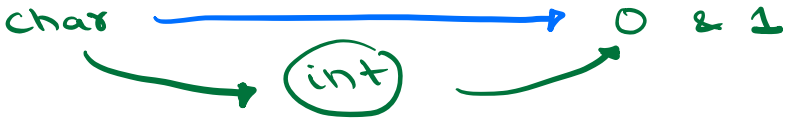
Strings → sequence of characters

"Apurva" "scaler"
"class" , "xyz123"

'a', 'A', '@', '-', '0', ...



'a' → 0 '0' → 50 'b' → 32

'a' → 50 'b' → 0

ASCII

256 chars → int

Unicode

UTF-8 UTF-16

'a' → 97
'b' → 98
'c' → 99
'd' → 100
⋮
'z' → 122

'A' → 65
'B' → 66
'C' → 67
⋮
'Z' → 90

'0' → 48
'1' → 49
'2' → 50
⋮
'8' → 56
'9' → 57

'10' → 58
'11' → 59
'12' → 60

Typed lang
↓
C++ / Java

```
char temp = 97;
```
↑
'a'

✓ Char → int ASCII

```
int x = 'a';
```
✓
97

256 < 0 ! 255

```
int z = 'b' - 'a';
```
int int
98 - 97
= 1

implicit
↓
internal conversion of data type

```
char temp = 300
```
(overflow)
↓
1 B → 8 B (C++)
2 B → 16 B (Java)

1 0 0 1 1 0 1 1 1

```
char z = 'b' - 'a'
```
int int
98 - 97
= 1

char is mapped to 1

ASCII
ch → 1

can't modify the string

Java, C#, Python → strings immutable
↓
stringbuilder

C++, JS → mutable strings

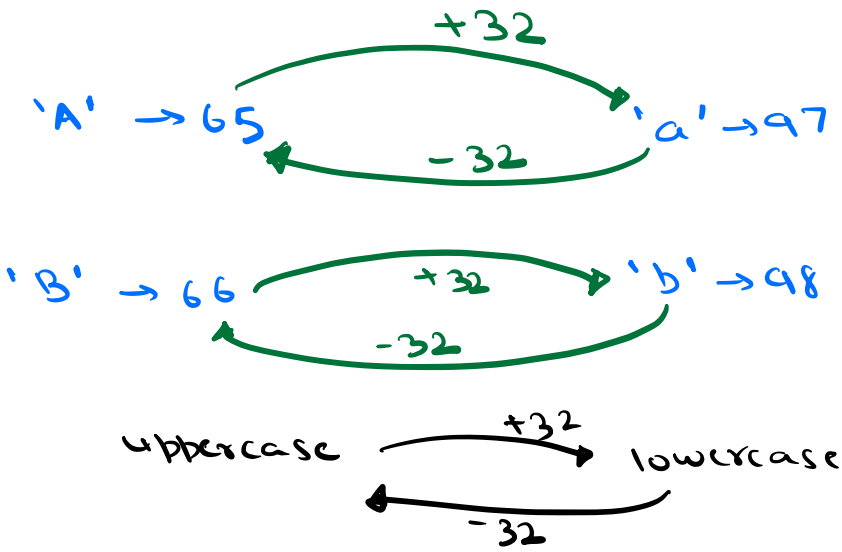
lowercase

uppercase

1. Given a string of uppercase & lowercase characters. Toggle case of each character.

S = "a ABbzZjDf"

S = "A a b B Z z J d F"

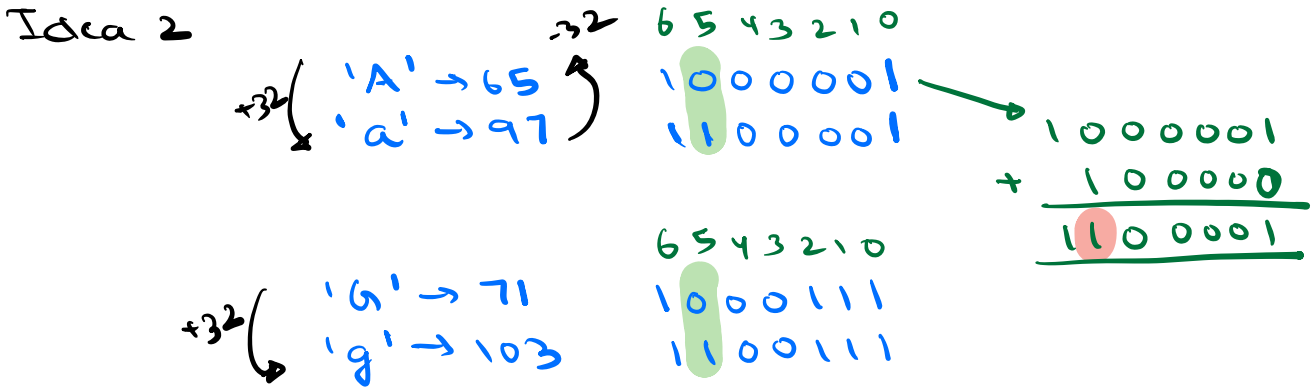


str → arr of chars

```
for (i=0 ; i<len; i++) {
    if (s[i] >='A' && s[i] <='z')
        s[i] = s[i] + 32
    else
        s[i] = s[i] - 32
}
```

TC: O(len)
SC: O(1)

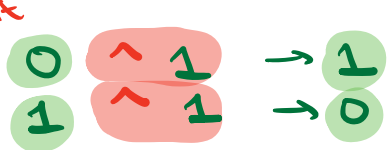
Idea 2



Toggle

toggle 5th bit

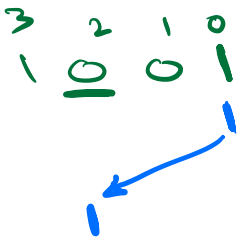
Bit



1 1
& 1

N ^ (1<<i)

1 is at ith bit




```
for (i=0 ; i< N ; i++)
{
    s[i] = s[i] ^ (1 << 5)
}
```

32

2. Given a string of lowercase characters, sort it.

s = adefacd

print(aacddef)

a → 2
b → 0
c → 1
d → 2
e → 1

aa
c
dd
e
⋮

'a' → 'z'
97 → 122

i=97 ; i<=122 ; i++

```
for(i='a'; i<='z'; i++){
    count=0
    for(j=0; j<len; j++){
        if(s[j]==i)
            count++
    }
    // print i cnt no. of times
}
```

26 N

TC: O(N)

26
↓
256 N

1. sorting fn for strings

2. inbuilt sort
↓
comparator

```
bool comparator(char x, char y){
    if(x<=y)
        return true
    else
        return false
}
```

freq of 26 chars

↓

array

'a' in S → 3

freq

freq [ind]

freq['a']

'a' → index

↓

97

we can use ascii value as our indexs

freq [97] → count of 'a'

freq [98] → 'b'

⋮

freq [122] → 'z'

freq [123] <0 - 122> valid indices

0 - 96 → wasting

* freq [26] <0 - 25>

freq []

'a' - 97 → 0

'b' - 97 → 1

'c' - 97 → 2

'd' - 97 → 3

a 97-97 → 0

⋮

z 122-97 → 25

ch - 97 → ind ?

bbca

↳ abcc

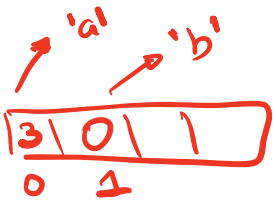
freq [ch - 97]

freq [s[i] - 97] ++

- freq [26] = <0> |0|2|0| ... |
  0 1 2 ... 25

```
for (i=0 ; i< len ; i++)
<
    freq [ s[i] - 97 ] ++
>
```

s[i] - 'a'



0 → 'a'

```
for (i=0 ; i <=25 ; i++) <        // 'a' ... 'z'
    char temp = i + 97 / i+'a' ;
    while(freq[i] >0) <
        print (temp)
        freq[i] --                // 'a ... a'
    >
>
```

N + 26 + N
= 2N + 26

TC : O(N)
SC : O(26)
↓
O(1)

'a'
'b'
:
'z'
} N

To DO str of uppercase & lowercase chars

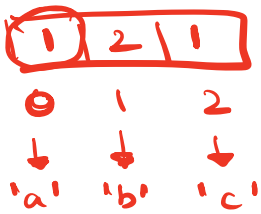
freq [52]

'A' < 'a'

Ex → BaA

Rs → ABa

bbca



freq

freq [26]

'a' → 'z'

ind → 'b' - 97

ind

0 +97 = 97

1 → ch

'a'

Intermediate Contest — 2 / Arrays

Advanced Contest - 7/8

Friday - Sunday 8 PM

2-3 hours

Leaderboard

Ques → Dashboard → Practice

Contest discussion class — 28th Aug → 9 PM — Recording / Notes

Easy
Medium
Hard

3. Given a string S, reverse substring from index L to R.

→ like Subarray

→ continuous part of string

	0	1	2	3	4	5	6	7	8	9	10	11
Ex	a	b	c	d	a	f	a	c	d	e	f	g

→ immutable

```
String reverse (String s, int l, int r) {
    while (l < r) {
        swap(s[l], s[r]);
        l++;   r--;
    }
    return s;
}
```

4. In string S, reverse the string word by word.
Note - All words are separated by space.
No trailing space at start and end.

Ex "This_is_a_key"
"key_a_is_This"

→ No extra space

→ No inbuilt fn

"mail_water_hi"
"hi_water_mail"

Reverse the string ①

"mail_water_hi"
"ih_retaw_liam"

② Reverse each word

ih_retaw_liam

l, r

word → [l, r-1]

1. reverse complete string → reverse(s, 0, n-1)

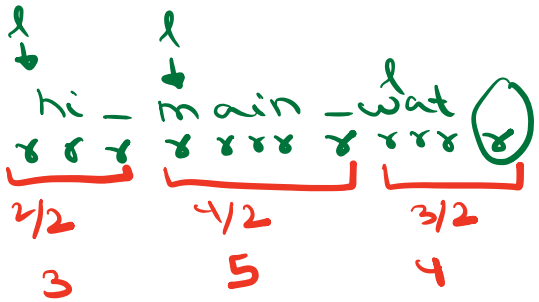
2. int l = 0, r = 0

```
while (r < len) {
    while (r < len && s[r] != '_') {
        r++
    }
    reverse(s, l, r-1)
    r++
    l = r
}
```

TC: O(len)

len + len/2

len <



5. In a string S, find length of longest palindromic substring.

palindrome
↓
s = reverse (s)

nitin ✓

malayalam ✓

abca ✗

```
bool ispal (str, l, r) {
    while (l < r) {
        if (str[l] == str[r])
            l++  r--
        else
            return false
    }
    return true
}
```

0 1 2 3 4
s = n i t i n
reverse → n i t i n
4 3 2 1 0

n x y z

→ ←
n i t t i n

Ex a b a c a b
ans = 5

aba → 3
bacab → 5

Ex a b a c a b a b c d
ans = 7

Basic → go to all substrings, check palindrome

(s, e)

```
ans = 0
for (s = 0 ; s < n ; s++)
    for (e = s ; e < n ; e++) {
        // [s e]
        if (ispal(str, s, e)) {
            ans = max(ans, e - s + 1)
        }
    }
return ans
```

1 substr → N/2 iteration

N^2 → $N^2 \times N/2$

TC: $O(N^3)$

SC: $O(1)$

N^2 →

N^2 → DP

N → Machaner's algo

E → C ← E

a b c d c b a

Idea 2

Palindrome

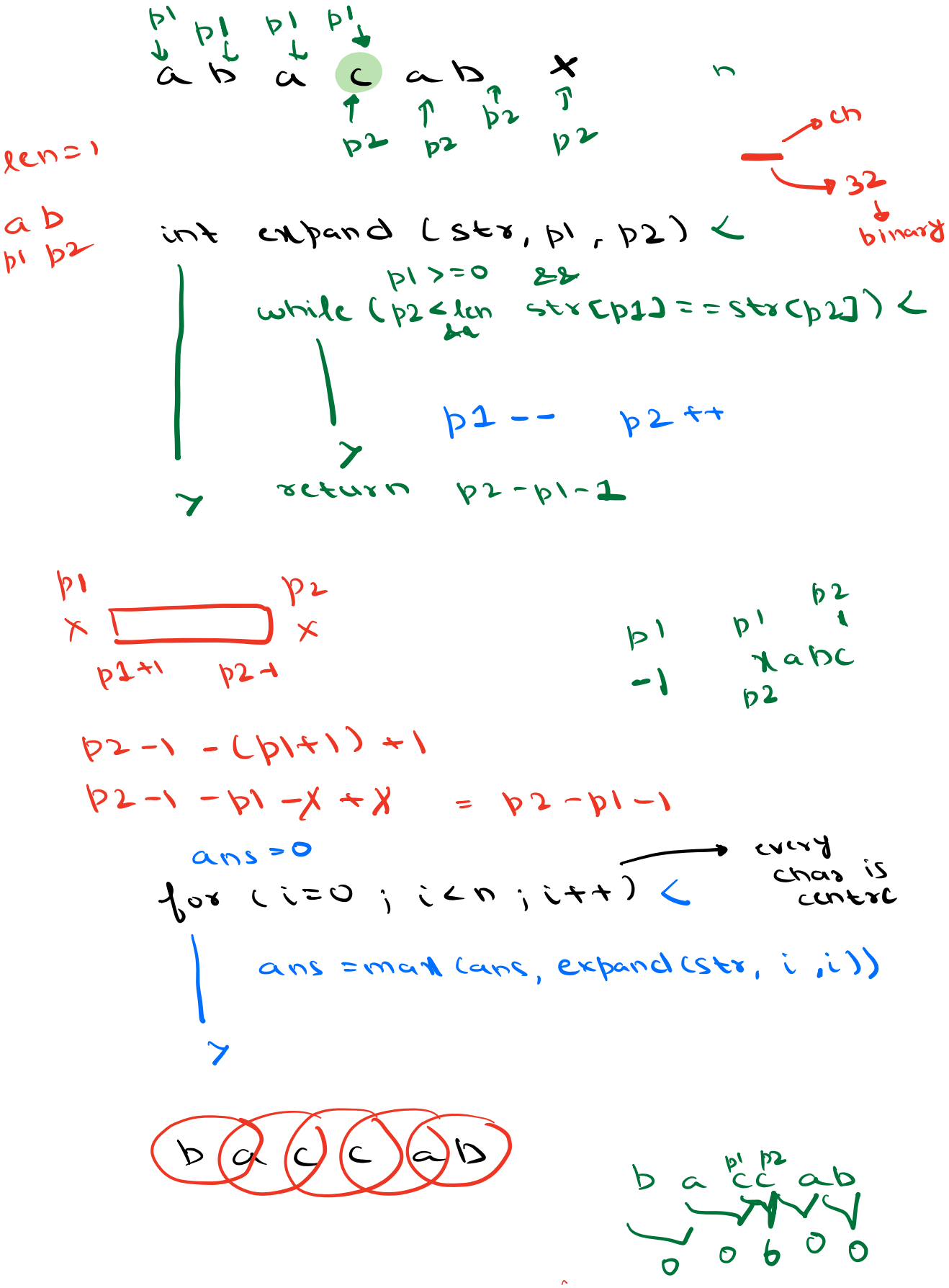
1) odd length

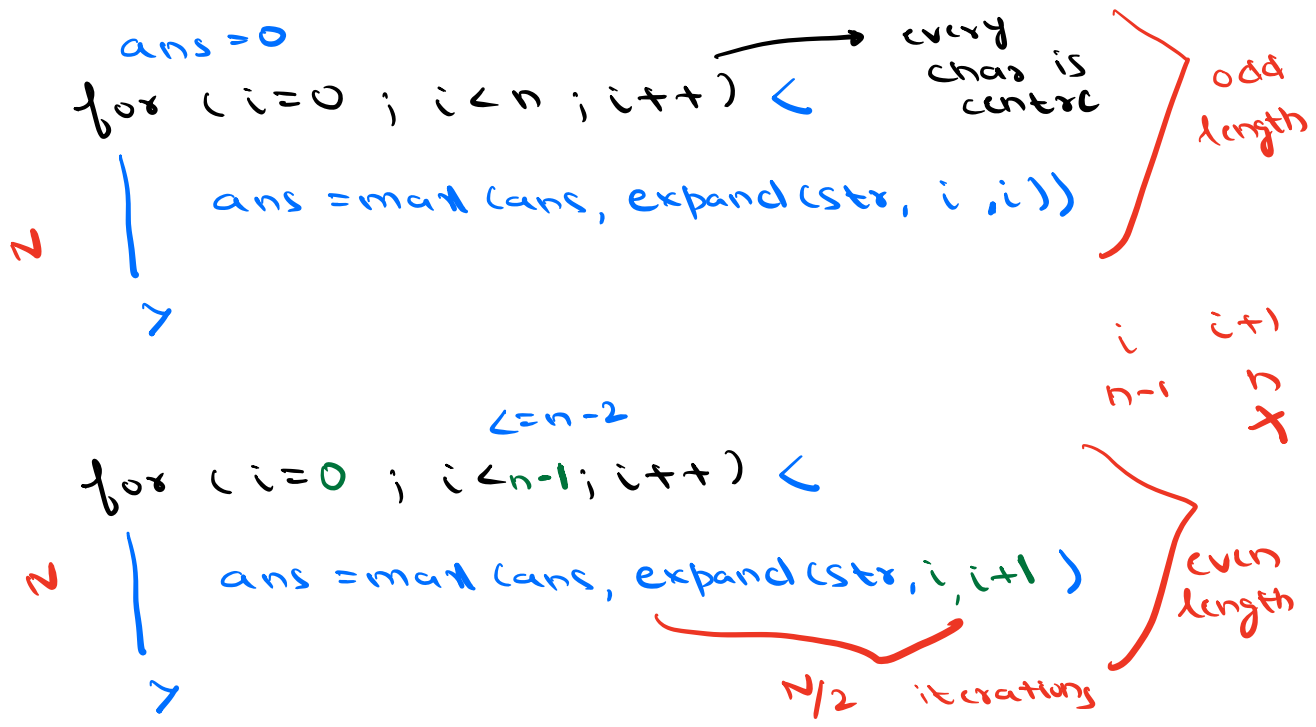
nitin

2) Even length

p1 p2 p2 p1

n i t t i n


p1 p1 p1 p1
a b a c a b x
p2 p2 p2 p2
n
ch
32
binary
len=1
a b
p1 p2
int expand (str, p1, p2) {
p1>=0 &&
while (p2<len && str[p1]==str[p2]) {
p1--
p2++
}
return p2-p1-1
p1
p2
x x
p2+1
p2-1
p1
-1
p2
p1
1
xabc
p2
p2-1-(p1+1)+1
p2-1-p1-x+x = p2-p1-1
ans=0
for (i=0; i<n; i++) {
every char is centre
ans=max(ans, expand(str, i, i))
}
b a c c a b
b a c c a b
p1 p2
0 0 6 0 0



TC : O(N²)

SC : O(1)

xyzabc

↓

1