DSA →

Array start from zero.

0 1 2

```
int num[] = {2,3,4};
```

```
sysо ( num[i]);
```

output

⇒ 3

array name [on which place];

Arrays ↳ A type of list to store data

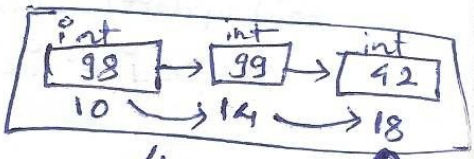
Stating with 0, 1, 2 . . .

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

Defination :-

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

↳ contiguous → one after another ex –



4 byte difference

This is address

* It is a varible which have so many sub-varibles

Size of int → 4 byte difference

imp

To see the length of array

Syso(~~new~~ name of array .length);

Operations on Array

1) Creating an Array

dataType arrayName[ ] = new dataType[size];

ex-

	size
int marks[ ] = new int[50];	→ 50 integeres
int number[ ] = ~~new~~ {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 ~~so~~ store zero

Passing array as An Argument

Important

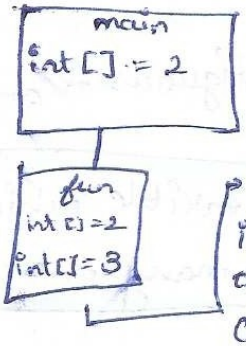
↳ If we made an array in main function And used it in another small function and update it there. then its value also changes in main function.

⇒ class

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

    }

void main (String args[]){
    int marks[] = {97, 98, 99};
```



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 → Its a type of search to search a element in an array.

By element by element.

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;
    int index = linearSearch(numbers, key); //calling the function
    if(index = -1){syso "Not found"};
    else{syso " The key is at "+index};
```

(numbers.length ; array)

---

to get

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

1) $-\infty$ (Smallest no.) → Integer.MIN_VALUE
2) $+\infty$ (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 $-\infty$

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.

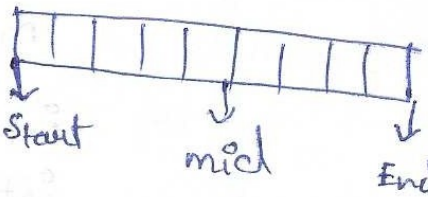
Binary Search

☆ Asked in interviews and coding tests.

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

↳ Prerequisite → Sorted Array → Accending or deccending order

⇒ If we have a long array



i) Find start and end of array.

ii) then find Mid.

iii) If requried key is greater than mid

Key < Mid → take LHS Array

It's S → is same

End → Mid-1

and Find the new mid

and then same process repeat

Key > Mid → take RHS Array

It's End is same

Start = Mid+1

Find the new mid

and then same

i) ii) iii) repeat

Pseudo Code → start = 0, end = n-1

1) while (start <= end)
2) find mid ⟹ [ ]
3) Compare mid & key
4) mid == key → Found
5) mid < key → take Right side array
6) mid > key → take Left side array

mid = start + end / 2

update
end
start

⭐ Important

* while writing array in the parameter write it with [ ]

ex - Public static int binary_Search (int num [], key) {

⊙ Time Complexcity of Binary Search

Iteration 1 → n → $\frac{n}{2^0}$
2 → n/2 → $n/2^1$
3 → n/4 → $n/2^3$
⋮

$n/2^k = 1$

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

$k = \log_2 n$

$TC \propto \log_2 n$

$O(\log n)$

Time Complexity

L.S (Linear Search) > B.S (Binary Search) $\lceil \log_2 n \rceil$

n

example ⇒ n = 8

1st
8 → Iteration
4 → 2nd
2 → 3rd
1 → 4th

L.S ↓ O(8)
B.S ↓ O(4)

O(8) > O(4)

Reverse an Array

↳ We will reverse the array by swaping their values.

- Classic swapping code

a ⟺ b

```
int temp;
temp = a;
a = b;
b = 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

⇒ $\frac{n(n-1)}{2}$

- Time Complexity ⇒ $O(n^2)$
  ↙ nested loop

Sub arrays ↳ A continous part of array.

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

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

Pseudo code

i) start for (i = 0 to n)
ii) end for (int j = i+1 to n)

for (start to end)

≡ } sub arrays.

Time complexity →

for each for loop → n
∴ 3 For loop = n^3

⇒ $O(n^3)$

Max Subarray Sum (Prefix Array)

[1, 2, 3, 4, 5] → [1 | 3 | 6 | 10 | 15]

0 1 2 3 4

Prefix[i] = Prefix[i-1] + arr[i]

Prev sum

→ like a cummulative array adding all the behind terms

Now to get any array sum

ex - 2 to 4 ⇒ Prefix[i=3] - Prefix[i=1]

Prefix[end] - prefix[start - 1] ⇒ 10 - 1 = 9

Prefix[i=3] - [i=0]

[10 - 1]

⇒ 2 + 3 + 4

⇒ 9

Creating prefix array

```
Prefix[0] = num[0]
for(int i = 1 ; i < prefixdength ; i++){
    Prefix[i] = prefix[i-1] + num[i];
}
```

Time Complexity = $O(n^2)$

because for { for }

Max Subarray (KADANE'S Algorithm)

Meaning → when we get

(-ve) → make it 0

(+ve) + (+ve) → ✓

(+ve) + (-ve) → ✓

(+ve) + (-ve) → x

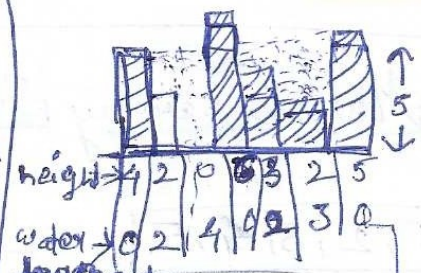
So take it as zero

0

Traping rain water ✡ Question ✡ DSA Sheet

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

Q n is non-negetive in elevation means  , width of each bar is 1, compute how much water it can trap?



Ans ⇒ Volume of water → [water level - bar level] * width of bar

⇒ 0+2+4+0+2+3+0 = 11

Final traped water → 11

Water level = Min (Max left boundry, Max right boundries)

• Some extreme cases →

~~[illegible]~~ → ~~[illegible]~~ → No water trap

~~[illegible]~~ → ~~[illegible]~~

① minimum Number of bar ≥ 3

② If bars are in accending or deccending order

Auxilary Array ⇒ Helper Arrays

ex → [4,4,4,6,6,6,6]

→ i) Make array of left max boundray
ii) Make array of Right max boundray → [6,6,6,6,5,5,5]
iii) Then take the min from right and left boundry.
iv) Minus the height from the min boundry.
v) And you will get your water.
vi) volume ⇒ [water of bars] x width

Functions

```
Math.max(a,b);
Math.min(a,b);
```

we will calculate rightMax from right onwards

Time complexity = O(n)

Most optimum way to solve this

Buy & Sell stocks

profit = selling price - buy price

```
var max profit = 0;
var buy price = +∞;
    └ track the lowest buying price.
for(int = 0 to n){
  if( buy price < selling price){   — Case 1
      P = SP - BP
        └──→ profit max?
  else (Buy price > selling price){
      Buy price = selling price or current price   — Case 2
```

For max profit = Selling price (Max) − lowest Buy price

- Time Complexity will be
  → only one loop from 0 to n
  ∴ O(n)

Array home work

→ Q.1

Array num [1, 2, 3, 1]

```
for(int i=0 ; int < num.length ; i++)
    for( int b=i+1 ; int < num. length ; i++)
        if (num[i] = num[b]){
            return true;
        else return false; }
```

Brute Force
time Complexity
→ (O(n^2))

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

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

4th

String ✓ lenght ✗

~~Strings~~ ✗ length ✓

- functions Inbuit

1) Arrays. fill (name, -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 → $O(n^2)$

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

1st turn

⇒ 5 4 1 3 2

⇒ 4 5 1 3 2

→ 4 1 5 3 2

≡ 41325

2nd turn

then for 4

→ 4 1 3 2 5

. . .

. . .

. . .

→ 1 3 2 4 5

Here we not compare as we already know 5 is larges

3rd turn

→ and then same

4th 5th

4th turn

⇒ we will get 1 2 3 4 5

∴ we take 3rd turn we will need n-2 turn

∴ we will need a for loop

→ for (turns = 0 to n-2)

for (int j = 0 to n-2-turns)

Selection Sort → (Pick the smallest [from unsorted] Put it at the beginning)

⇒ ex- 5,4,1,3,2 , n=5

⇒ → 1 [5 4 3 2] → For (int turn = 0 to n-2) → smaller=
→ 1,2, [5 4 3] for (j = i+1 to n-1)
→ 1,2,3, [5 4] (j = i+1 : unsorted array start.)
→ 1,2,3,4, [5]
→

time → O(n²)

→ First loop → Pick one-one Element
Second loop → check ~~with~~ with every next element is smaller or greater
→ And gives this element a place And Puch 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 min = 0
array[0] = 5

~~if~~ if (5 > 4) { ⇒ ~~5 4~~ / ~~4 5~~ ~~5 4~~

⇒ min = 4
4 > 1 → min = 1

min > 3 ✗
min > 2 ✗

→ swap → temp = array[min] = array[i] = 1 (get value of)
5 ← gets ← arr[min] = [array[i] = New array
1 ← arr[i] = temp ∴ → 1, 4, 5, 3, 2
and the process goes on

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

ex → [5] sorted | [4, 1, 3, 2] → unsorted

→ then we will take 4 = temp

→ is 4 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] sorted | [1, 3, 2] unsorted

time $O(n^2)$

→ same process repeat

→ [1, 4, 5 | 3, 2]

→ [1, 3, 4, 5 | 2]

→ [1, 2, 3, 4, 5]

Idea of this sort as cards

Inbuit 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  ending
               index     index
```

but the ending index is inclusive

to sort first 2 indexes then you need to ~~right~~ write

→ Array.sort(arr, 0, 3)

→ 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

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

→

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

Frequency array

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

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

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

Times remaining

one count remaining

Time complexity

(1) for(int i = 0 to n) → O(n + range)

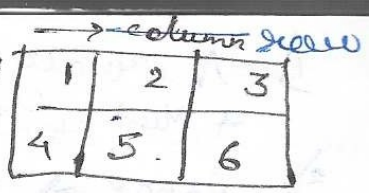
frequency

(2) for(i = 0 to max no./Range)

Selling

1) An array that containg all frequencyies
   + Must start ~~with~~ from 0 (Name it count)
2) Range → It must be the largest element.

2-D Arrays




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

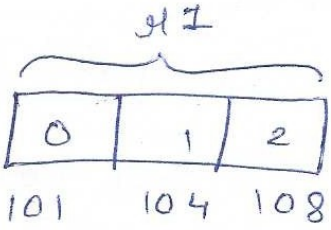
There are 3D, 4D nD arrays

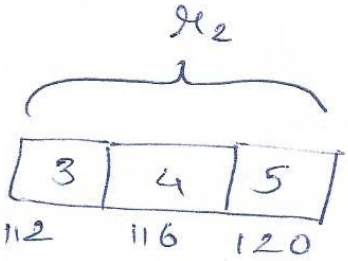
n = ~~Rows~~ Column length = matrix.length = No. of rows

m = ~~Column~~ rows length = matrix[0].Length = No. of Columns

2D arrays in memory

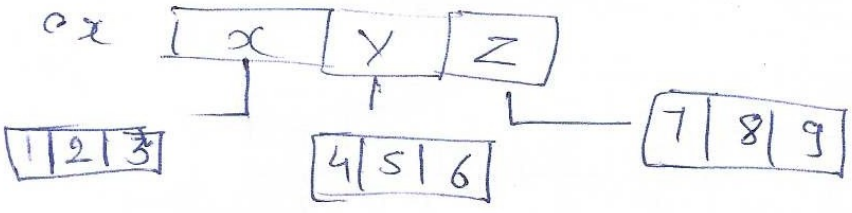
→ row major method





→ Column major.

→ But in java it stores as three different array



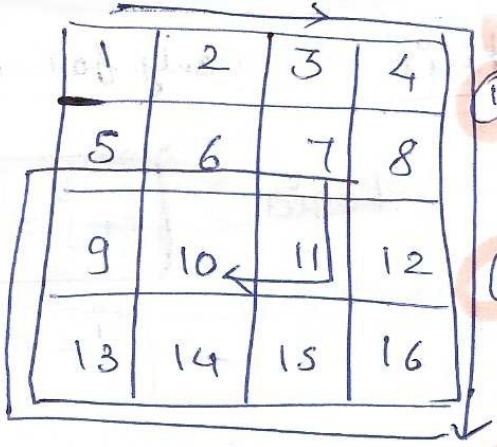
- google
- amz
- adobe

Spiral Matrix (Important)

⭐ Important companies ⭐

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

ex –



exception → So that element ! Repeat

① if (startRow = endRow) { → bolam
break;
}

② if (startCol = EndCol) { → Left
break;
}

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

Approach

① Declare 4 variables.

	For outer elements	For inner elements	trend
• startrow (SR)	0	1	startrow ++
• Ending row (ER)	(n-1) 3	2	endrow – –
• Start column (SC)	0	1	startcolumn + +
• End column (EC)	(n-1) 3	2	endcolumn – –

Variable

~~for~~ while ( )

{ 1) top part → SR ✓ SC → End Colum
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 (startColum ≤ EndColumn && StartRow ≤ EndRow)

ex - n x m

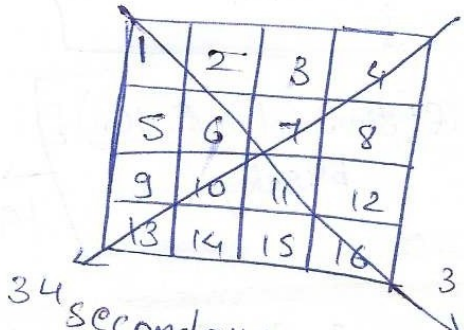
This is so that ~~in~~ matrix like 3x3 middle element should not be left

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

Exceptions : ~~For n x m type of matrix~~

① ~~if row > col → break in bottom part~~

Sum of Digonal Matrix only for n=n



→ Logic → $i+j = n-1$

$j = n-1-i$

34 secondary digonal

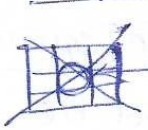
34 Primary digonal

→ 34+34 = 68

↳ logic → $i = j$

```
=> int sum = 0;
   # Primary digonal
for(i=0; i<matrix.length; i++){
   for(j=0; j<matrix[0].length; j++{
      if(i=j){
         sum = sum + matrix[i][j]
# secondary digonal
for(i . . . ){
for(j . . . ){
   if(i+j = n-1){  if(i=j){continue;}
      sum = sum + matrix[i][j]}}
```

exception:



overlaping

=> Add a condition in secondary matrix

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

Search in sorted Matrix

Company- Oracle

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

→ (0, m-1)

① key < cell value

direction for search → Left

② key > cell value

→ Bottom

→ Search for a key in row wise and column wise sorted matrix

* When ever there will be some extra information will be given in Question it 100% for some use -

For example

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

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

* Approach

① Brute force

time → $O(n^2)$

② Row wise (Binary search)

search key row wise → ↓ log(n)

→ time $n(\log(n))$

③ Staircase search

ex - search element 33

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

→ (n-1, 0)

① if key < cell value

direction for search → Top

② key > cell value

direction for search → Right

④ Approach

Time Complexity

extreme case -

① if n >> m then

→ $O(n)$

② if m >> n then

→ $O(m)$

∴ Total time Complexity

$O(n+m)$

H.W

Q. Transpose of matrix

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

$a_{01} \Longleftrightarrow a_{10}$

(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
col = matrix[0].length; = 3
```

ex- {{2, 3, 7},
{5, 8, 9}};

:) Happy

* Strings *

↳ We can store any thing in type of text.

```
char [] = {'a', 'b', 'c', 'd'}
   ↓
String str = "abcd"
```

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

ex - Initialisation

① `String str = "abcd";` (str → Name)

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. ~~control~~ make a change in string

⇒

```
Scanner sc = new Scanner(System.in);
String name;
String fullName;

name = sc.next();
Syso(name)
fullName = sc.nextLine();
Syso(fullName);
```

output:

Tony Stark [Enter]

Tony

Tony Stark [Enter]

Tony stark.

① String is a class String str = "Name";

(i) to take out length of string

str.length() = 4

Concatenation → Joining two strings.

ex- String firstName = "Mandar";
String lastName = "Tule";
String fullName = firstName + " " + lastName;
Sysd(fullName);

Output:

Mandar Tule

charAt(i) → To print single elements of a String.

ex- String str = "abcd";
Syso(str.charAt(0));

output:
N

Company: intuit

Check Palindrome → example

important

(1) racecar

(2) noon

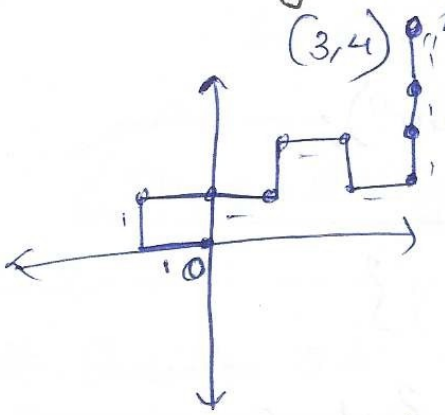
⇒ Compare [i and n-i-1]

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

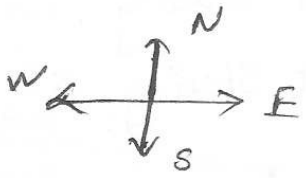
- void boolean(){ for(i=1 to i=n/2)
⇒ - here - if(i-1 != n-i-1) then
→ { It is not palidrome
return false; } }
→ return false; }

Q. Shortest Path

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



"WNEENESENNN"
one unit each



⇒ Shortest distance → O and A

→ $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

→ $\sqrt{(3)^2+4^2}$

⇒ $\sqrt{9+16}$

⇒ $\sqrt{25} = 5$

N ↑ y++
S ↓ y--
W ← x--
E → x++

To make square → `int x2 = x*x;`

as we know x_1 and y_1 will always be zero

∴ → Math.sqrt(x2+y2] → Final.

```
for (int o to path.length();)
    char direction = ~~path.length~~
                   = path.charAt(i);
```

Time Complexity ⇒ linear for loop

∴ ~~time co~~

⇒ O(n)

Compare String Function.

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

① `if (s1 = s2) { Syso(s1 = s2); }`

② `if (s1 = s3) { syso(s1 = s3); } else { syso(s1 ≠ s3); }`

Output: $s_1 = s_2$

$s_1 \neq s_3$.

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

→ To see are they truely equal. we use

`s1.equals(s3)` ← Function

ex- `if(s1.equals(s3)) { syso(s1 = s3) }`

Output → $s_1 = s_3$

This function checks the value inside the strings.

Substring

ex - Str = "Hello world"

↓ substring ↓ substring

~~ex~~ for (i = 0 to 3)

Sysout(charAt(i));

Str.substring(0,3);

Output :-

⇒ Hel

Question 3 ⇒ Print largest string

Lexicographic Order → ex- Akash, adil, Mandar

Then ⇒ adil ← here as there is d after a and d is smaller than k.

akash

Mandar

Str1.compareTo(str2)

here A ≠ a

Output = 0 is 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 = no. of letters

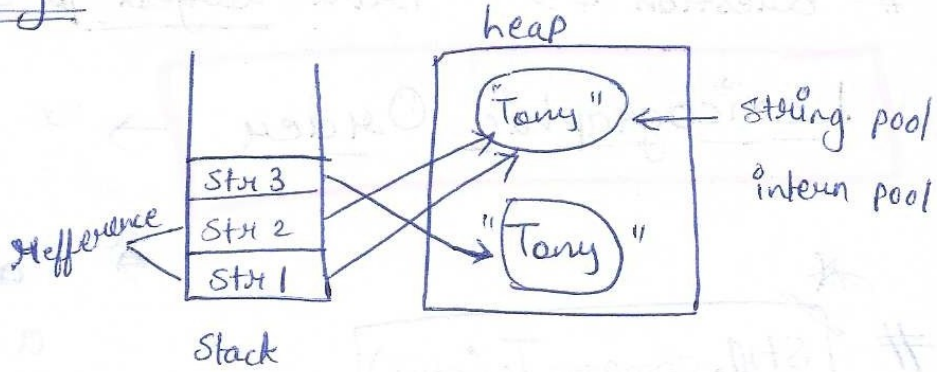
∴ O(x × N)

What are strings ?

```
String str1 = "Tony";
String str2 = "Tony";
String str3 = new String("Tony");
```

↓

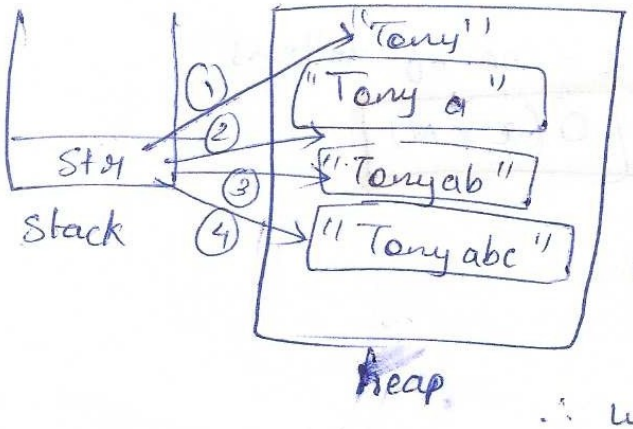
Interning



Example 2

```
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 ∴

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.

⇒
```
StringBuilder sb = new StringBuilder("");
```
we have initialised it empty

```
name.append(datatype);
```
→ to add elements behind a string

ex- sb.append(ch);

Time complexity ⇒ O(n)

O(26)

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"

```
Character.toUpperCase(ch);
```

First word start from i=0
and each new word start from the
~~not~~ next string after Empty space

time complexity ⇒ O(n)

String Compression

⇒ Example → "aaa bb ccddd"
3 2 3 2

Compressed → a3 b2 c3 d2

but is abcde
→ a1 b1 c1 d1 e1 ← decompression

~~int count = 1~~ | Integer Count = 1

↑ This is a data type.

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

time complexit → If we use Sb.append. then it is less.
→ If not then O(n).

---

replace function

ex - String str = "Ap Mandar";

~~str2~~ str.replace("a", "o")

→ output ⇒ Mondor.

- `str.toCharArray();`
  - ↳ Converts string to character array.
- `Arrays.sort(str1);` → 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 (`Arrays.equals(str1, str2);`)
- ⑥ If same print (It's an angram)

~~Sup~~ substring function

```
String str = "abcde";
Syso(str.substring(1));
Syso(str.substring(3));
Syso(str.substring(5));
Syso("Hi");
```

output →
1 - bcde
2 - de
3 - ▭ ← empty string
4 - Hi

Substring

Substring Function.

('A' 'B' 'C' 'D' 'E')
0 1 2 3 4 ← this are not indises Normal count.

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

② Type 2 → str.substring(1) ⇒ 'ABCDE'
↑ Here automatical till last element.

③ str.substring(0,0) = " "

* 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 weather the all elements of s2 are available is s1 or not.