

DSA

A way start
from zero.

$\begin{array}{c} 0 \ 1 \ 2 \\ \text{int num[]} = \{2, 3, 4\}; \end{array}$

sysout num[i]);

Output

⇒ 3

array name [on
which
place];

Arrays ↳ A type of list to store data starting with 0, 1, 2...

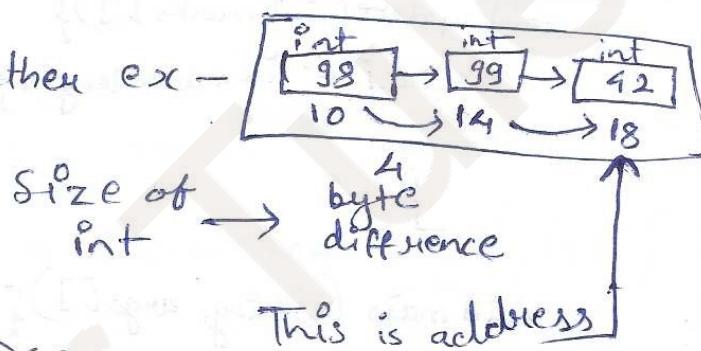
↳ In Java and C++ indexing starts with 0 always.

Definition:-

A List of elements of the same type placed in contiguous memory location.

↳ contiguous → one after another ex -

* It is a variable which have so many sub-variables



Operations on Array

To see the length of array

`System.out.println(arrayName.length);`

1) Creating an Array

`dataType arrayName[] = new dataType[size];`

Ex -

`int marks[] = new int[50];` → 50 integers

`int number[] = {1, 2, 3};` → 3 integers

`int num[] = {4, 5, 6};` → 3 integers

`String fruits[] = {"apple", "mango"};` → 2 strings.

If no elements are stored then it automatically stores zero.

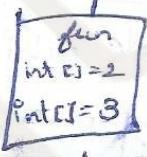
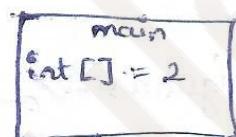
Passing array as an Argument

Important

- ↳ If we made an array in main function And used it in another small function and update it there. then its value also changes in main function.

→ class
void update(int marks[]){
 for(int i=0; i<mark.length; i++){
 []
 }
}

void main (String args[]){
 int marks [] = ~~new~~ {21, 98, 99};
}



Now it will also change in main function.

Time Complexity →

time taken by the algorithm to execute each set of instruction.

Space Complexity →

It is the amount of space or memory taken by an algorithm to run as a function of the length of the input.

- * They both are taken as a measures to choose the best algorithm to solve a particular problem

Linear Search → It's a type of search to search a element in an array.

ex -

0	1	2	3	4	5	6
2	4	6	8	10	12	14

find key = 10

∴ Key = 10 is at index 4

That means 10 got room no. 4 in that array.

Code :

```
→ a function -( numbers[], key) {  
    for( int i=0; i < numbers.length; i++) {  
        if( key == number[i] ) { return i; }  
    }  
    return -1;  
  
main( String args[] ) {  
    int numbers [] = { 2, 4, 6, 8, 10, 12, 14, 16 };  
    int key = 1; // calling the function  
    int index = linearSearch linearSearch(numbers, key);  
    if( index == -1 ) { System.out.println("Not found"); }  
    else { System.out.println("The key is at " + index); }
```

to get import java.util.*;

- 1) -∞ (Smallest no.) → Integer.MIN_VALUE
- 2) +∞ (Largest no.) → Integer.MAX_VALUE

Largest Number in an Array

or Smallest

→ Compare each no. Pseudo Code

1) First make a integer largest and give value - ∞

2) then in a for loop add a ~~if else~~

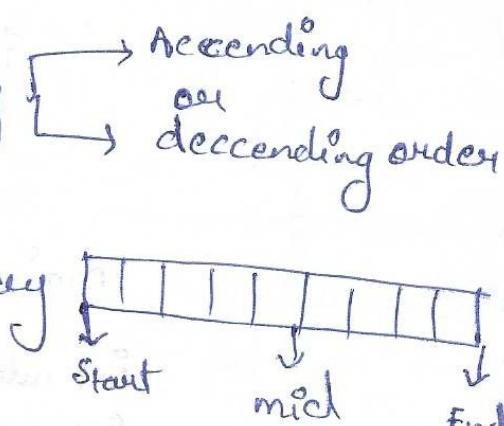
and compare each ~~no~~ number

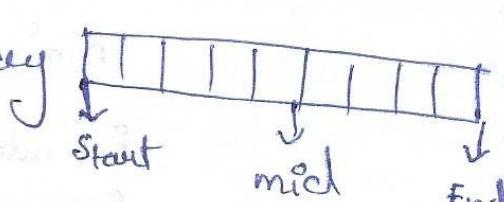
and then change the largest int to the largest no. in array.

Binary Search

* Asked in interviews and coding tests.

↳ It's a search that we do to search words in dictionary.

↳ Prerequisite → Sorted Array  Ascending or decending order

→ If we have a long array 

i) Find start and end of array.
ii) then find Mid.

iii) If required key is greater than mid

key < Mid \rightarrow take LHS Array
key > Mid \rightarrow take RHS Array

take LHS Array

If S → is same

End → Mid - 1

and Find the new mid

and then same process repeat

If End is same

Start = Mid + 1

Find the new mid

and then same

i) ii) iii) repeat

Pseudo Code

$\rightarrow \text{start} = 0, \text{end} = n-1$

1) while ($\text{start} \leq \text{end}$)

2) find mid $\Rightarrow \boxed{[]}$

3) Compare mid & key

4) $\text{mid} == \text{key} \rightarrow \text{Found}$

5) $\text{mid} < \text{key} \rightarrow \text{take}$

6) $\text{mid} > \text{key} \rightarrow \text{take Left side array}$

update

end

start

* Important

* While writing array in the parameter write it with $[]$

e.g - public static int binarySearch(int num[], key){}

① Time Complexity of Binary Search

$$\begin{array}{l} \text{Iteration 1} \rightarrow n \\ 2 \rightarrow n/2 \\ 3 \rightarrow n/4 \end{array} \left. \begin{array}{l} \rightarrow \frac{n}{2^0} \\ \rightarrow \frac{n}{2^1} \\ \rightarrow \frac{n}{2^2} \end{array} \right\}$$

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$$

$$k = \log_2 n$$

TC $\propto \log_2 n$

$O(\log n)$

Time Complexity

L.S
(Linear Search) $>$ $\boxed{O(\log_2 n)}$ B.S (Binary Search)

example

$$\rightarrow n = 8$$

L.S

\downarrow

$$O(8) > O(4)$$

B.S

\downarrow

1st Iteration

4 \rightarrow 2nd

2 \rightarrow 3rd

1 \rightarrow 4th

Reverse an Array

↳ We will reverse the array by swapping their values.

• Classic swapping code

$$a \leftrightarrow b$$

```
int temp;
```

$$\text{temp} = a;$$

$$a = b;$$

$$b = \text{temp};$$

Pseudo Code

- i) take start and end
- ii) swap their values
- iii) start + 1, end - 1
- iv) swap their value
- v) till start = end

Pairs in Array

[2, 4, 6, 8, 10]

ex → [2, 4], [2, 6], [2, 8], [2, 10]

[4, 6], [4, 8], [4, 10]

(6, 8), (6, 10)

[8, 10]

• Concept used

- → Nested Loops

for (int i = 0 to n)

 current

 for (int j = i + 1 to n)

 → pairs

Important

Total no. of pairs

$$\Rightarrow \frac{n(n-1)}{2}$$

• Time Complexity → $O(n^2)$

↓
nested loop

#

Sub arrays

→ A continuous part of array.

ex - [2, 4, 6, 8, 10]

→ [2], [2 4], [2 4 6 8], [2 4 6 8 10]

$$\text{Formula} = \frac{n \times (n+1)}{2}$$

Pseudo code

i) start for ($i = 0$ to n)
ii) end for ($\text{int } j = i+1$ to n)
for (start to end)
} subarrays.

④ Time complexity \rightarrow

$$\Rightarrow O(n^3)$$

for each for loop $\rightarrow n$

$$\therefore 3 \text{ For loop} = n^3$$

Max Subarray Sum

$[1, 2, 3, 4, 5]$

\rightarrow

$[1, 3, 6, 10, 15]$

$$\text{Prefix}[i-1] + \text{arr}[i]$$

\downarrow

Prev sum

(Prefix Array)

like a cumulative array
adding all the behind terms

Now to get any array sum

Ex - 2 to 4 \Rightarrow

$$\text{Prefix}[i=3] - \text{Prefix}[i=1-1]$$

$$\text{Prefix}[\text{end}] - \text{prefix}[\text{start}-1]$$

$$\Rightarrow 10 - 1 \left| \begin{array}{l} \text{Prefix}[i=3] - [i=0] \\ [10-1] \end{array} \right.$$

$$\Rightarrow 2 + 3 + 4 \\ \Rightarrow 9$$

* Creating prefix array

$$\Rightarrow \text{Prefix}[0] = \text{num}[0]$$

for($\text{int } i = 1$; $i < \text{prefixlength}$; $i++$) {

$$\text{prefix}[i] = \text{prefix}[i-1] + \text{num}[i];$$

}

Time Complexity = $O(n^2)$

because for every

Max Subarray (KADANE's)

Algorithm

Meaning →

When we get

$(-ve) \rightarrow 0$
make it

$(+ve) + (+ve) \rightarrow \checkmark$

$(+ve) + (-ve) \rightarrow \checkmark$

$(+ve) + (-ve) \rightarrow \times$

So take it
as zero

0

Trapping Rain Water Question

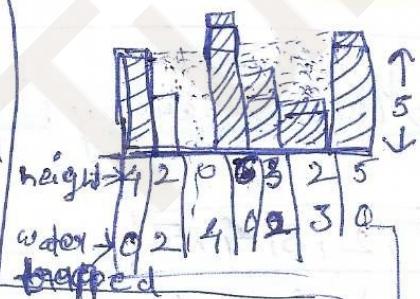
DSA
Sheet

→ Height of bars is given in a array [4, 2, 0, 6, 3, 2, 5]

Q n is non-negative in elevation

means  , width of each bar is 1

compute how much water it can trap?



Ans

→ Volume of water → $\left[\frac{\text{water level}}{\text{level}} - \text{bar level} \right] \times \text{width of bar}$

$$\Rightarrow 0 + 2 + 4 + 0 + 2 + 3 + 0 = 11$$

Final trapped

water → 11

$$\text{Water level} = \min(\max(\text{left boundary}), \max(\text{right boundary}))$$

• Some extreme cases →

→ ~~all bars same height~~

→ ~~no bars~~

No water trap

~~when all bars in a row~~

① minimum Number of bar ≥ 3

② If bars are in ascending or descending order

Auxiliary Array \Rightarrow Helper Arrays

- ex → [2, 4, 4, 6, 6, 6, 6]
- i) Make array of left max boundary
 - ii) Make array of right max boundary → [6, 6, 6, 6, 5, 5, 5]
 - iii) Then take the min ~~at~~ from right and left boundary.
 - iv) Minus the height from the min boundary.
 - v) And you will get your water ~~level~~.
 - vi) volume \rightarrow [water level] \times width

Functions

$\text{Math::max}(a, b);$
 $\text{Math::min}(a, b);$

we will calculate right Max from right onwards

Time complexity = $O(n)$

Most optimum way to solve this

Buy & Sell stocks

var max profit = 0;
var buy price = +∞;

L track the lowest buying price.

for (int i = 0 to n) {

if (buy price < selling price) {

$$P = SP - BP$$

→ profit max?

else (Buy price > selling price) {

Buy Price = selling price

at current price

profit = selling price - buy price

For max profit
= Selling price - $\frac{\text{Max Buy Price}}{\text{Buy Price}}$

Time Complexity
will be
→ only one loop
from 0 to n
 $\therefore O(n)$

Array home work

Q.1

→ Array num [1, 2, 3, 1]

for (int i=0, int < num.length, i++)

for (int b=1, int < num.length, i++)

if (num[i] = num[b]) {

return true;

else return false; }

Brute Force

Time Complexity

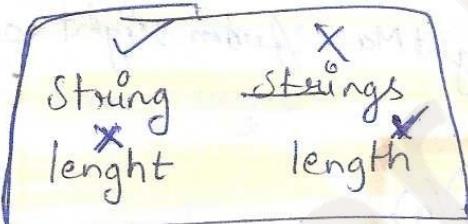
$\rightarrow O(n^2)$

→ We just need to check that is any number is repeated.

→ Just take a number and see is it repeated ahead.



4B



functions Inbuilt

1) Arrays.fill(~~ways~~, -1); ← the whole array will be filled with -1.

Sorting → Arranging items in a particular order.

→ It must be any arranging order or it must be decreasing or any other.

• Bubble sort

• Selection Sort

• Insertion sort

• Counting sort

Time complexity $\Rightarrow O(n^2)$.

Bubble Sort

{5, 4, 1, 3, 2}

unsorted array

{1, 2, 3, 4, 5}

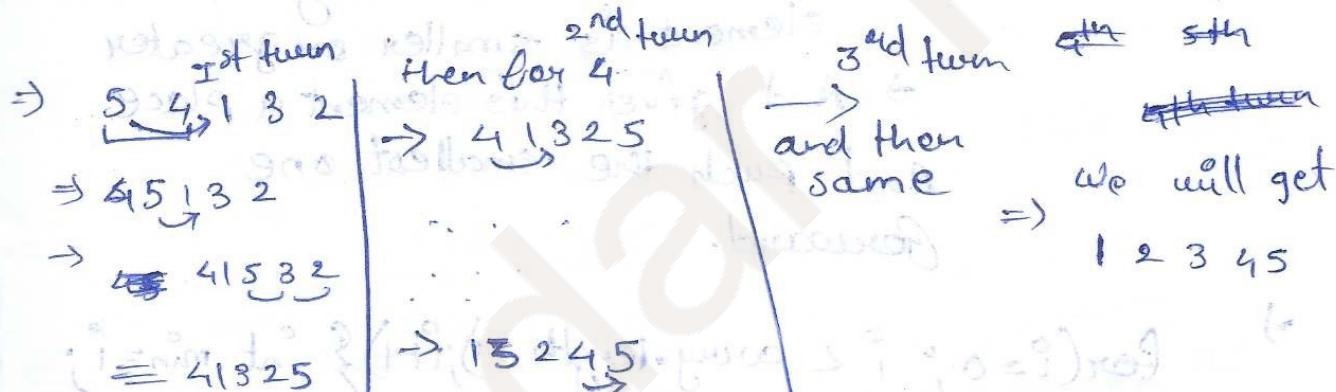
increasing array

{5, 4, 3, 2, 1}

decreasing array.

ex - 5, 4, 1, 3, 2,

$n=5$



Here we
not compare

as we already
know 5 is largest

so we take 3rd turn we will need $n-2$ turns
∴ we will need a for loop

→ for (turns = 0 to $n-2$)

for (int i = 0 to $n-2$ -turns)

i = [] push = [] push = [] push = [] push
push back = [] push = [] push = [] push
push back = [] push = [] push = [] push
push back = [] push = [] push = [] push

Selection Sort → (Pick the smallest [from unsorted] & move it to the beginning)

→ ex- {5, 4, 1, 3, 2} 2nd turn i = 1, n = 5

→ $\rightarrow 1 \boxed{5432}$

for (int i = 0 to n-2) {
 for (j = i+1 to n-1) {
 if (array[j] < array[i]) {
 temp = array[i];
 array[i] = array[j];
 array[j] = temp;
 }
 }
}

→ 1, 2, $\boxed{543}$

→ 1, 2, 3, $\boxed{54}$

→ 1, 2, 3, 4, $\boxed{5}$

→ moves between {5, 4, 1, 3, 2}

time $\rightarrow O(n^2)$

→ First loop → Pick one-one element
Second loop → check ~~with~~ with every next element
→ And gives this element a place
And push the smallest one forward.

→ for (i = 0; i < array.length - 1; i++) { int min = i;
 for (j = i + 1; j < array.length; j++) {

check

5, 4, 1, 3, 2

array[0] = 5

if (5 > 4) {

 min = 4

4 > 1 \rightarrow min = 1 of a - next) not <

min > 3 X 1 < 1 of a - next)

min > 2 X

get value of

→ Swap → temp = array[min] = array[2] = 1

5 \leftarrow array[min] = array[2] = ... New array

1 \leftarrow array[2] = temp $\therefore 1, 4, 5, 3, 2$

and the process goes on

Insertion Sort → Here we just take the first element of array and take it as sorted.

ex → ~~sorted~~ 5, 4, 1, 3, 2 → unsorted

→ then we will take $a = \text{temp}$

→ ~~if~~ $a < 5$ & smaller than 5, if yes
then push it in forward of 5 ←

→ place it in last place if No
and then place the ←
other elements of array
as it is.

→ 4, 5 | 1, 3, 2
sorted unsorted

time $O(n^2)$

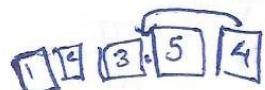
→ same process repeat

→ 1, 4, 5 | 1, 3, 2

→ 1, 3, 4, 5 | 2

→ 1, 2, 3, 4, 5

Idea of this
sort as cards



Inbuilt Sort

{ import java.util.Arrays; ← library
Arrays.sort(arr) ← syntax

($O(n \log n)$) ← time complexity

Time taken is so less than any other.

We can sort up to a dedicated index by

specific
range
sorting

Arrays.sort(arr, si, ei)

↑
name
starting index

ending index

but the ending index is inclusive
 If you want first 2 indexes then you need
 to write ~~to~~ write
 $\rightarrow \text{array.sort}(\text{arr}, 0, 3)$
 \rightarrow Then it will sort till

Counting Sort

(special for positive numbers.
 Quantity must be large but range must be low)

ex -

1	4	1	3	2	4	3	7
---	---	---	---	---	---	---	---

- \rightarrow i) first will make an array that will consist of frequencies of the element (No. of time repeated)

Frequency array
0 2 1 2 0 2 0 0 1

0 1 2 3 4 5 6 7

Here if index is same as input no then
 do count down the element how many times repeated.

- ii) Then we will ~~put back~~ the elements in original array.

1	1	2	3	3	4	4	7
times remaining	1	4	1	3	2	4	3

one count remaining

Time complexity

- ① $\text{for}(\text{int } i=0 \text{ to } n)$ $\rightarrow O(n + \text{range})$
 frequency
- ② $\text{for}(\text{int } i=0 \text{ to } \text{max no. / Range})$
 Setting

- 1) A array that contains all frequencies
+ Must start ~~with~~ from 0 (Name it count)
- 2) Range → It must be the largest element.

2-D Arrays

row	Column	Column 3
1	2	3
4	5.	6

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

There are
30, 40
in arrays

n = ~~Rows~~ ^{Column length} = matrix.length = No. of rows
 m = ~~Rows~~ ^{Column length} = matrix[0].Length = No. of columns

2D arrays in memory

→ Row major method

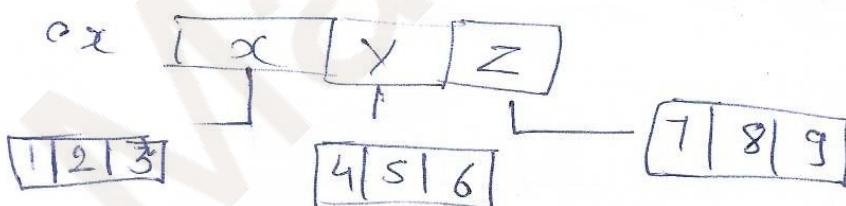
0	1	2
101	104	108

3	4	5
112	116	120

→ Column major.

C ₁	C ₂	C ₃
0 3	1 4	2 5

→ But in java it stores at three different array



• google
• amz
• adobe

Spiral Matrix (Important)

Important companies

→ A matrix will be given you need to print it in spiral way

ex -

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

exception → So that element! Repeat

(1) if ($\text{startRow} = \text{endRow}$) {
 break;
}

(2) if ($\text{startCol} = \text{endCol}$) {
 break;
}

→ bottom left

output → 1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10

Approach

① Declare 4 variables.

- Start Row (SR)
- Ending Row (ER)
- Start column (SC)
- End column (EC)

For outer elements

0 (n-1) 3

0 (n-1) 3

For inner elements

1 2

1 2

front

startRow

++ 3

endRow -

startColumn +

endColumn -

variables

for while ()

{ 1) top part → SR ✓

[SC → End Column]

2) right part → SR+1 → ER ✓

EC ✓

3) bottom part → ER ✓

[EC-1 → SC]

4) left part } → SC ✓

[ER-1 → SR+1]

→ then we will update boundaries

and the again while loop

while (startColumn < EndColumn && StartRow < EndRow)

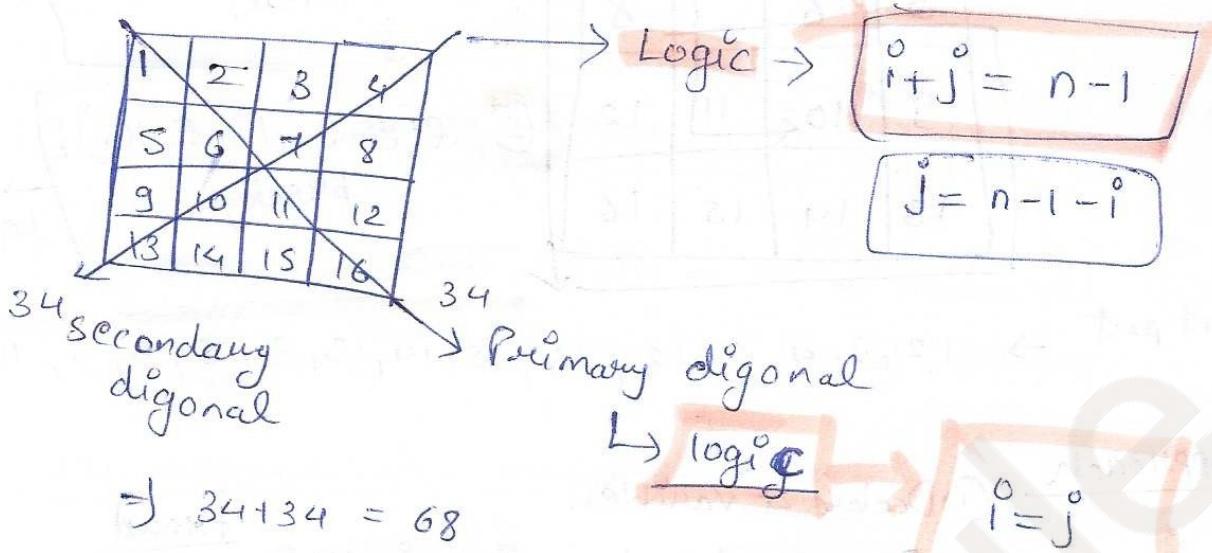
ex - n x m

This is so
that in matrix
like 3×3 middle
element should not be
left

This so that non square
matrix should not be
printed repeatedly.

Exceptions: For n x m type of matrix
(1) if startRow > col. → break in bottom part

Sum of Diagonal Matrix only for $n=m$



$$\Rightarrow \text{int sum} = 0;$$

```

    # Primary diagonal
    for(i=0; i<matrix.length; i++) {
        for(j=0; j<matrix[0].length; j++) {
            if(i=j) {

```

$$\text{sum} = \text{sum} + \text{matrix}[i][j]$$

Secondary diagonal

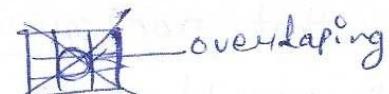
```

    for(i . . . ) {
        for(j . . . ) {
            if(i+j=n-1) {
                if(i=j) { continue; }

```

$$\text{sum} = \text{sum} + (\text{matrix}[i][j]) \} \}$$

exception:



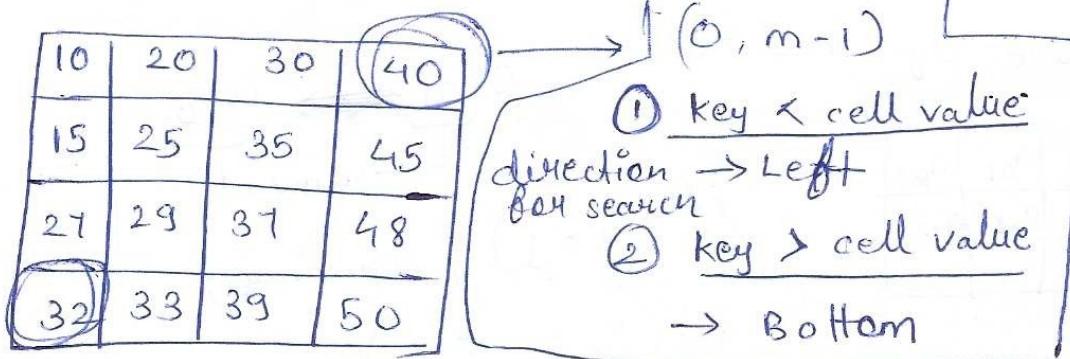
\Rightarrow Add a condition in secondary matrix

```

        if(i=j) {
            continue;
        }
    }
```

Search in sorted Matrix

company - crack



→ Search for a key in

Row wise and column wise sorted matrix

* Approach

① Brute force

Time $\rightarrow O(n^2)$

② Row wise (Binary search)

Search key row wise $\rightarrow \downarrow$

\rightarrow time $n(\log(n))$ $\log(n)$

Row - Search			
10	20	30	40
15	25	35	45
27	31	37	48
32	33	39	50

③ Stair case search

$\rightarrow O(n+1, 0)$

① if key < cell value

direction for search \rightarrow Top

② key > cell value

direction for search \rightarrow Right

* Whenever there will

be some extra information
it will be given in
Question it 100% for
some use -

For example

\rightarrow Here in this question
it is said that
the matrix is sorted
row and col both

\rightarrow This type of information
is 100% for some use
to reduce the time or
space complexity

Approach

Time Complexity

① if $n >> m$ then

extreme case $\rightarrow O(n)$

② if $m >> n$ then

$\rightarrow O(m)$

\therefore Total time complexity

$O(n+m)$

H.W

list of values in arrays

Q. Transpose of matrix

a_{00}	a_{01}	a_{02}
a_{10}	a_{11}	a_{12}
a_{20}	a_{21}	a_{22}

$a_{01} \leftrightarrow a_{10}$
same

if ($i = j$) { continue; }

if ($i + j = i + j$) { swap }

Transpose

a_{00}	a_{10}	a_{20}
a_{01}	a_{11}	
a_{02}		a_{22}

To take out matrix row and column

row = matrix.length; = 2

ex - {{1, 3, 7}, {5, 8, 9}}

col = matrix[0].length; = 3

int col = 3; int row = 2;

for (int i = 0; i < row; i++)

for (int j = 0; j < col; j++)

System.out.println(matrix[i][j]);

System.out.println("-----");

* Strings *

↳ We can store anything in type of text.

char [] = { 'a', 'b', 'c', 'd' }
↓
NOT FOR STRING

String str = "abcd"

* Java also has inbuilt class for storing that is "String".

Ex - Initialisation

① String str = "abcd";

② String str2 = new String("xyz");

* Strings are IMMUTABLE in Java

↳ changes can not be made in them once a string is created.

↳ We always need to create new, ~~but~~ make a change in string

→ Scanner sc = new Scanner(System.in);

String name;

String fullName;

name = sc.next();

System.out.println(name);

fullName = sc.nextLine();

Output:

Tony Stark ↵

Tony

Tony Stark ↵

Tony Stark.

① String is a class

String str = "Name";

② To take out length of string

↳ $\text{str.length()} = 4$

Concatenation

Joining two strings.

ex- String FirstName = "Mandar";

String lastName = "Tole";

String fullName = firstName + " " + lastName;

System.out.println(fullName);

Output :-

Mandar Tole

Char At (i)

To print single elements of a String.

ex- String str = "abcd";

System.out.println(str.charAt(0));

Output : a

Check Palindrome

★ Important

→ Example

① racecar

② noon

company
intuit

→ Compare

[i and $n-i-1$]

for ($i=0$ to $i<a.length/2$)

→ - void boolean() { for ($i=1$ to $i=n/2$)
→ - here - if ($i != n-i-1$) then

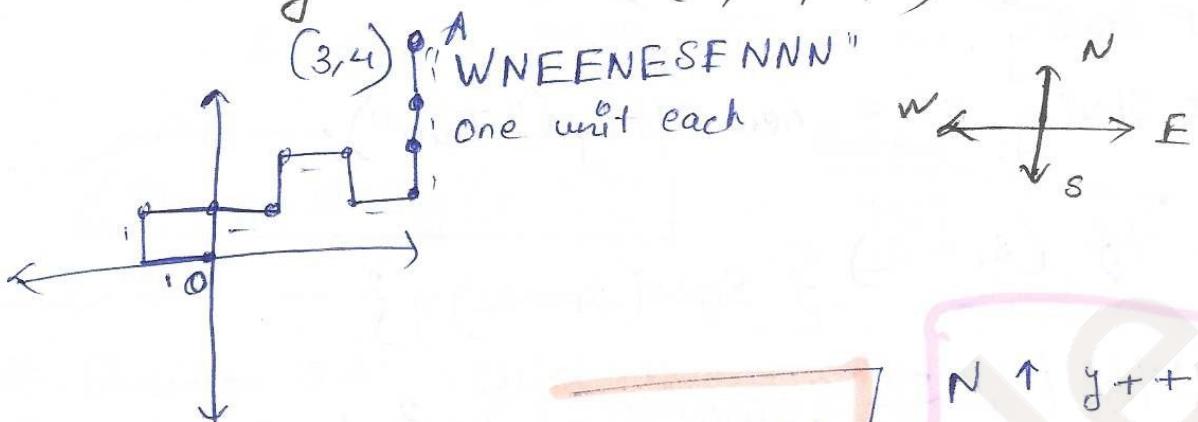
→ { It is not palindrome }

→ return false; } }

→ return false; }

Q. Shortest Path

→ Route containing 4 direction (E, W, N, S)



⇒ Shortest distance \rightarrow O and A

$$\Rightarrow \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\Rightarrow \sqrt{3^2 + 4^2}$$

$$\Rightarrow \sqrt{9+16}$$

$$\Rightarrow \sqrt{25} = 5$$

N \uparrow y++

S \downarrow y--

W \leftarrow x--

E \rightarrow x++

To make square \rightarrow int $x_2 = x * x$

as we know x_1 and y_1 will always be zero

\therefore Math.sqrt($x_2 + y_2$) \rightarrow final.

for (int i = 0 to path.length())

char direction = ~~path.length~~
= path.charAt(i);

Time Complexity \Rightarrow ~~O(n)~~ linear for loop

~~Time co.~~

$\Rightarrow O(n)$

Compare String Function.

int String s1 = "Tony";
int String s2 = "Tony";
int String s3 = new String ("Tony");

① If ($s_1 = s_2$) { sys0($s_1 = s_2$); }

② If ($s_1 = s_3$) { sys0($s_1 = s_3$); } else { sys0($s_1 \neq s_3$); }

Output: $s_1 = s_2$
 $s_1 \neq s_3$.

→ because when we use new string it makes
a new place for string otherwise it
only points towards the old string.

→ To see are they truly equal. we use

s1.equals(s3) ← Function

ex- if($s1.equals(s3)$) { sys0($s_1 = s_3$); }

Output → $s_1 = s_3$

This function checks
the value inside
the strings.

Substring

for ($i = 0$ to 3)

System.out.println(str.substring(i));

ex - str = "Hello, world!"
↓ ↓
substring substring

str.substring(0, 3);

Output :-

Hel

Question 3 → Print largest string

Lexicographic Order

→ ex - Akash, adil,
Mandar

Then

adil ← here as there is d
a kash after a and d is smaller than k.
Mandar

str1.compareTo(str2)

here

A ≠ a

output = 0 if strings are equal

output = -ve then string str1 < str2

output = +ve then str1 > str2

str1.compareToIgnoreCase(str2)

here

A and a treated same.

A = a

Time complexity → Comparison (letter to letter)

$x = \text{no. of letters}$

$O(x \times N)$

What are strings?

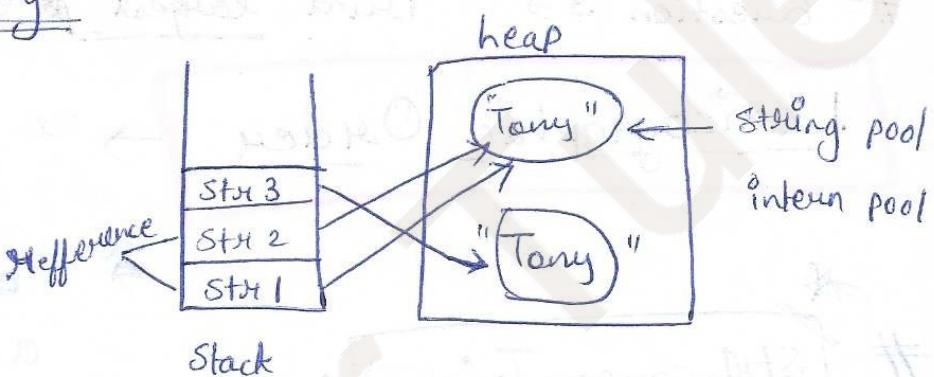
String str1 = "Tony";

String str2 = " Tony";

String str3 = new String("Tony");



Interning

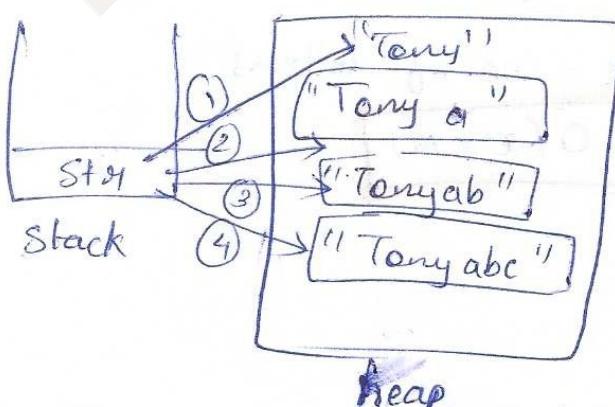


Example ↴

String str = " Tony

```
for (char ch = a; ch <= z; ch++)  
{   str += ch  
}
```

→ here new string will be created each time and our str name will point the new string in each turn.



→ Here the time complexity is so bad because each string needed to be copied from the previous string.

$$\Rightarrow \text{Time} = O(n \times m)$$

$$\approx (n^2)$$

∴ we use String builder,

String builder → same as string but more fast

name. toString() → Convert any ~~datatype~~ Object into string

int a = 10;
a. toString()

X

Integer a = 10;
a. toString()

It need to be the form of class.

we have initialised it empty

→ String Builder sb = new String Builder (" ");

name. append(datatype); → to add elements behind a string
ex- sb.append(ch);

Time complexity ⇒ O(n)

O(2n)

it is less than the string.

Question-

Code Nation

→ For a given string convert each first letter of each word to uppercase

"hi , i am mandar"

or "Hello World"

⇒ "Hi , I Am Mandar" → Hello World"

1. Character.toUpperCase(ch);

First word start from $i=0$

and each new word start from the
~~at~~ next string after space ^{Empty}

time complexity $\Rightarrow O(n)$

String Compression

Example \rightarrow "aaa bbccodd"
Compressed:
 $\frac{3}{a}$ $\frac{2}{b}$ $\frac{3}{c}$ $\frac{1}{d}$
a3 b2 c3 d1

but is abcde

\rightarrow a b1 c1 d1 e1 \leftarrow decompression

int count = 1

↑
This is a
data type.

Integer : Count = 1

↑
This is a class of Integer
it can be used as
string.

time complexity \rightarrow If we use sb.append
then it is less.

\rightarrow If not then $O(n)$.

replace function

Ex - String str = "Ap Manchar";

str.replace ("a", "o")

\rightarrow output \Rightarrow Mondor.

str. toCharArray();

↳ Converts string to character array.

Array. sort(str);

→ Sort the array

Anagram → ex-
① earth, heart
② race, care.

- Q. →
 - ① Convert the strings to lower case. (str.toLowerCase();)
 - ② Check if the lengths are same (str.length();)
 - ③ Change the strings to character array (str.toCharArray();)
 - ④ Sort the arrays both (Arrays.sort(str);)
 - ⑤ Compare the arrays (Array.equals(str1, str2);)
 - ⑥ If same print (It's an anagram)

substring function

String str = "abcde";

sysc(str.substring(1));

sysc(str.substring(3));

sysc(str.substring(5));

sysc("Hi");

Output →

1 - bcde

2 - de

3 - ← empty string

4 - Hi

Substring

(A B C D E)

0 1 2 3 4

this are not indices
Normal count.

Substring Function.

① Type 1 → str.substring(0, 2) → 'AB'
↑ not included.

② Type 2 → str.substring(1) → 'ABCDE'

③ str.substring(0, 0) = "

Here automatically till last element.

* String Function *

s1 = "a ball"

s2 = "is sphere"

① s1.concat(s2) → add a string from behind
print s1 → "a ball is sphere"

② s1.contains(s2) → check whether all elements of s2 are available in s1 or not.