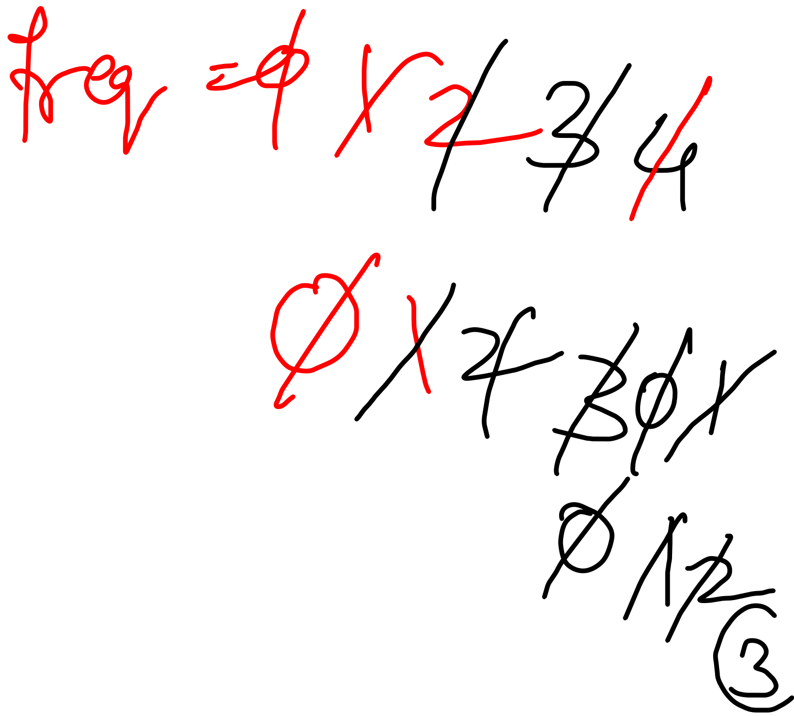




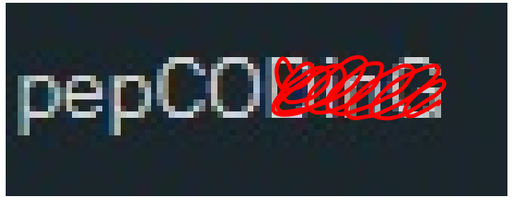
```java
String output = "";
int freq = 0;

for(int idx=0; idx<input.length(); idx++){
    // is character first element of it's group
    if(idx == 0 || input.charAt(idx) != input.charAt(idx - 1)){
        if(idx > 0 && freq > 1){
            output = output + freq;
        }
        output = output + input.charAt(idx);
        freq = 0;
    }
    freq++;
}

if(freq > 1){
    output = output + freq;
}
return output;
```

Input: wwww aaa de xxx

output: w4a3dex3



String With Difference Of Every Two Consecutive Characters



p -11 e 11 p -45 C 12 O

Topics to be covered today

hello

charAt

1. StringBuilder
   - initialization & declaration
   - Insert { at last, at given index }
   - Update { setCharAt }
   - Delete { deleteCharAt }

2. ArrayList
   - size()
   - C
   - R
   - U
   - D → remove()
   - C → Declaration & initialization
   - R → get
   - U → Set

(Q) Remove Primes

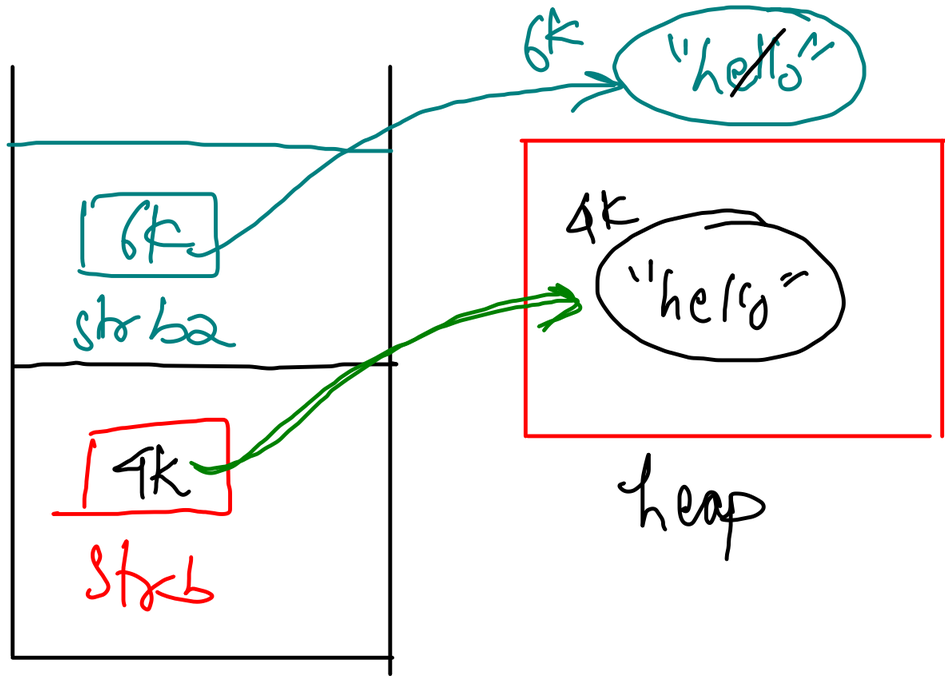
{HW} (Q) Ring Rotate

Discussion

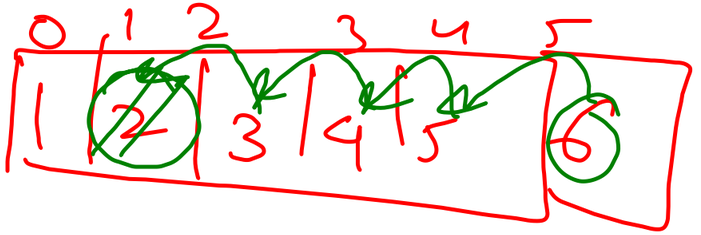
StringBuilder ⟹ String
StringBuilder ⟹ mutable : → References can be changed
Object can be changed
mutable → C++ strings
mutable → Interning

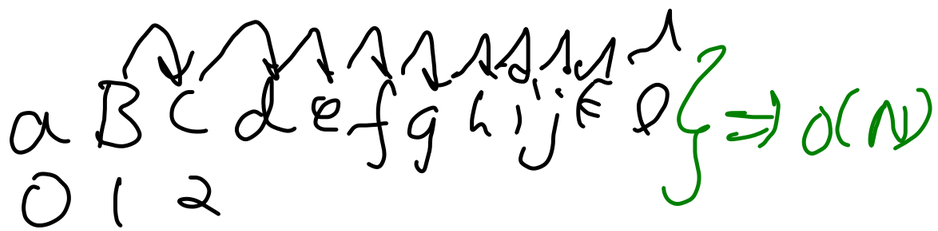
StringBuilder strb = new SB("hello");

StringBuilder strb2 = new SB("[illegible]");

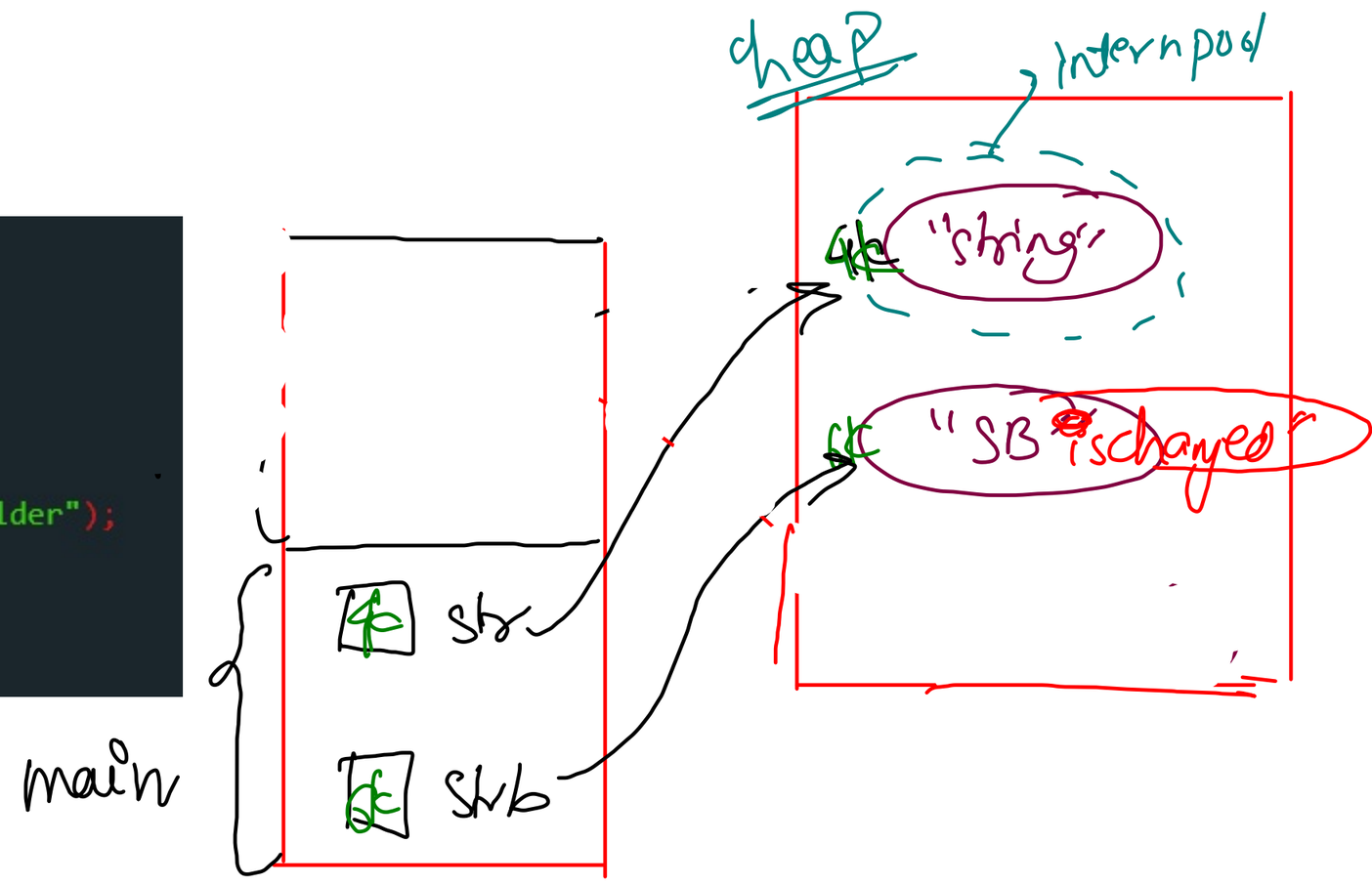


```java
// Update -> Constant Operation
strb.setCharAt(3, 'd');
System.out.println(strb);

// Delete
strb.deleteCharAt(1);
System.out.println(strb);

// Insert
strb.insert(2, 'e');
System.out.println(strb);

// Append (Insert at Last) -> Constant Operation
strb.append('s');
System.out.println(strb);
```





```java
public static void fun(String x, StringBuilder y){
    x = x + " is changed";
    y.append(" is changed");
}

public static void main(String[] args) {
    String str = "string";
    StringBuilder strb = new StringBuilder("stringbuilder");
    fun(str, strb);
    System.out.println(str);
    System.out.println(strb);
}
```



ArrayList

① dynamic memory

② generic

Array int [] arr = new int[n];

static / fixed

10000 1000th

Size

Shrink {Memory wastage ↓}

grow {dynamic}

8 primitive datatypes

stack

heap

0

'\0' null character

0.0

0.0

false

0

int → Integer

char → Character

float → Float

double → Double

boolean → Boolean

long → Long

short → Short

byte → Byte

parseInt

MAX VALUE

MIN-VALUE

Wrapper Classes

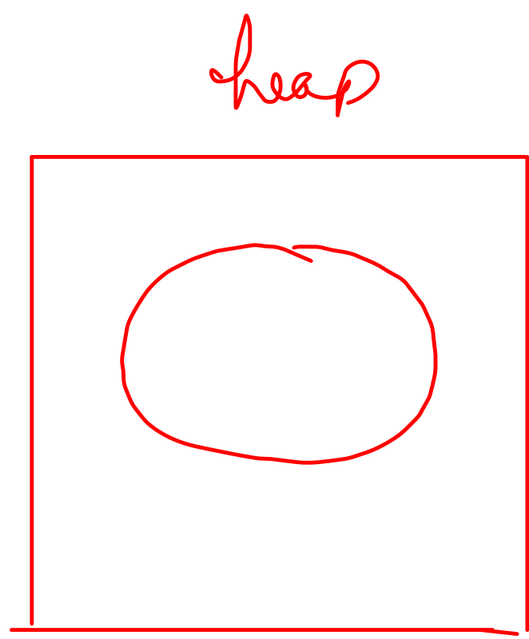
Integer i;

if not initialised
null in reference

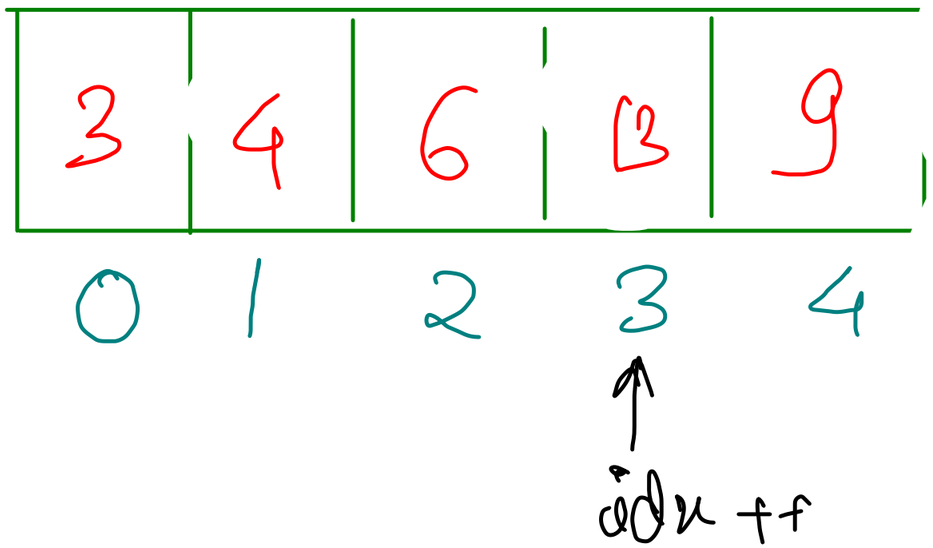
ArrayList<Integer> arr = new ArrayList<>();

arr.add(1)

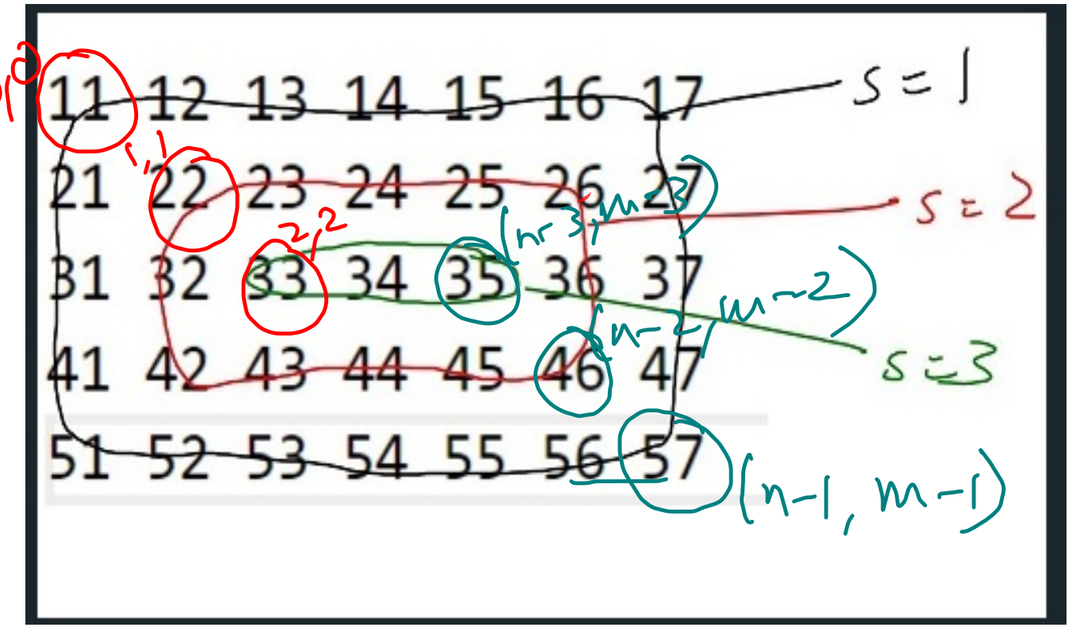
arr.add(2)



Remove Primes



Ring Rotate



1. fill shell in 1D array
2. Rotate 1D array { Reversal Algo }
3. Fill Rotated array back into the shell of matrix

shell = (2) k = (2)

s^{th} shell
- top left : (s-1, s-1)
- bottom right : (n-s, m-s)

Strings {Contest → 05}

Challenges

Pangrams

Success Rate: **90.24%** Max Score: **10** Difficulty: **Easy**

Solve Challenge

Maximum Frequency Character

Success Rate: **89.13%** Max Score: **10** Difficulty: **Easy**

Solve Challenge

Funny String

Success Rate: **89.74%** Max Score: **10** Difficulty: **Easy**

Solve Challenge

Remove Consecutive Duplicates

Success Rate: **97.73%** Max Score: **15** Difficulty: **Easy**

Solve Challenge

CamelCase

Success Rate: **100.00%** Max Score: **15** Difficulty: **Easy**

Solve Challenge

Gemstones

Success Rate: **74.07%** Max Score: **20** Difficulty: **Easy**

Solve Challenge

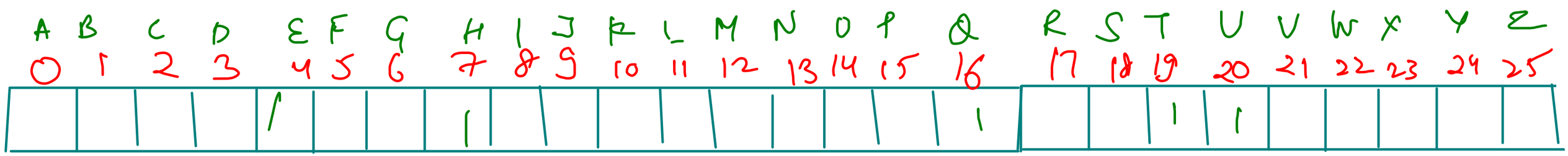
Game of Thrones - I

Success Rate: **88.24%** Max Score: **20** Difficulty: **Easy**

Solve Challenge

(Q1) PANGRAM

s = 'The quick brown fox jumps over the lazy dog'

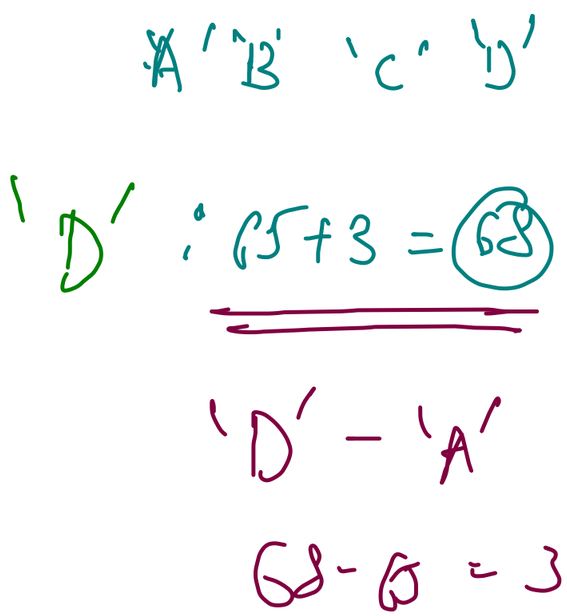


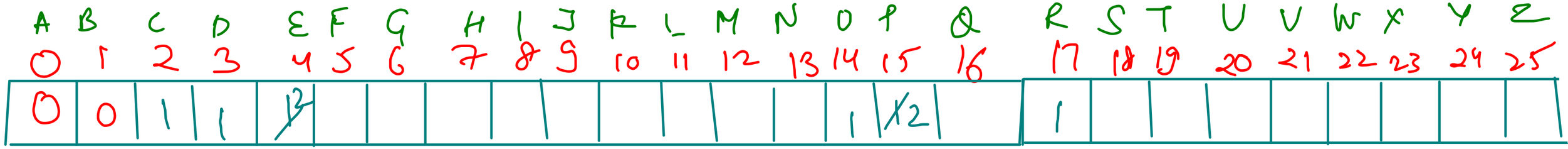
① Freq char : 26

② Iterate on input & update freq

③ Iterate on freq array & check if any cell is 0, then false else true

```java
public static String pangrams(String s) {
    int[] freq = new int[26];

    // Fill Frequency Array
    for(int i=0; i<s.length(); i++){
        if(s.charAt(i) >= 'A' && s.charAt(i) <= 'Z'){
            freq[s.charAt(i) - 'A']++;
        } else if(s.charAt(i) >= 'a' && s.charAt(i) <= 'z'){
            freq[s.charAt(i) - 'a']++;
        }
    }

    // Check if string is pangram: Iterating on frequency array
    for(int i=0; i<26; i++){
        if(freq[i] == 0){
            return "not pangram";
        }
    }
    return "pangram";
}
```






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

    int[] freq = new int[26];

    for(int i=0; i<s.length(); i++){
        freq[s.charAt(i) - 'a']++;
    }

    int maxFreq = 0;
    char maxFreqChar = 'a';
    for(int i=0; i<26; i++){
        if(freq[i] > maxFreq){
            maxFreqChar = (char)('a' + i);
            maxFreq = freq[i];
        }
    }

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

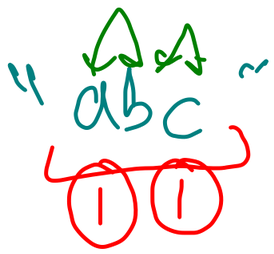
(Q2) Maximum Frequency char

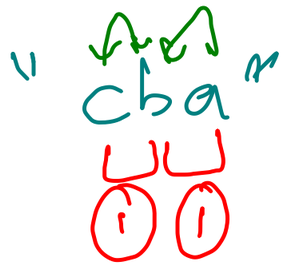
"pepcoder"

maxFreq = ~~'c'~~ 'e'

~~1~~ 2

(Q3) Funny String






```java
public static String funnyString(String s) {
    // Write your code here
    StringBuilder rev = new StringBuilder(s);
    rev.reverse();

    for(int i=0; i<s.length()-1; i++){
        int val1 = Math.abs(s.charAt(i + 1) - s.charAt(i));
        int val2 = Math.abs(rev.charAt(i + 1) - rev.charAt(i));

        if(val1 != val2) return "Not Funny";
    }

    return "Funny";
}
```

04) Remove Consecutive Duplicates

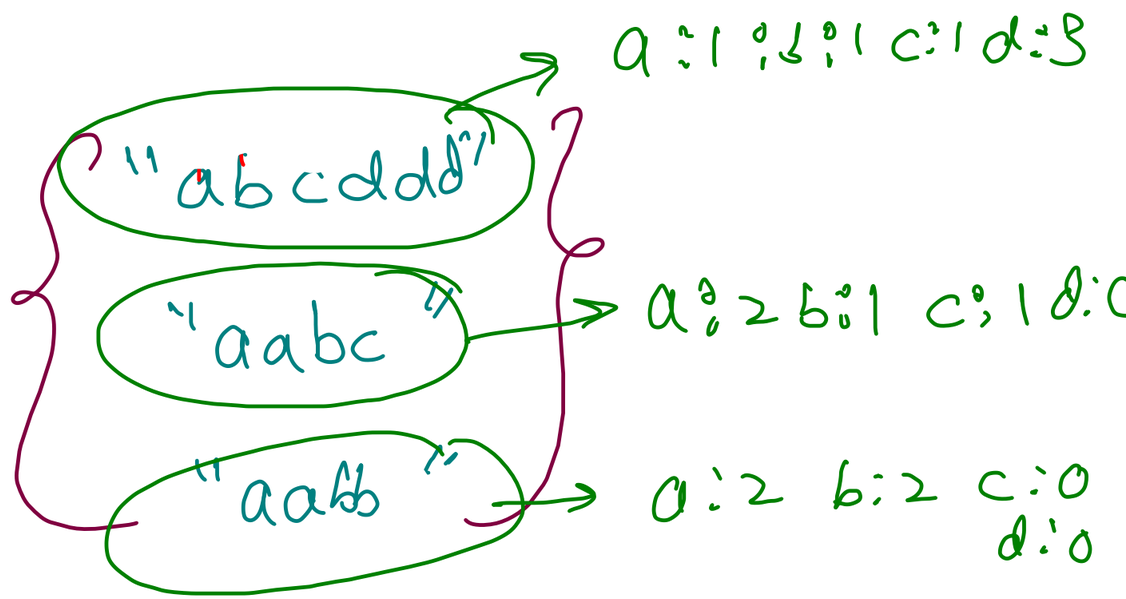
aabbbccccccc

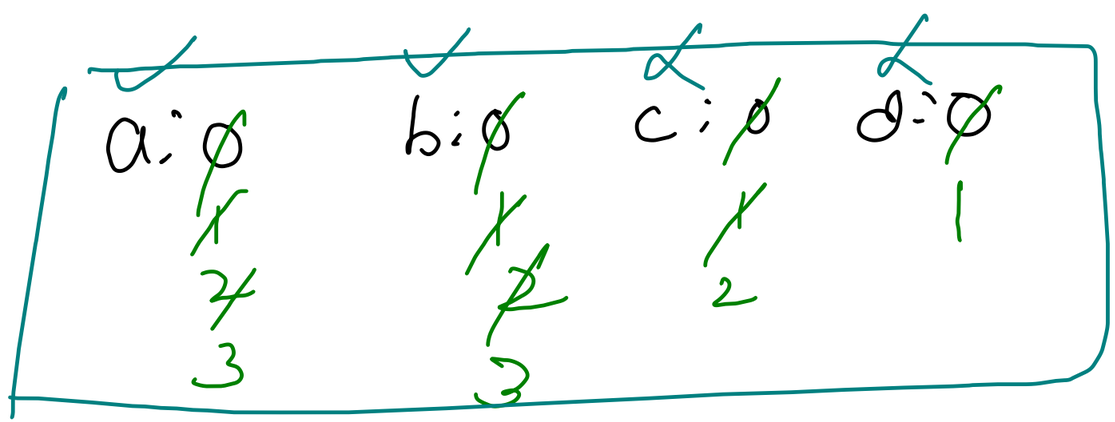
{ String Compression -I }

```
public static int camelcase(String s) {
// Write your code here
    int count = 0;
    for(int i=0; i<s.length(); i++){
        if(i == 0 || (s.charAt(i) >= 'A' && s.charAt(i) <= 'Z')){
            count++;
        }
    }
    return count;

}
```

(05) Gemstones

```
// Frequency Array of Minerals -> LowerCase Alphabets
int[] freq = new int[26];

// Loop on Array of Strings
for(int i=0; i<arr.size(); i++){
    // Loop on Character of String
    int[] tempFreq = new int[26];
    for(int j=0; j<arr.get(i).length(); j++){
        char ch = arr.get(i).charAt(j);
        tempFreq[ch - 'a']++;
    }

    for(int j=0; j<26; j++){
        if(tempFreq[j] > 0){
            freq[j]++;
        }
    }
}

int countOfGemStones = 0;
for(int i=0; i<26; i++){
 // is Mineral present in all the stones
    if(freq[i] == arr.size()){
        countOfGemStones++;
    }
}
```





(QC) Game of Thrones {Palindromic Permutation}

"aabbaadd"

a : 4

b : 2

d : 2

even

if all characters have even freq

000...000 | 000

abad | daba

"abbcddabcdddadb"

a:3 b:4 c:2 d:4

abbcddadcbba

odd length

- Only one character will have odd frequency
- remaining should have even

```java
public static String gameOfThrones(String s) {
// Write your code here
    int[] freq = new int[26];
    for(char ch: s.toCharArray()){
        freq[ch - 'a']++;
    }

    int oddCount = 0;
    for(int i=0; i<26; i++){
        if(freq[i] % 2 == 1){
            oddCount++;
        }
    }

    if(oddCount <= 1) return "YES";
    else return "NO";
}
```

1. R&B
2. S&S
   - Time Comp
   - Space Comp
3. LL
4. OOPS
5. S&O