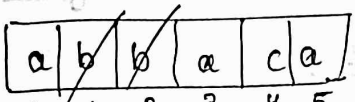
Week 6

# Char-Arrays & Strings - L2

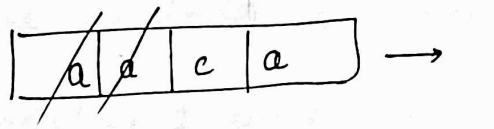
## Q. Remove all adjacent duplicates in String.

i/p :- string s = 'abbaca' 'abbaca'  o/p :- 'ca'

Eg :-

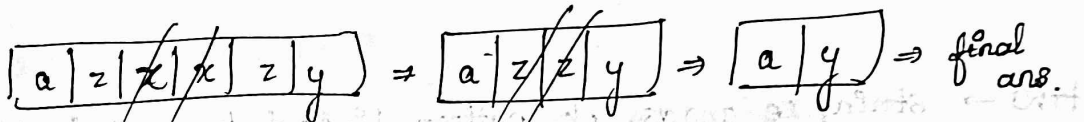



→



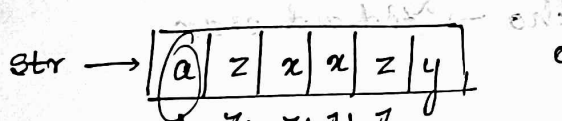

⇒ final ans.

Eg:



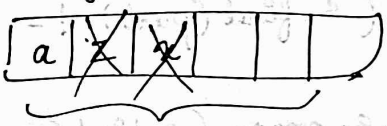

⇒ final ans.

* Approach → new string ← nayi string ke andar answer build karo.




create empty string.

string temp = "";




nightmost = a

a ≠ z → insert z

z ≠ x → insert x

x = x → x delete

z = nightmost = z → z del

y ≠ a ? → insert y.

nightmost = a

x ≠ z, ∴ insert z

z ≠ x

different ⇒ push

same ⇒ pop

∴ a y

SC → O(n)

TC ⇒ O(n)

Q> Remove all Occurance of a Substring.

str → "abbcbadddbabbbaae"

part → "cba"

↳ o/p:- 'abbddbbbaae'

* This can be called as substring. and we can use find function To remove, we can use erase.

∴ while (substring exist in badi string)
{
remove;
}

HW → string ke andar ek pattern findout karna hai, toh uske liye kon kon se Algorithms exist karti hai? with TC.
↳ (named algorithms)

* Functions ko khud se likho → find and erase

```
while (s.find(part) != string::npos)
{
    s.erase(s.find(part), part.length());
}
```

## Q) Valid Palindrome - II

→ Given a string s, return true, if the s can be palindrome after deleting the <u>atmost one</u> character from it.

input → [string] → 0 or 1 removal

- palindrome (true)
- not a palindrome (false)

Q) 'aba' → already palindrome → true

Q) 'abca' → 1 remove → Palindrome → true

Q) 'abc' → a → 'bc'
b → 'ac'
c → 'ab'
X → 'abc'
} False, no palindrome.

Eg:- 'MADMAM'

| M | A | D | M | A | M |
|---|---|---|---|---|---|
| ✓ | ✓ | ✗ | ✗ | ✓ | ✓ |

equal → s++, e--;

not equal → 1 removal → remaining palindrome

Eg:-

| M | A | D | A | R | R | A | C | A | M |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 = j |

i = 0

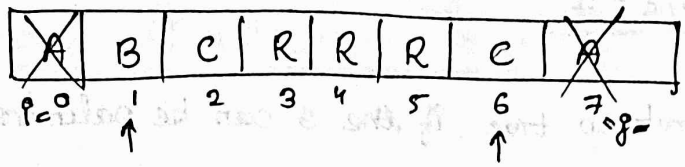
s[0] == s[9] → true → i++; j--;

s[1] == s[8] → true → i++, j--;

s[2] != s[7] → false → only 1 allowed. (have 2 options.)

- if D: A R R A C → X no
- if C: D A R R A → X no

→ Exit → Overall ans = false;




if S[0] == S[7] → true → i++, j++

S[1] == S[6] → false

1 removal

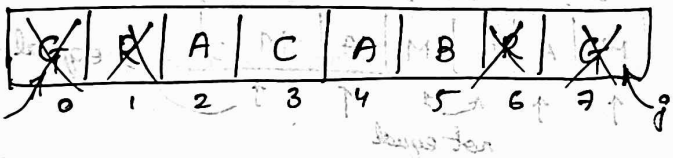
remove B → CRRRC (true) ✓

remove C → BCRRR (false)

agar ek bhi true aayega, toh overall ans true → OR

∴ inn dono answers ko 'OR' karo, then get ans.

| | |
|---|---|
| 0 | 0 → 0 |
| 0 | 1 → 1 |
| 1 | 0 → 1 |
| 1 | 1 → 1 |




S[0] == S[7] → true → i++, j--

S[1] == S[6] → true → i++, j--

S[2] == S[5] → false

ith char → remaining → (i+1 se j) → CAB → false

jth char → remaining → (i se j-1) → ACA → true

false || true

true

# Q) 647. Palindromic Substrings

i/p → "aba"

count → kitne substring palindrome hai?

substring

a → ✓
aba → ✓
ab → ×
ba → ×
a → ✓
b → ✓

} ans = 4.

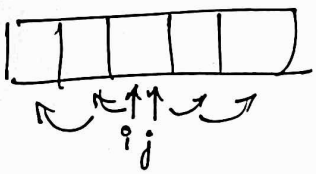
**Approach 1 :-** Bachha jo sabhi substring nikalega, phir check palendrome.

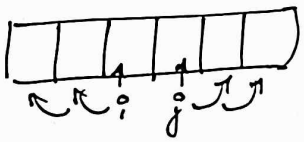
↳ Bruteforce se selection nahi hoga !!!

⇒

**Approach 2 :-** 2 pointer, but different one.

palindrome
length
length
Odd substring
Even substring




if (match)
↳ count++

else
↳ no need to compare further

match

Eg :- a | b | c

odd ✓

X ↳ stop.

count = 0
count++
⇒ 1

a = a
i == j

Eg :- a | b | c

i j

count = 1
if (i == j)
c++
count = 1 + 1 = 2

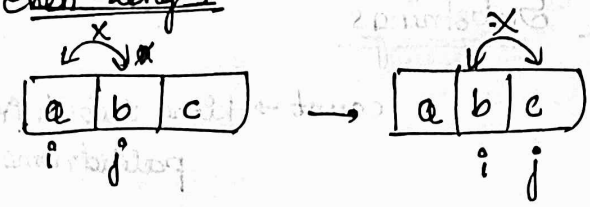
then is a == c → false
nik jao

a | b | c
0 1 2
i j

X

i == j
↳ c++
count = 3

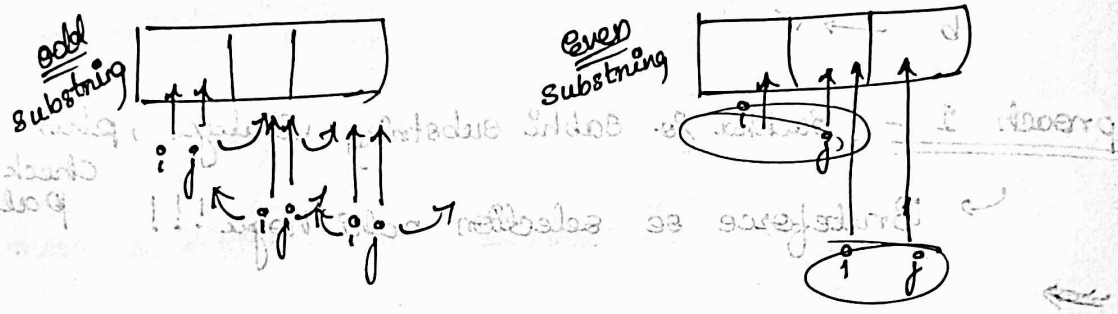
Eg:- Even length




∴ count = 0

Hence, total count = 3 + 0 = 3 //

* * *




---

Code :-

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

int countSubstrings (string s)
{
  int totalCount = 0;
  for (int center = 0; centre < s.length(); centre++)
  {
    int oddAns = expand (s, centre, centre);
    int evenAns = expand (s, centre, centre+1);
    totalCount = totalCount + oddAns + evenAns;
  }
  return totalCount;
}
```

| M | A | D | A | M |
|---|---|---|---|---|

centre = 0 → oddAns = expand (s, 0, 0);

| M | A | D | A | M |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |

count = 0

s[i] == s[j] → (M) ⇒ oddcount = 1, i = 0, j = 4

evenAns = expand (s, 0, 1);

count = 0,

s[0] == s[1] → X → even count = 0

∴ total = 0 + 1 + 0 → total = 1

centre = 1 → oddAns = expand (s, 1, 1) → oddcount = 1 (A)

~~even Ans~~

i = 0, j = 2

↓

s[i] == s[j] → M ≠ D.

evenAns = expand (s, 1, 2)

↳ X

total = 1 + 1 ⇒ total = 2

centre = 2 → oddAns = expand (s, 2, 2) → oddcount = 1 (D)

now, i = 1, j = 3

↳ s[i] == s[j] → oddcount = 2 (ADA)

now, i = 0, j = 4

↳ s[i] == s[j] → oddcount = 3 (MADAM)

evenAns = expand (s, 2, 3)

↳ s[i] ≠ s[j] → even = 0.

∴ total = (2) + 3 → total = 5

centre = 3

centre = 3

| M | A | D | A | M |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |

odd Ans = expand (s, 3, 3) ⟶ odd Ans = 1.

even Ans = expand (s, 3, 4) ⟶ 0

∴ total = 5 + 1 ⇒ 6

centre = 4

odd Ans = expand (s, 4, 4) ⟶ odd Ans = 1

even Ans = expand (s, 4, 5) ⟶ 0

∴ total = 6 + 1 ⟶ total = 7