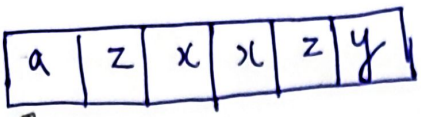
Lec3 Week5

1. Remove all adjacent duplicates in String.

(Leetcode)



ans

= "a" = "az" = "azx" = "az" = "ay"

ans ka last character

ans[ans.length()-1] == s[i]

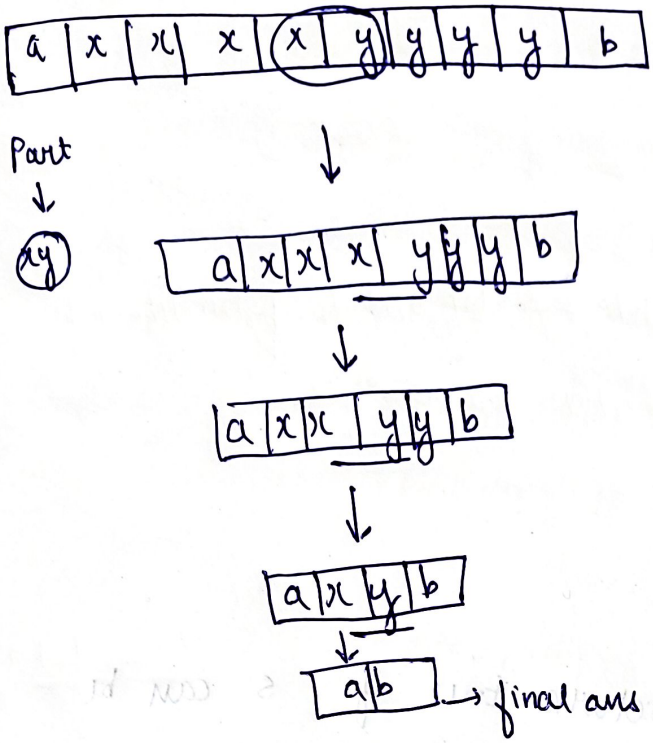
Code:

```
string removeDuplicates(string s){
    string ans = "";
    int i = 0;
    while(i < s.length()){
        if(ans[ans.length()-1] == s[i]){
        if(ans.length()-1>=0) && ans[ans.length()-1]==s[i]
            ans.pop_back();
        if(ans.length()>0){
            if(ans[ans.length()-1] == s[i]){
                ans.pop_back();
            }else{
                ans.push_back(s[i]);
            }
        }else{
            ans.push_back(s[i]);
        }
        i++;
    }
    return ans;
```

Very important

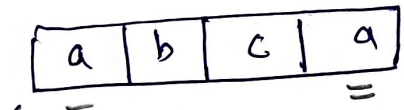
2. Remove all occurences of a substring.

10, 20 Marks



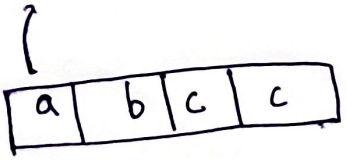
- HW: Implement s.find, erase.

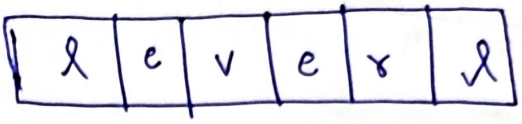
3. Given string s, return true if s can be palindrome after deleting at most one character from it.



s[i] == s[j] → no need to remove
↳ i++, j--

s[i] != s[j] → ya to i wala
ya to j wala remove





① already palindrome

↓

true

② after 1 removal

↓

true

③ >1 removal

↓

false

Case ①

s[i] == s[j]

↓

no removal needed on i & j

↓

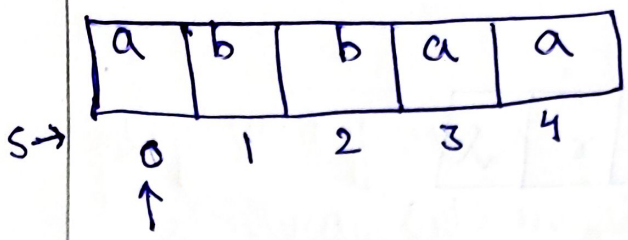
aage badha do

i++, j--

Case 2: (s[i] != s[j])

e remove
r remove
} → T

→ F (multiple removal)



0 removal → abb a a
elements

1 element remove → bb a a —check— F

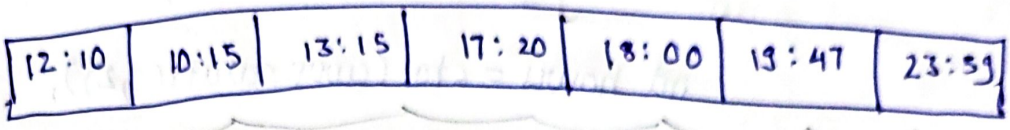
→ abaa — F
→ abaa → F
→ abba → T //

Code:

```
bool validPalindrome(string s){
    int i=0;
    int j = s.length() - 1;
    while (i<=j){
        if (s[i] != s[j]){
            // remove i once, remove j once
            return checkPalindrome(s, i+1, j) ||
                   checkPalindrom(s, i, j-1);
        }else{
            // s[i] == s[j]
```

Note: Inbuilt sort ka TC :- O(nlogn).
(Rotate)

4. To find minimum minute difference.
(Important)

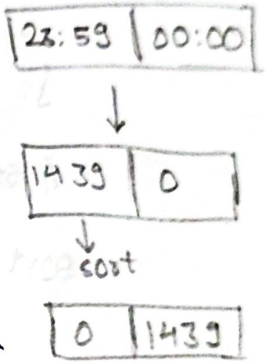


↓

To find minimum difference, b/w two data.

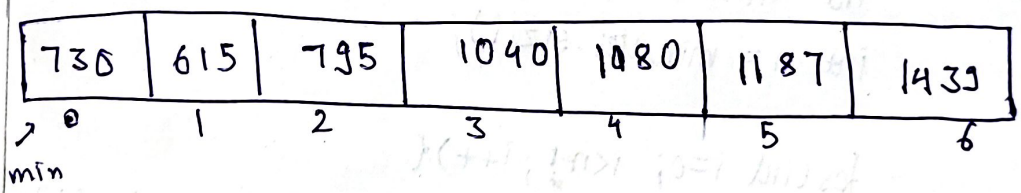
12:10 — 13:10
↓
1hr → 60 min

12:10 — 12:35
25min

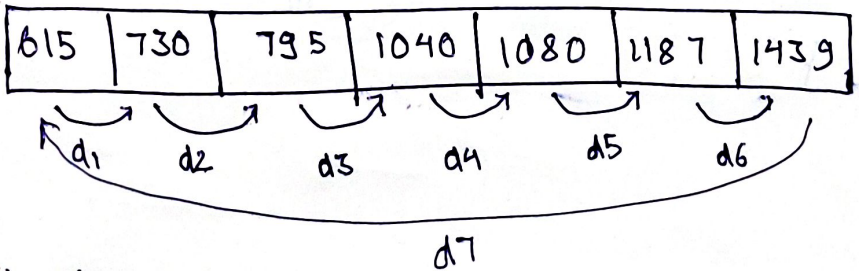


· Steps:

(i) Convert into minutes.



(ii) Sort: // only finding difference with adjacent element



23 26 1 / 13 50 / 54 / 430

(iii) difference

(iv) min

(v) To compare : 1439 to 615.
(d7)

```
int findMinDifference (
    vector<int> minutes;

    for(int i=0; i<timePoints.size(); i++){
        string curr= timePoints[i];
        int hours = stoi(curr.substr(0,2));
        int min.. = stoi(curr.substr (3, 2));
        int totalMinutes = hours * 60 + min..;
        minutes.push_back(totalMinutes);
    }

    // step 2 sort
    sort(minutes.begin(), minutes.end());

    // step 3 difference and calculate min diff
    int mini = INT_MAX;
    int n = minutes.size();

    for(int i=0; i<n-1; i++){
        int diff = minutes[i+1] - minutes[i];
        mini = min(mini, diff);
    }
```

Note: To convert string to int

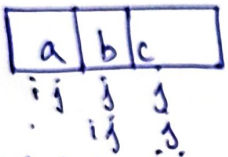
→ Use 'stoi'

// something missing

```
int lastDiff = (minutes[0] + 1440) - minutes[n-1];
mini = min(mini, diff);
return mini;
}
```

Ques 5: (Given string, return number of palindromic substring in it.)

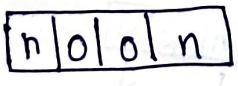
→ Number of palindromic substrings.



substring if → (is continuous)

a b c
ab bc
abc

```
for (in → 0 → <n)
{
    for (j → i → <n)
}
```
→ To print substring



substring:-

n✓ o o n✓
nox oo✓ onx
noox oonx
noon✓

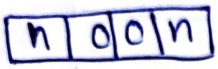
n
noon
o
oo

Approach #1

① find all substring

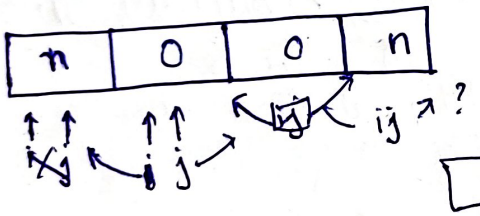
② check palindrome and count

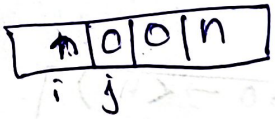
→ T.C = $O(n^3)$



substring

- odd length substring
- even length substring





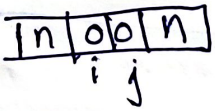
Count → 0.1

Odd

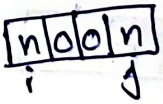
- n
- o
- noo ✗
- o
- oon ✗
- n

Even

(i) no ✗

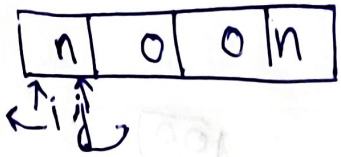


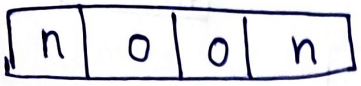
(ii) oo ✓

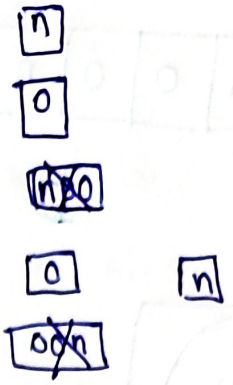


(iii) noon ✓

(iv) on ✗

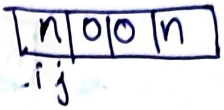




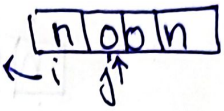


odd len substring

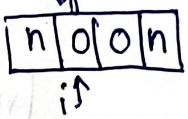
- n -
- o -
- noo -
- oon -



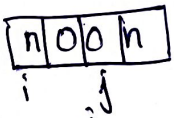
count = 0



count = 1

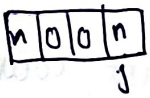


count = 2



count = 3

count = 4



Match case

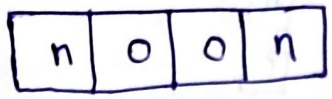
↓

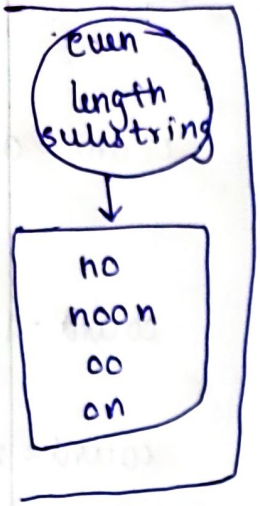
- count increment
- i--
- j++

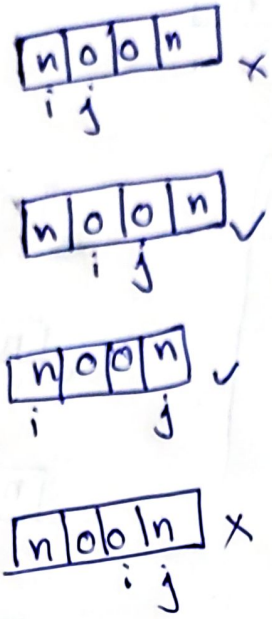
No match case

↓

- go to next case







Code:-

```
int countSubstrings (string s) {
        int count = 0
        int n = s.length();
        for ( int i = 0; i<n; i++) {
            // odd.
            int oddAns = expandAroundIndex(s, i, i);
            count = count + oddAns;

            //even
            int evenAns = expandAroundIndex(s, i, i+1);
            count + = evenAns;

        }
        return count;

}
```

Now, Writing expandingAroundIndex function:-

```
int expandAroundIndex(string s, int i, int j){
    int count=0;
    while(i>=0 && j<s.length() && s[i]==s[j]){
        count++;
        i--;
        j++;
    }
    return count;
}
```