

"Art enables us to find ourselves and
lose ourselves at the same time."

- §. - Intro
- flip
- Sort
- palindrome

§. String:
collection of char(s)
series of char(s)
array of char(s)
Order is important.

$\{a, c, b\}$ 'acb'
 $\{a, b, c\}$ "abc"

→ Array of char(s)

S = "Hello world"

S = ['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']

⇒ String myString = " " "

'H' = ? // series of 0 and 1

⇒ How characters are stored in machines.

⇒ ASCII values ⇒ '0 to 127'

'A' → 65	'a' → 97
'B' → 66	'b' → 98
'C' → 67	'c' → 99
⋮	⋮
'Z' → 90	'z' → 122

Diagram illustrating ASCII values for uppercase and lowercase letters. A vertical line separates the two columns. A pink arrow labeled +32 points from 'A' (65) to 'a' (97). A green arrow labeled -32 points from 'a' (97) to 'A' (65). Another pink arrow labeled +32 points from 'B' (66) to 'b' (98). A third pink arrow labeled +32 points from 'C' (67) to 'c' (99). A fourth pink arrow labeled +32 points from 'Z' (90) to 'z' (122).

'0' → 48

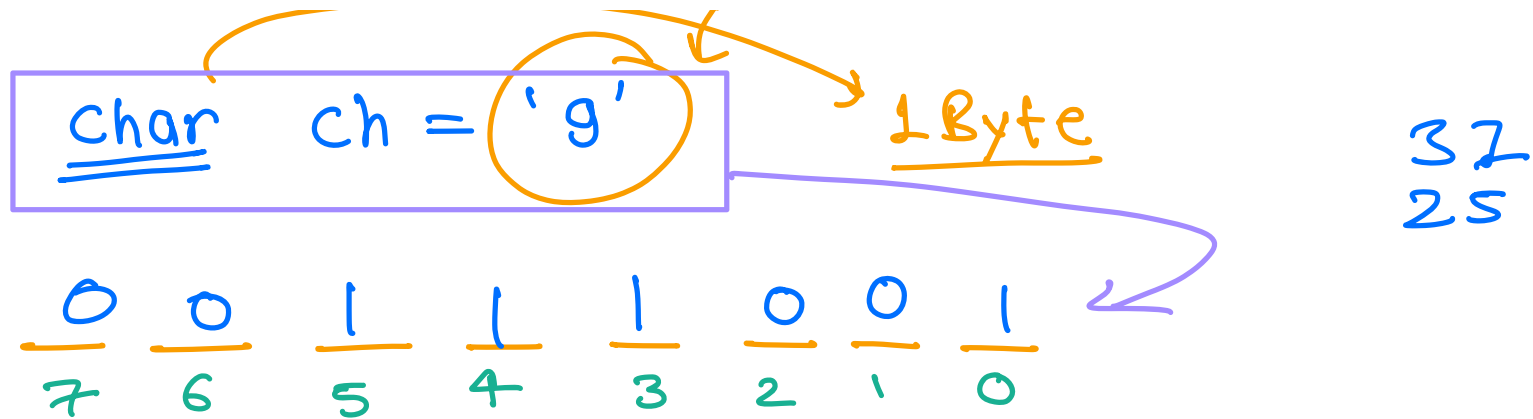
'1' → 49

'2' → 50

⋮

'9' → 57

'0' → 48 49



ch1 = 'g' // 57

⇒ char ch2 = ch1 + 8 // 65

⇒ print (ch2) = ? ⇒ A

Q How to store 2 digit no. as binary?

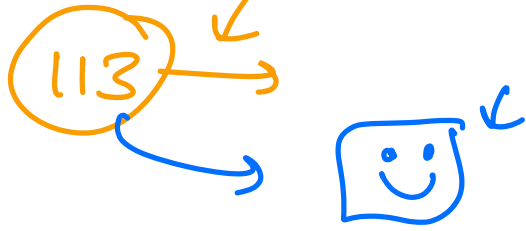
⇒ ['1' '0']

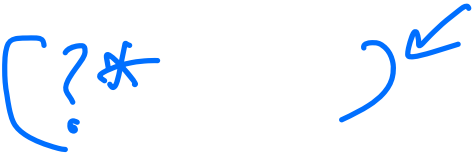
⇒ [

'10' ⇒ int
⇒ string


Q. $ch = 'g' + '8' = ?$ ASCII

\downarrow
 $57 + 56 \Rightarrow 113$





\Rightarrow Open non txt file in text editor?

\Rightarrow 

\Rightarrow an image MS paint \Rightarrow renamed it \rightarrow .txt

string $st = ?$ $ch[]$ $st = ?$

Print "Hello World".

Q. Given a string of alphabets (lower case and upper case characters), toggle the case of each letter (upper case \Rightarrow lower case)

ex.

"Hello" \Rightarrow "hELLO"

idea

- ① if upper case \rightarrow lower case // add 32
② if lowercase \rightarrow upper case // - 32

```
void toggle (ch[] st) {
```

```
    int N = st.size();
```

```
    for (int i=0; i<N; i++) {
```

```
        if (st[i]  $\geq$  65 'A' and st[i]  $\leq$  90 'Z') {
```

```
            st[i] += 32
```

```
        }
```

```
        if (st[i]  $\geq$  97 'a' and st[i]  $\leq$  122 'z') {
```

```
            st[i] -= 32
```

```
        }
```

```

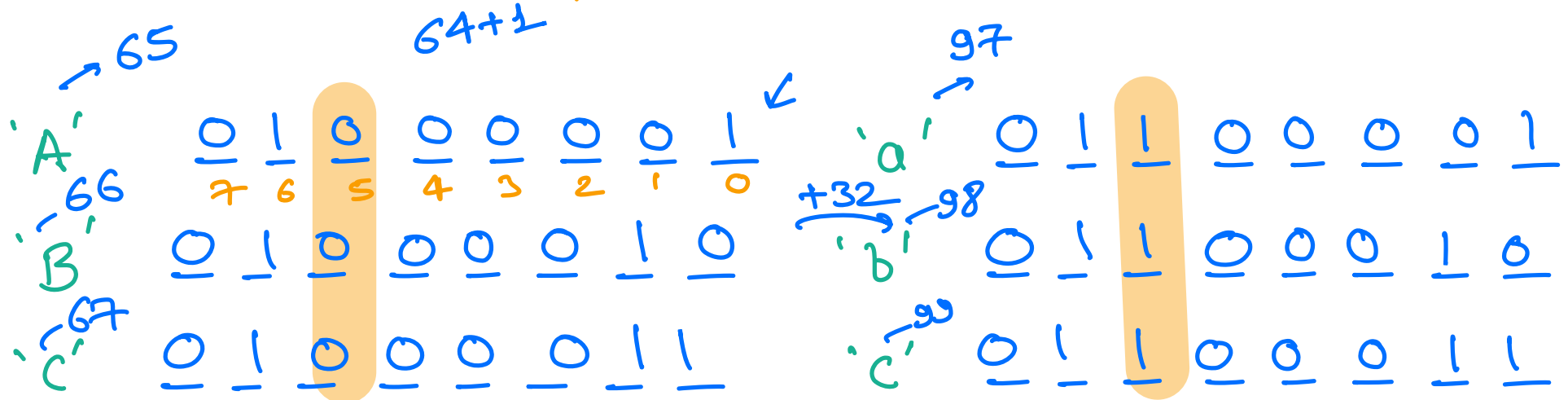
    }
    return;
}

```

"Hello 123" \Rightarrow 'hELLO 123"

(*) Solve the same problem without if/else

\Rightarrow switch, ?!



obsⁿ \Rightarrow ① If capital letter unset sth bit

\Rightarrow ② If small letter set sth bit

XOR¹

```
void toggle ( ch[] st ) {
```

```
    int N = st.size();
```

```
    for (int i=0; i<N; i++) {
```

```
        st[i] = st[i] ^ (1 << 5)
```

```
    }
```

```
    return;
```

```
}
```

input constraints
all the chars are
alphabets

st ^ 32

Q. Given a char array ch[], which contains only lower case alphabets. sort the array. alphabetical order

ex. S = "d a c d d b b a c"

$S = "a a b b c c d d d"$

Constraints, $1 \leq N \leq 10^5$ string size

$'a' \leq S[i] \leq 'z'$

BF

① library sort function

`sort(S);`

T.C. = $O(N \log N)$

S.C. = $O(N)$

merge sort

②

Bubble Sort:

T.C. = $O(N^2)$

S.C. = $O(1)$

TLF
 $O(10^{10})$

③

optimal solⁿ:

ex. $\Rightarrow S = \underline{d} \underline{a} \underline{c} \underline{d} \underline{d} \underline{b} \underline{b} \underline{a} \underline{c}$

'a'

$\Rightarrow 2$

'b' $\Rightarrow 2^2$

|

\Rightarrow

sorted string?

'c' \Rightarrow 2
'd' \Rightarrow 3

| "a a b b c c d d d"

'a' $\rightarrow (-97) \Rightarrow 0$

'b' $\rightarrow (-97) \Rightarrow 1$

'c' $\rightarrow (-97) \Rightarrow 2$

...

...

'y' $\rightarrow (-97) \Rightarrow 24$

'z' $\rightarrow (-97) \Rightarrow 25$

int count[26] = {0}

count[0] \Rightarrow count of 'a'

count[1] \Rightarrow count of 'b'

...

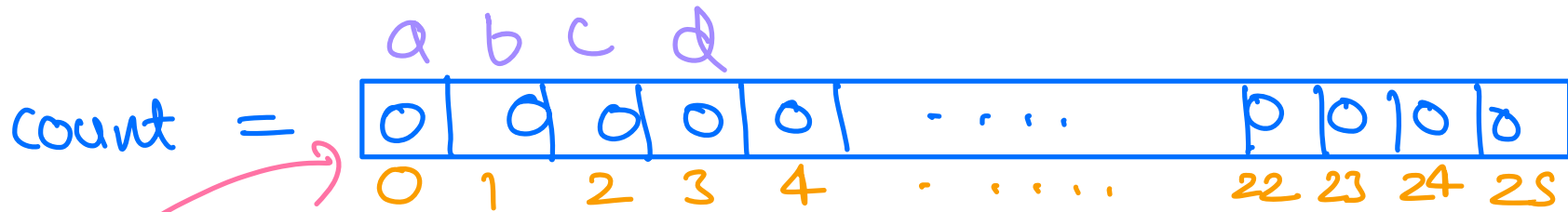
'b'

count[1]

ch == 98 \Rightarrow count[i]++

count[ch - 97]

ex. $\Rightarrow S = "d a c d d b b a c"$



freq

frei b

count ['d' - 97] ++;

Frei-z'

```
count[ch-'a']++;
```

```
void sortString (char[] s) {
```

```
int N = S.size(); char ch;
```

```
int c[26] = {0} // S.C.
```

```
 $O(N)$  → for (i=0; i<N; i++) {  
    int index = S[i] - 97 ——— 'd' - 97  
    c[index]++;  
}
```

```
 $O(N)$  → for (i=0; i<26; i++) {  
    ch = 'a' + i;  
    for (int j=0; j<c[i]; j++) {  
        print(ch); ——— 'c' - 97  
    }  
}
```

```
return;
```

```
}
```

S = "d a c d d b b a c"
N = 9

$C =$

a	b	c	d	e	y	z
2	2	2	3	0	0	0	0
0	1	2	3	4	24	25

$i = 0 \rightarrow 1 \rightarrow 2$
 $ch \Rightarrow 'a' + 0 \Rightarrow 97 + 0 \Rightarrow \underline{\underline{97}} \xrightarrow{\text{char}} 'a'$
 $i = 1 \rightarrow 2 \rightarrow 3$
 $ch \Rightarrow 'a' + 1 \Rightarrow 97 + 1 \Rightarrow \underline{\underline{98}} \xrightarrow{\text{char}} 'b'$
 $i = 2 \rightarrow 3 \rightarrow 4$
 $ch \Rightarrow 'a' + 2 \Rightarrow 97 + 2 \Rightarrow \underline{\underline{99}} \xrightarrow{\text{char}} 'c'$

$c[i] \Rightarrow 223$

"aabbccdd"

i	j	iteration
0	$[0, c[0])$	$c[0]$
1	$[0, c[1])$	$+ c[1]$
\vdots	\vdots	$+ \vdots$
25	$[0, c[25])$	$c[25]$

total iterations = sum of frequency
of each char

$$= c[0] + c[1] + \dots + c[25]$$

$$= \underline{\underline{N}}$$

$$\text{Total T.C} = O(N)$$

$$\text{S.C} = O(26) \Rightarrow O(1)$$

Bucket Sort better than library function,
also known as counting sort

§. Substring: (continuous part of a string)

"Hello" \Rightarrow "eo" ? x

Q. Given a string S, given starting and end point of the substring, check whether the substring is palindrome or not.

palindrome:

naman
n a m a n

madam | madam
reverse

Hello | olleH

ex. S = "abcdcbz", S=1, e=5

0 1 2 3 4 5 6

b c d c b

ans = ✓

10:46

```
bool isPalindrome (char S[], int start, int end) {
```

```
    while (start < end) {  
        if (S[start] != S[end]) {  
            return false  
        }  
    }
```

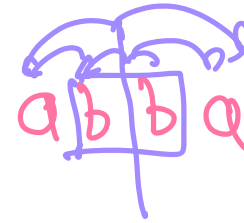
```

    else {
        start++;
        end--;
    }
}


return true;

```

substring length
↓
T.C = $O(N)$
S.C = $O(1)$



Q. Given a string S, find the length of the longest palindromic substring in S.

ex. S = "anamadam", ans = 5


Brute force approach: ① for all the substrings.
 ② check if it's a palindrome

```

int longestPalindrome (char[] St) {
    | int N = St.size();

```

```

int maxLength = 0;
for (int s=0; s<N; s++) {
    for (int e=s; e<N; e++) {
        if (isPalindrome(st, s, e)) {
            maxLength = max(maxLength, e-s+1);
        }
    }
}
return maxLength;
}

```

T.C. = $O(N^3)$
 S.C. = $O(1)$

[e d b a a b c f]
et

Optimised solⁿ:

```

int longestPalindrome (char [] st) {
    int N = st.size();

```



```
int maxLength = 0;
```

```
for (int i=0; i<N; i++) {
```

```
// checking only odd lengths  
// i is the center of my string
```

```
left = i;  
right = i;
```

```
while (left >= 0 and right < N) {
```

```
    if (st[left] != st[right]) {  
        break;
```

```
    }  
    left--;  
    right++;
```

```
    }  
    maxLength = max (maxLength, right-left+1)
```

// even length

left = i
right = i + 1;

while (left \geq 0 and right $<$ N) {
 if (st[left] != st[right]) {

break;

}

~~left++;~~ left--;
right--; right++;

}

}

maxlength = max (maxlength, right - left + 1)

return maxlength;

T.C. = $O(N^2)$
S.C. = $O(1)$

}

St = NITINPQ

ans $\Rightarrow \cancel{0} 1$

DP \Rightarrow Recursion + Memory