SCALER

String Basics

Starting 9:05

TABLE OF CONTENTS

Revision :-

1.

```
        ←
+0  +0  +1  +1  +0
    1   0   1   1   0
+   0   0   1   1   1
-------------------
    1   1   1   0   1
```

digit	Carry
1+1+2	1/2
2+1+2	2/2
3+1+2	3/2
1+1+2	1/2
1+1+2	1/2

2. $1 @ 1 = 0$

```
  0 0 1
^ 0 0 1
-------
```

a) $1 | 1 = 1$ b) $1 * 1 = 0$ c) $1 \wedge 1 = 0$

d) None

@ = ^.

3. $A \wedge A = 0$

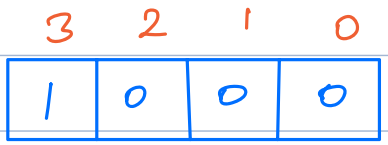
let $A = 5$

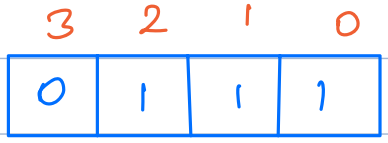
$$\begin{array}{r} 101 \\ \wedge\ 101 \\ \hline 000 \end{array}$$

4. MSB determines +ve or -ve no's

0 → +ve, 1 → -ve

-2^3





$= 2^0 + 2^1 + 2^2$

$= 2^3 - 1$

$-2^3 + 2^2 + 2^1 + 2^0 = -1$

5. For N bits,

$$[-2^{N-1}, +2^{N-1} - 1]$$ long.

For $N = 64$

$$[-2^{63}, 2^{63} - 1]$$

$\sim -9 * 10^{18}, +9 * 10^{18}$.

String :- An array / sequence of characters.

ex:- "Hello"

Character :- Any symbol that is understandable by my machine.

ex :- 'A', 'B', 'C'
'a', 'b', '0', '1', '2', '$'
'_', '*'

Q1- How do computers understand character?

Concept of ASCII values:

"$"

" " → 32

ASCII Values

'A' → 65	'a' → 97	'0' → 48
'B' → 66	'b' → 98	'1' → 49
⋮	⋮	'2' → 50
'Z' → 90	'z' → 122	⋮
		'9' → 57

Some Operations

1.
```
char ch = (char) 65;
print(ch);
```
'A'

2.
```
char ch = (char) ('a' + 1)
print(ch);
```
'b'

3.
```
int x = 'a';
print(x);
```
97

Switch Case

< *Question* > : Given a string consisting of lower-case and upper-case alphabets.

Convert: (1) lowercase → uppercase

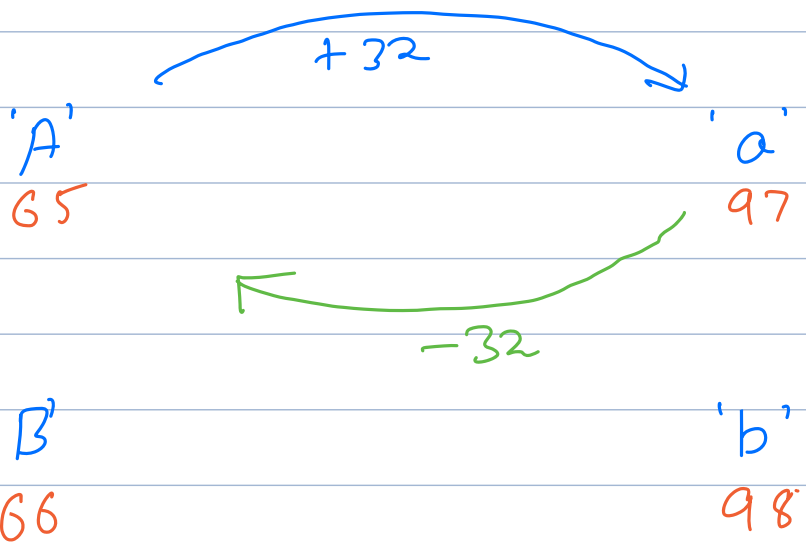
(2) uppercase → lowercase

1. **"H e l l o"** -

"hELLO"

Q:- "aDgbHJe"

"AdGBhjE"



Approach: Iterate over the string. Check if its capital or small.

a) If capital, add 32 to convert to small.

b) If small, subtract 32 to convert to capital.

Code

C++

```
string toggle(string s) {
    int N = s.length();
    for(i = 0; i < N; i++) {
        # Check for capital
        if(s[i] >= 65 && s[i] <= 90){
            s[i] = s[i] + 32;
        } else {
            s[i] = s[i] - 32;
        }
    }
    return s;
}
```

T.C → O(N)

S.C → O(1)

Java | Python

Immutable

```
String toggle (String s) {
    int N = s.length(); String ans="";
    for(i=0; i<N; i++) {
        # Check for capital
        if(s[i] >= 65 && s[i] <= 90){
            ans += (char)(s[i]+32);
        } else {
            ans += (char)(s[i]-32;)
        }
    }
    return ans;
}
```

S → "abcd"

i: 1 2 3

"a" ✗ as "ab" ✓

ans = "abc"

T.C → $O(N^2)$

1. Use String Builder

```
StringBuilder sb = new StringBuilder();
for (int i = 0; i < n; i++) {
    sb.append('a');  // Adds a character efficiently
}
String result = sb.toString();  // Converts StringBuilder to String
```

Avoid O(1)

2. Always convert your String to a character array. Update it & convert it back to a String.

```
----JAVA----
String solve(String A) {
    char arr[] = A.toCharArray(); //for java

    for(int i=0; i < arr.length;i++) {
        if(arr[i] >= 'A' && arr[i] <= 'Z') {
            arr[i] = (char)(arr[i] + 32);
        }
        else {
            arr[i] = (char)(arr[i] - 32);
        }
    }

    return new String(arr);
}

----PYTHON----
class Solution:
    def solve(self, A):
        char_list = []
        for c in A:
            if 'A' <= c <= 'Z':
                char_list.append(chr(ord(c) + 32))
            else:
                char_list.append(chr(ord(c) - 32))
        return ''.join(char_list)
```

3 N

N

3 N

T.C → O(N)

T.C → O(N)

"abcdef"

i

"" ""

1	"a"	ans = 6.
+2	"ab"	
+3	"abc"	
+4	"abcd"	
+5	"abcde"	
+6.	"abcdef"	

$$\frac{6 * (6+1)}{2}$$

Substring

1. Contiguous part of a string.
2. A single character is also a substring.
3. Whole string is also a substring.
4. Empty string (" ") is not a substring.
5. String of length N. How many substrings will be there?

$\frac{N(N+1)}{2}$ Substrings.

ex:- S = "a b c" (0 1 2) = $\frac{3*(3+1)}{2} = 6$

Substrings: a, b, c, ab, bc, abc

Q1- S = "b x c d" (0 1 2 3)

b, x, c, d, bx, xc, cd, bxc, xcd, bxcd

$= 10$.

Check for substring if it's a palindrome or not.

↓ reads the same forward & backward.

"NAYAN"
"MOM"
"RACECAR"
"MALAYALAM"

ex:- S = "a n a m a d a m S p e"
0 1 2 3 4 5 6 7 8 9 10
s e

start → 3
end → 7

$\frac{(e-s+1)}{2}$.

3 7
↓ ↓

```
bool isPalindrome(String str, int s, int e){
    while(s <= e){
        if(str[s] != str[e]){
            return false;
        }
        s++; e--;
    }
    return true;
}
```

T.C → O(N).
S.C → O(1).

10:15.

256 values.

-2^7 to $2^7 - 1$

< *Question* > : Given a string s. Find the length of the longest palindrome substring in s.

ex:- " a n a m a d a m n "

5.

Q:-

" f e a c a b a c a b g f "

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

7.

Q:-

" a d a e b c d f d c b e t g g t e "

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

9

Idea 1: Consider all substrings & check for palindrome.

```
int LPS(String S) {
    int N = S.size();   int ans = 0;
                        # fixing the S.I.
    for(i=0; i<N; i++) {
                        # for the E.I.
        for(j=i; j<N; j++) {
            if(isPalindrome(S, i, j)
                == true) {
                ans = max(ans, j-i+1);
            }
        }
    }
    return ans;
}
```

T.C → $O(N^3)$

S.C → $O(1)$

Idea 2:- Any Palindromic String / Substring is going to be Symmetric around its centre.

N A Y A N   O D D O

Algo: Iterate over each index 'c' of the string & build a palindromic substring by taking (a) c as center.
(b) c & c+1 as center.

l   r.   l   r

" a d a e b c d f d c b e t g g t e "

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

c

r − l − 1   (l, r] :− r − l + 1

3 − 2 − 1 = 0

ans = ~~0~~ ~~1~~ ~~3~~ 9   12 − 2 − 1 = 9

17 − 10 − 1 = 6

(l, r)   = 1 − (−1) − 1 = 1.

= r − l − 1   = 1 − 0 − 1 = 0

= 3 − (−1) − 1 = 3.

= 2 − 1 − 1 = 0

```
int lPS(String s) (

    int N = s.length();   int ans = 0;

    for(c = 0; c < N; c++){
```

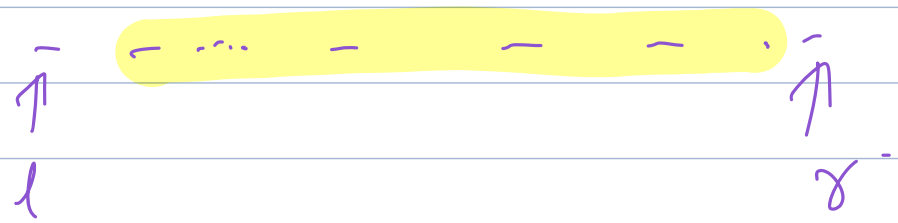
(l, r).

```
# C of a odd.
l = r = c;
while (l >= 0 && r < N) {
    if (S[l] != S[r]) {
        break;
    }
    l--; r++;
}
ans = max(ans, (r-l-1));

# C & C+1 of even;
l = c; r = c+1;
while (l >= 0 && r < N) {
    if (S[l] != S[r]) {
        break;
    }
    l--; r++;
}
ans = max(ans, (r-l-1));
}
return ans;
}
```

T.C → O(N²).

S.C → O(1).

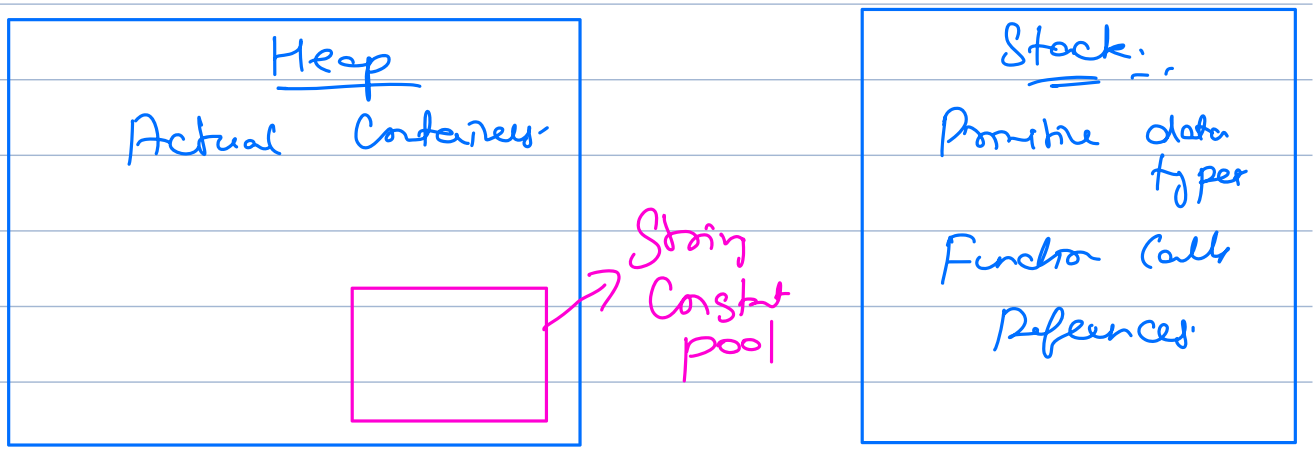


ada

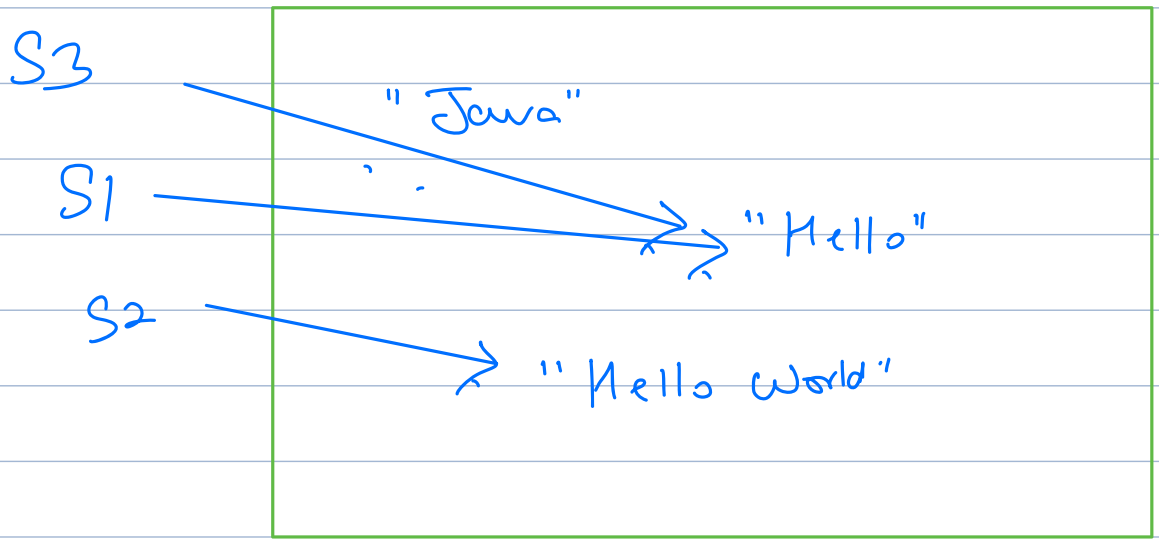
Immutability of Strings

↓

Cannot be modified.



String Constant Pool



1. String S1 = "Java"

2. String S2 = "Java"

3. String s3 = s1;

4. s1 = "Hello";

5. s1.concat("World");

6. s2 = s1.concat("World");

7. s3 = s1

8. String s = new ("Hello");