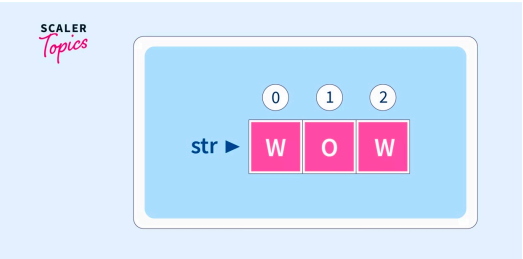
Strings



Agenda

- Introduction
- Flip
- Count arr[]
- Reverse string
- Longest palindromic substring

What is a string ?

~~Collection of chars~~

Sequence of chars

"scaler"

"rsaelc"

How are strings stored ?

ASCII

'a' → 97

'b' → 98

'c' → 99

'd' → 100

⋮

'z' → 122

'A' → 65

'B' → 66

'C' → 67

⋮

'Z' → 90

'0' → 48

'1' → 49

'2' → 50

'3' → 51

⋮

'9' → 57

"10"

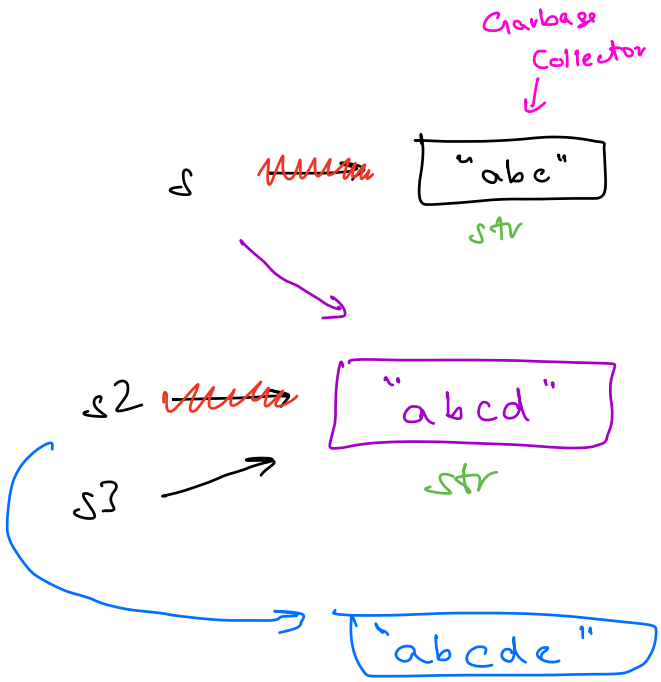
'1' → 49

'0' → 48

Strings are immutable

(In Java & Python)

```
s = "abc"
s += "d"
```



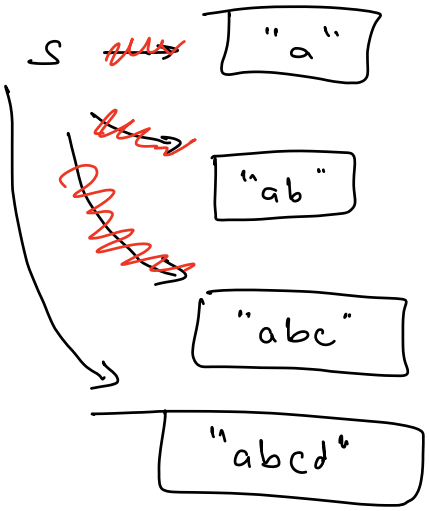
- → Security
- → Saves Memory

```
s2 = "abcd"
s3 = "abcd"
s2 += "e"
print(s)  // abcd
```

Issue with Immutable Strings - Frequent updates

```
s = "a"
s += "b"   ← O(N) time
           ← O(N) space
s += "c"   ← O(N) time
           ← O(N) space
s += "d"   ← O(N) time
           ← O(N) space
```

N updates : :



Quiz 1

N updates

Total time - $O(N^2)$

Total space - $O(N^2)$

How to handle frequent updates ?

Java

```java
class Updates {
    public static void main(String[] args) {
        String s = "abc";
        StringBuilder sb = new StringBuilder(s);
        sb.append("d");
        sb.append("e");
        sb.append("f");

        s = sb.toString();

        System.out.println(s);
    }
}
```

Python

```python
s = "abc"
s = list(s)  # [a, b, c]

s.append("d")
s.append("e")
s.append("f")

s = "".join(s)
print(s)  # abcdef
```

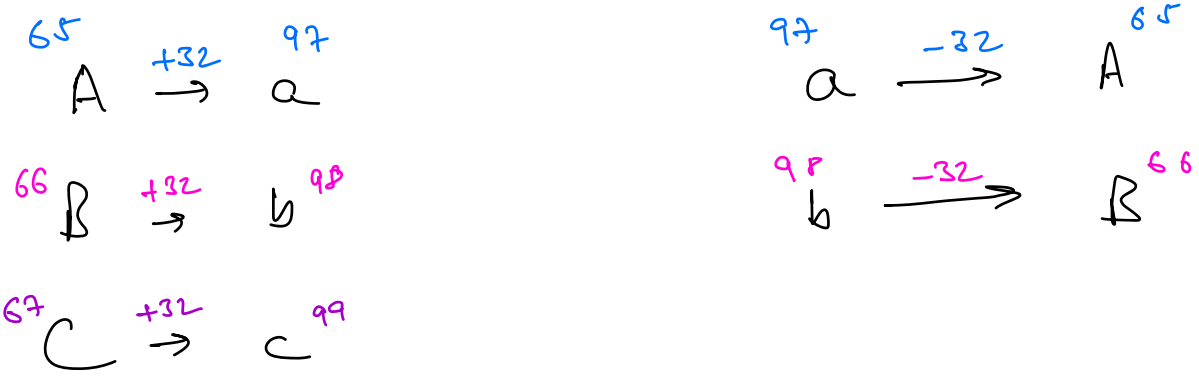
TC : O(N)
SC : O(N)

s = list("abc") → ['a', 'b', 'c']

Q1 Given a string, toggle the case of every character.

No inbuilt functions.

aBcdEf ⟶ AbCDeF




```
toggleCase ( String s) {
    diff = 'a' - 'A'
    for(i=0; i< s.size() ; i++) {
        if( 'A' <= s[i] <= 'Z' )
            s[i] += 32 diff
        else if ( 'a' <= s[i] <= 'z' )
            s[i] -= 32 diff
    }
    return s
}
```

Java

```java
String toggle(String s) {
    StringBuilder sb = new StringBuilder(s);
    int diff = Math.abs('a' - 'A');
    for (int i = 0; i < sb.length(); i++) {
        char c = sb.charAt(i);
        if (c >= 'a' && c <= 'z') {
            sb.setCharAt(i, c -= diff);
        } else {
            sb.setCharAt(i, c += diff);
        }
    }
    return sb.toString();
}
```

Python

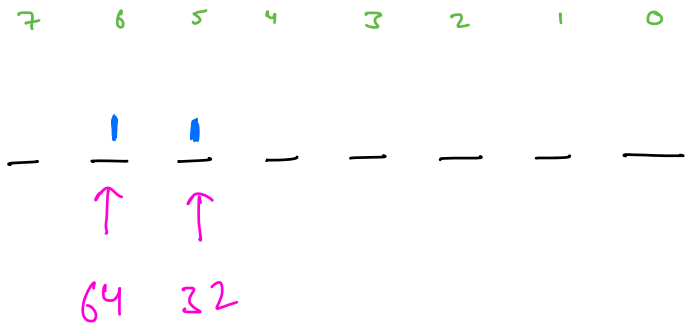
```python
def toggle(s):
    l = list(s)
    diff = abs(ord("a") - ord("A"))
    for i in range(len(l)):
        if "a" <= l[i] <= "z":
            l[i] = chr(ord(l[i]) - diff)
        elif "A" <= l[i] <= "Z":
            l[i] = chr(ord(l[i]) + diff)

    return "".join(l)
```

Time - O( )

Space - O( )

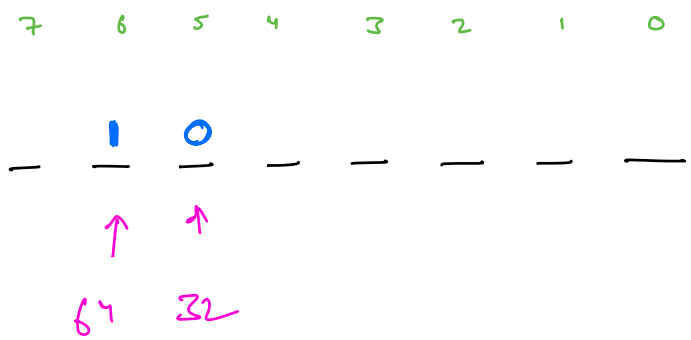
Approach 2 - Using Bit Manipulation

a → 97
b → 98
c → 99
⋮ ⋮
z → 122

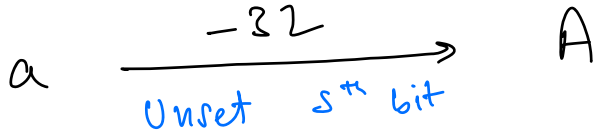


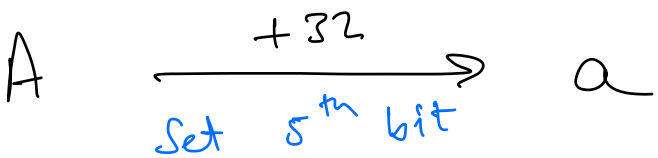
64 + 32 = 96

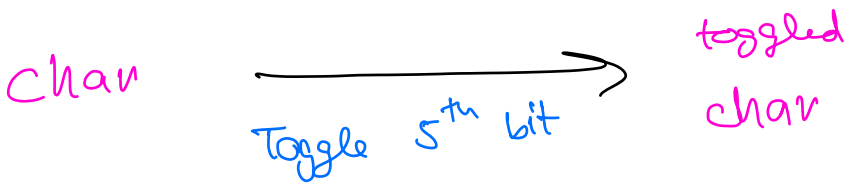
A → 65
B → 66
C → 67
⋮ ⋮
Z → 90



Diff = 32 = 2^5







$$s[i] = s[i] \wedge (1 << 5)$$

$$= s[i] \wedge 32$$

Java

```java
String toggleXor(String s) {
    StringBuilder sb = new StringBuilder(s);
    for (int i = 0; i < sb.length(); i++) {
        char c = sb.charAt(i);
        sb.setCharAt(i, (char) (c ^ 32));
    }
    return sb.toString();
}
```

Python

```python
def toggleXor(s):
    l = list(s)
    for i in range(len(l)):
        l[i] = chr(ord(l[i]) ^ 32)
    return "".join(l)
```

Time - O(N)

Space - O(N)

Q2. Given a string of lowercase characters, sort it in dictionary order.

s : d a b a e d b ⟶ a a b b d d e

Library Sort ⟶ O(N log N)

Lowercase chars → 26

[a - z]

s : d́ á b́ á é d́ b́

1) Count the frequency of every char

2) Construct the sorted string from the freq map.

char	freq
d	✗ 2
a	✗ 2
b	✗ 2
e	1

a a b b d d e

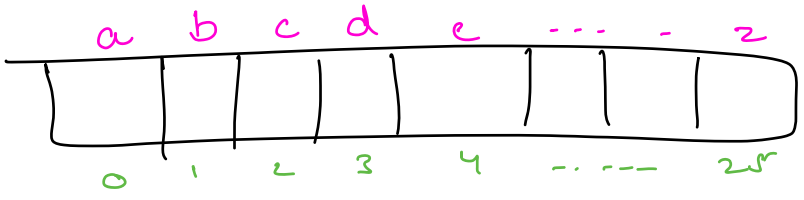
Data structure for → Hashmap

freq map → Dictionary

Array of size = 26

int count[26] = {0}

count = [0] * 26



char		index
a=97	-97 →	0
b=98	-97 →	1
c=99	-97 →	2
⋮		⋮
z=122	-97 →	25

```
int count[26] = {0}

// Construct freq map
for(i=0; i < s.size(); i++) {
    // index for s[i] = s[i] - 97
    index = s[i] - 97
    count[index] ++
}

// Construct sorted string from freq map
res = ""
for(i=0; i < 26; i++) {
    char ch = i + 97
    int freq = count[i]
    for(j=0; j < freq; j++) {
        res.append(ch)
    }
}
return res
```

Java

```java
String dictionarySort(String s) {
    int[] count = new int[26];
    for (int i = 0; i < s.length(); i++) {
        int index = s.charAt(i) - 'a';
        count[index]++;
    }

    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < 26; i++) {
        for (int j = 0; j < count[i]; j++) {
            sb.append((char) ('a' + i));
        }
    }

    return sb.toString();
}
```

Python

```python
def dictionarySort(s):
    count = [0] * 26  # O(1) Constant Space

    for i in range(len(s)):
        index = ord(s[i]) - ord("a")
        count[index] += 1  # } N

    res = []  # Extra space
    for i in range(26):
        for j in range(count[i]):
            res.append(chr(ord("a") + i))  # } N

    return "".join(res)
```

Time = O(N)

Space → Immutable – O(N)

Space → Mutable – O(1)

Quiz 2

Total

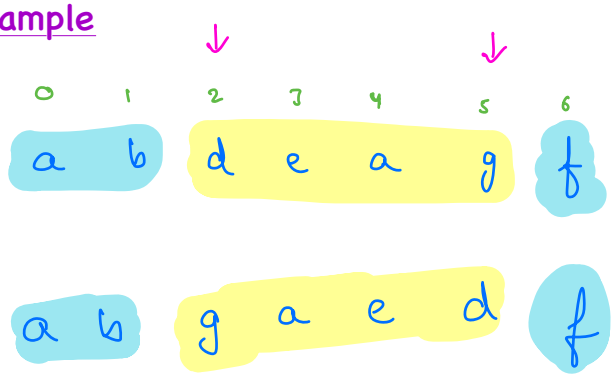
$|a| + |b| + |c| + |d| + \dots + |z| = N$?

Count Sort Algorithm

Break till 10:36 PM

Q3 Given a string s and two indices l & r.
Reverse the substring from l to r.

Example



l = 2
r = 5

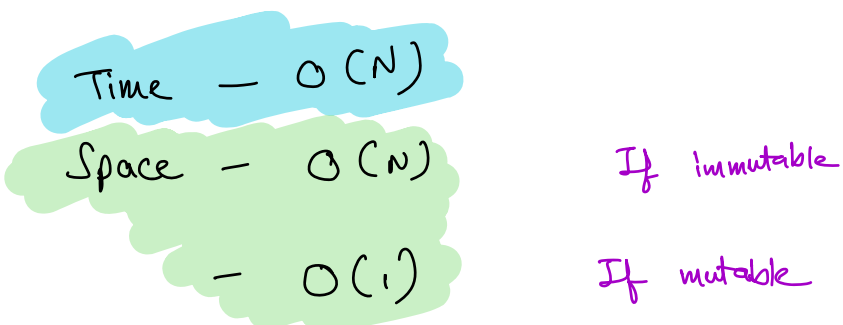
```
String reverseSubstring(String s, int l, int r) {
    while ( l < r ) {
        swap( s[l], s[r] )
        l++
        r--
    }
}
```

Java

```java
String reverseSubString(String s, int l, int r) {
    StringBuilder sb = new StringBuilder(s);
    while (l < r) {
        char temp = sb.charAt(l);
        sb.setCharAt(l, sb.charAt(r));
        sb.setCharAt(r, temp);
        l++;
        r--;
    }
    return sb.toString();
}
```

Python

```python
def reverseSubstring(s, l, r):
    res = list(s)

    while l < r:
        res[l], res[r] = res[r], res[l]
        l += 1
        r -= 1

    return "".join(res)
```



Q4 Given a character array, reverse it word by word

- No extra space is allowed
- Every word is separated by a single white space " "
- No inbuilt method

Example

s = "here is a string" → string a is here

h	e	r	e	" "	i	s	" "	a	" "	s	t	r	i	n	g

Quiz 3

s = "Are you as clever as I am"

am I as clever as you Are

Correct Option

A

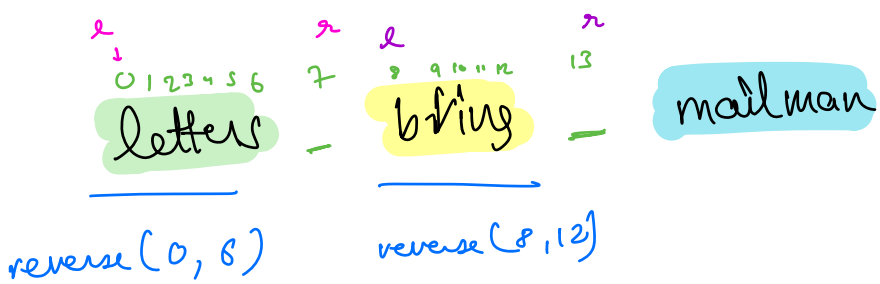
Approach

mailman brings letters

→ Reverse entire char array

srettel sgnirb namliam

→ Reverse each word



reverse (l, r-1)

l = r+1

TODO : write code on your own

Palindromes

madam

nitin

bob

racecar

wow

malayalam

pop

wow

radar

lol

Me: I'm scared of palindromes

Therapist: Wow

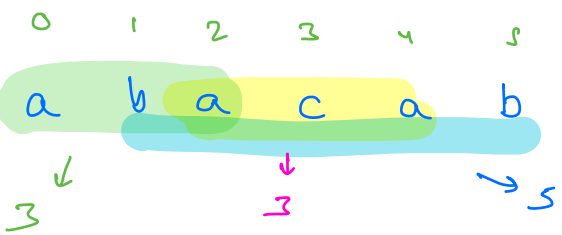
Me:

Aibohphobia is the fear of palindromes. A palindrome is a word that is spelt the same backwards as it is forward.

Q5 Longest Palindromic Substring

Given a string, calculate the length of longest palindromic substring.

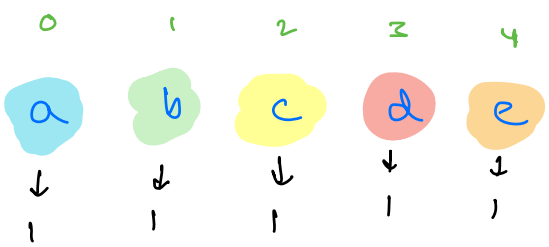
Example 1



Ans = 5

Example 2

Quiz 4



Ans = 1

Brute Force Idea

For every substring, check if it is palindrome or not & get the max length palindrome.

```
int lps(String s) {
        ans = 1
        for (l = 0;  l < N;  l++) {
            for(r = l ;  r < N;  r++) {
                if ( isPalindrome (s, l, r)) {
                    len = r - l + 1
                    ans = max (ans, len)
                }
            }
        }
        return ans
}
```

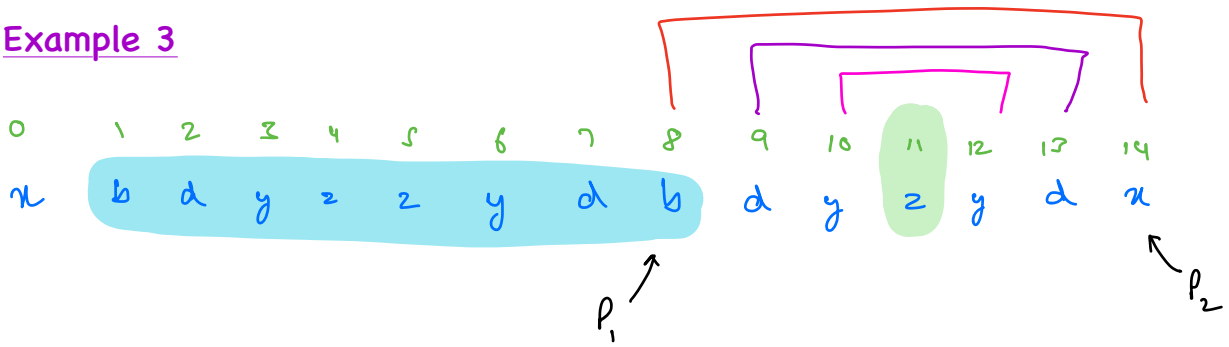
TODO

Time - $O(N^3)$

Space - $O(1)$

Optimised Approach

Example 3

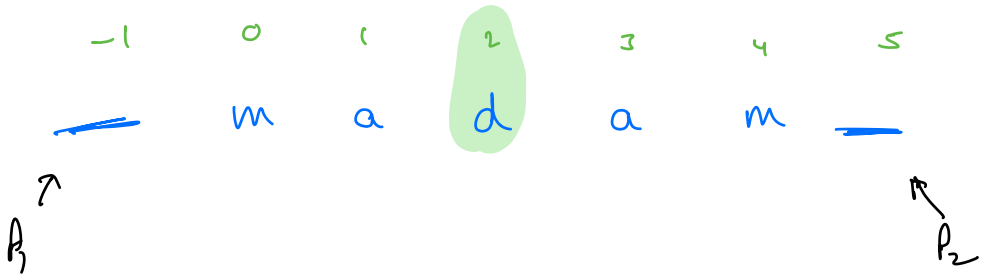


Given the center index of longest palindromic substring = 11

Palindrome − (P_1, P_2) Both excluded

= $P_2 - P_1 - 1$

Example 4



$P_1 >= 0$

$P_2 < N$

Center = 2

$$len = P_2 - P_1 - 1$$

$$= 5 - (-1) - 1$$

$$= 6 - 1 = 5$$

How to choose the center ?

Check for every element as the center.

```
// Function to find the length of longest palindrome, given the two pointers.
int palindromeLength(string str, int p1, int p2) {
        N = str.length
        while( p1 >= 0 and p2 < N and str[p1] == str[p2]) {
                p1--;
                p2++;
        }
        return p2 - p1 - 1
}
```

```
// Function to compute longest palindromic substring length
int lps(string s) {
        ans = 1
        for(i=0; i<N; i++) {
                // s[i] is center
                p1 = i, p2 = i
                ans = max(ans, palindromeLength(s, p1, p2))

                // s[i] and s[i+1] as centers
                p1 = i, p2 = i+1
```

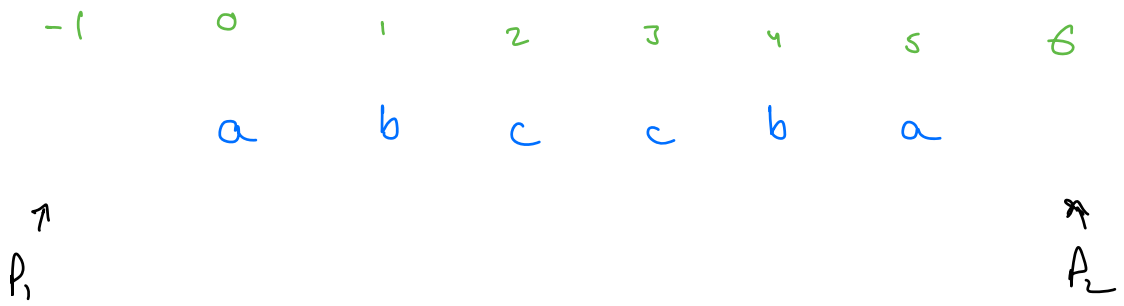
ans = max (ans, palindrome length (s, P_1, P_2))

}

return ans

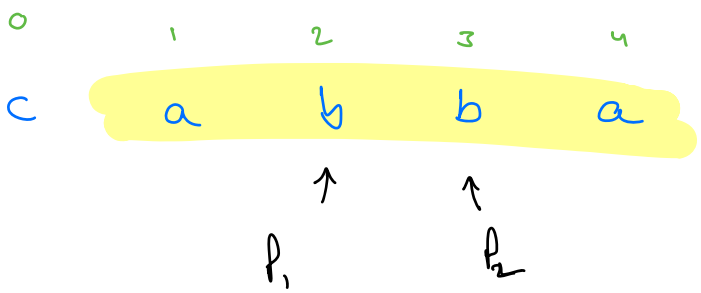
}

Example 5



$P_2 - P_1 - 1$

Example 6



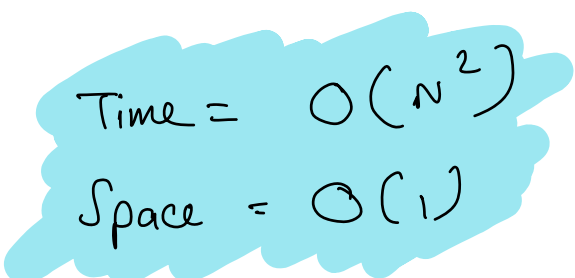
Ans = 4

Java

```java
int palindromeLength(String s, int p1, int p2) {
    while (p1 >= 0 && p2 < s.length() && s.charAt(p1) == s.charAt(p2)) {
        p1--;
        p2++;
    }
    return p2 - p1 - 1;
}

int lps(String s) {
    int ans = 0;

    for (int i = 0; i < s.length(); i++) {
        // s[i] is center
        int p1 = i, p2 = i;
        ans = Math.max(ans, palindromeLength(s, p1, p2));

        // s[i] and s[i+1] are center
        p1 = i;
        p2 = i + 1;
        ans = Math.max(ans, palindromeLength(s, p1, p2));
    }

    return ans;
}
```

Python

```python
def palindromeLength(s, p1, p2):
    while p1 >= 0 and p2 < len(s) and s[p1] == s[p2]:
        p1 -= 1
        p2 += 1
    return p2 - p1 - 1


def lps(s):
    ans = 0

    for i in range(len(s)):
        # s[i] is center
        p1 = i
        p2 = i
        ans = max(ans, palindromeLength(s, p1, p2))

        # s[i] and s[i+1] are center
        p1 = i
        p2 = i + 1
        ans = max(ans, palindromeLength(s, p1, p2))

    return ans
```



Doubts

Thank

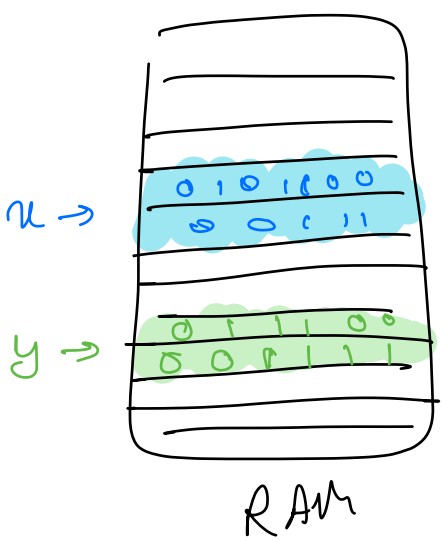
You

Hint for Magic Number

→ Check the notes

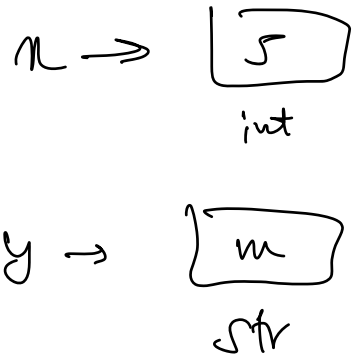
Largest Number → Comparator problem.

```
int x = 5
char y = 'm'
```



$x = 5$

$y = 'm'$



Good Night

Thank You

Monday