

## Today's Content:

- a) SubString Intn
- b) Palindrome or Not
- c) length of longest Palindromic Substring
- d) longest substring with all distinct characters
- e) Given a char[] reverse word by word

Substring: Concept of subarray on a string is called Substring.

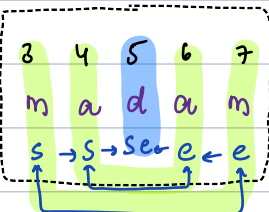
- Continuous Part of a string
- Full string is a substring
- Single character is also a substring

Palindrome:

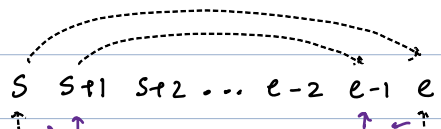
- |          |              |          |         |
|----------|--------------|----------|---------|
| a) madam | b) malayalam | c) civic | d) rada |
| YES      | YES          | YES      | No      |

Q1: Check if a given substring is Palindrome or Not?

0 1 2 3 4 5 6 7 8 9 10  
String str = a n a m a d a m s p e  
S = 3 l = 7 : Yes  
S = 5 l = 8 : No



```
bool ispal (String str, int s, int e) { // str = s s+1 s+2 ... e-2 e-1 e
    while (s < e) { // s < e is also fine
        if (str.charAt(s) == str.charAt(e)) {
            s = s+1, e = e-1;
        } else { return false; }
    }
    return true;
}
```



TC:  $O(N)$  SC:  $O(1)$

20) Given a String, return length of longest Palindromic Substrings<sup>1</sup>

Constraints:

$$1 \leq N \leq 10^3$$

Ex1:  $S = a \overset{0}{b} \overset{1}{a} \overset{2}{c} \overset{3}{a} \overset{4}{b}$  ans = 5

Ex2:  $S = a \overset{0}{n} \overset{1}{b} \overset{2}{c}$  ans = 1

Ex3:  $S = d \overset{0}{a} \overset{1}{b} \overset{2}{b} \overset{3}{a}$  ans = 4

Idea1: For every substring, check if it is palindrome or not?

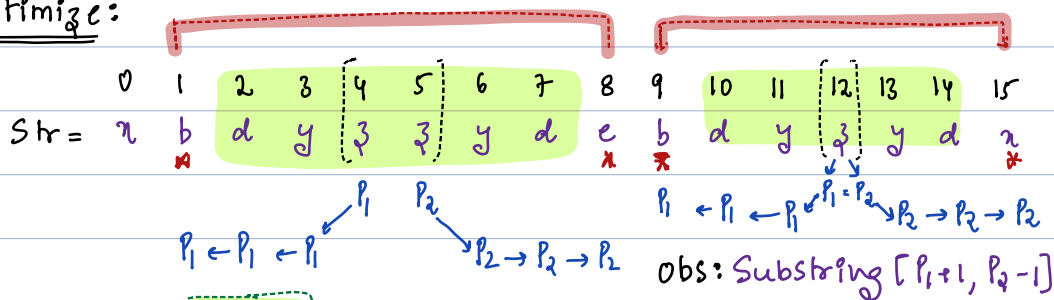
TC:  $O(N^2) \times O(N) \approx O(N^3)$  SC:  $O(1)$

↳  $\approx N^2$  substring

$\approx O(N)$  time to check if a substring is palindrome

```
int longPal(char ch[]) {  
    |  
    TODO:  
}
```

Optimize:



len:  $p_2 - p_1 - 1 = 8 - 1 - 1 = 6$

$b - a + 1$

Idea:

len:  $p_2 - 1 - (p_1 + 1) + 1 =$

$p_2 - 1 - p_1 - x + x = p_2 - p_1 - 1$

### 1. Odd length Palindromes

Take every character as center  
& calculate, longest palindromic  
length & consider overall max.

TC:  $O(N) * O(N) = O(N^2)$  SC:  $O(1)$

### 2. Even length Palindromes

Take every adjacent characters as center  
& calculate longest palindromic  
length & consider overall max.

TC:  $O(N) * O(N) = O(N^2)$  SC:  $O(1)$

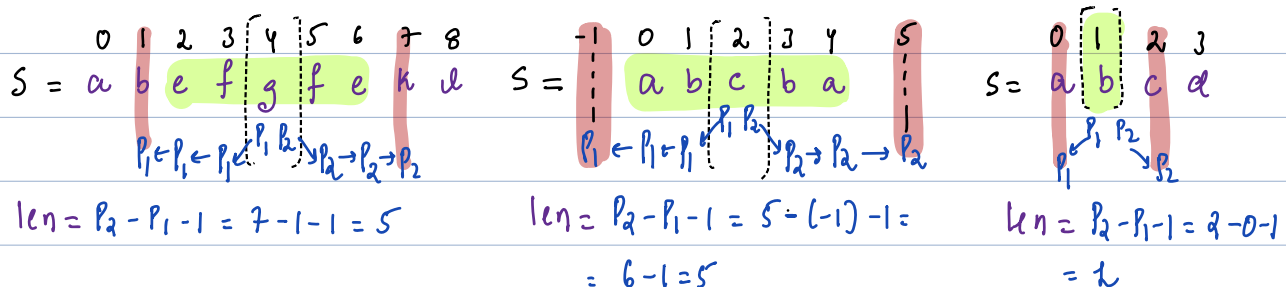
### 3. Final ans

: max of [even length Palindrome, Odd length Palindrome]

TC:  $O(N^2 + N^2) = O(N^2)$  SC:  $O(1)$

TODO: Self learning = manacher's =  $O(N)$   $O(N)$

Expand on Center?



int longestPal(String s) { Tc:  $O(N^2)$  Sc:  $O(1)$

int N = s.length();

int even = 0, odd = 0; // max even length & max odd length

for (int i = 0; i < N; i++) { // Max odd length

// Take i as center

int p1 = i, p2 = i;

0 1 2 3 . . . N-1 N  
p1 p2 p2

while (p1 >= 0 && p2 < N) {

if (s.charAt(p1) == s.charAt(p2)) {

p1--; p2++

else { break; }

int len = p2 - p1 - 1;

odd = max(odd, len);

for (int i = 0; i < N; i++) { // Max even length

// Take i & i+1 as centers

int p1 = i, p2 = i+1;

while (p1 >= 0 && p2 < N) {

if (s.charAt(p1) == s.charAt(p2)) {

p1--; p2++

else { break; }

10:20 break

int len = p2 - p1 - 1;

even = max(even, len);

return max(even, odd);

Note: Once take a small string & dry run & check for err?

28 length of longest substring with all distinct characters?

0 1 2 3 4 5 6 7  
 $S_1 = a b c a b c d d$  ans = 4

0 1 2 3 4 5 6 7  
 $S_2 = s i p p i e r g$  ans = 5

0 1 2 3 4  
 $S_3 = a a a a a$  ans = 1

Idea1: For all substrings, check if a substring contains all distinct &

TC:  $O(N^2) * O(N) = O(N^3)$  SC:  $O(N)$  get max length

↳ // Substrings

↳ // if a substring contains all distinct characters

Idea: Insert all character in hashset & check

Idea2: For every index i:

Calculate length of longest substring with all distinct char

& get overall max

0 1 2 3 4 5 6 7  
 $S = a b a c e a f e$  hs: c e a f  
len = 2 4 3 4 3 3 2 1 Final ans = 4

int longestSub(String s) { TC:  $O(N^2)$  SC:  $O(N)$

int N = s.length, ans = 0;

for (int i = 0; i < N; i++) { // For i: get length of longest substring distinct

HashSet<Char> hs;

for (int j = i; j < N; j++) {

if (hs.contains(s.charAt(j)) == false) {

hs.add(s.charAt(j))

} else { break }

ans = max(ans, hs.size());

}  
return ans;

Ideas:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14  
 Str = ~~a~~ ~~b~~ ~~c~~ ~~d~~ ~~e~~ ~~f~~ ~~g~~ ~~h~~ i m h a b k  
 i j

j: reach end: break

valid:

HashSet:

(0 0)

(0 1)

(0 2)

(0 3)

(0 4)

(0 5)

(1 1)

(1 2)

(1 3)

(1 4)

(1 5)

~~a~~ ~~b~~ ~~c~~ ~~d~~ ~~e~~ ~~f~~ ~~g~~ h i m  
 h a b k

skip them

(0 6)  $\xrightarrow[\text{S[0]}]{\text{HS*}}$  (1 6)  $\xrightarrow[\text{S[1]}]{\text{HS*}}$  (2 6)  $\xrightarrow[\text{S[2]}]{\text{HS*}}$  (3 6)  $\xrightarrow[\text{S[3]}]{\text{HS*}}$  (4 6)  $\rightarrow$  (4 7)  $\rightarrow$  (4 8)  $\rightarrow$  (4 9)  
 (4 9)  $\rightarrow$  (4 10)  $\rightarrow$  (5 10)  $\rightarrow$  (5 11)  $\rightarrow$  (5 12)  $\rightarrow$  (5 13)  $\rightarrow$  (6 13)  $\rightarrow$  (7 13)  
 (8 14)  $\leftarrow$  (8 13)  
 stop

int longestSubstr(String S) { TC: O(N) SC: O(N)

int N = S.length;

int ans = 0;

HashSet<char> hs;

int i = 0, j = 0;

while (j < N) {

if (hs.contains(S.charAt(j)) == false) {

hs.add(S.charAt(j));

ans = Math.max(ans, hs.size());

j++;

else {

hs.remove(S.charAt(i));

i++;

}

return ans;

}

To check if substring contains all distinct characters

38) Given a char ch[] reverse given substring?

Ex:

ch[] = { a n a c o n d a }

s = 2 e = 6

0 1 2 3 4 5 6 7

a n a c o n d a

d n o c a

{ a n d n o c a a }

```
void reverse(char ch[], int s, int e) {
```

```
    while(s <= e) {
```

```
        char t = ch[s]; // swap ch[s] with ch[e]
```

```
        ch[s] = ch[e];
```

```
        ch[e] = t;
```

```
        s++, e--;
```

```
    }
```

40) Given a char[] reverse word by word.

Note: a. No Extra Space at start and end

b. Every word Separated by single space

c. No usage of Inbuilt library

d. Expected TC: O(N) SC: O(1)

Ex: ch[] = love hate data codes

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Step 1: Reverse full = sedoc atad etah evol

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Step 2: Reverse each

word = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Final output: codes data hate love