Strings

Monday, 13 December 2021 9:04 PM

"abcd"

s = "abc" ←
s += 'd'; ✓ CPP TC = O(1) Java TC = O(len of s) ✓

Q → Given a character array as a line of words.
Rippling
Reverse the line word by word without extra space.

s = "the sky is blue"
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

reversed → "eulb si yks eht"
TC = O(N)
SC = O(1)
0 1 2 - -

reverse word by word → "blue_is_sky_the"
TC = O(N)
SC = O(1)

Ans → "blue is sky the"

TC = O(N) SC = O(1)

Q → Boring Substring → length 2 → both character's are consecutive ✓ wrt english alphabets. Eg → ab, ba, xy, yx, st, ts, cd, dc..

[c → (c+1) or (c−1)]

(I/P → only lower case characters.)

Given a char array check if it is possible to rearrange the characters s.t there is no boring substring in the string/char array.

Eg → "abcd"
0 1 2 3
→ "cadb" ✓
→ "bdac" ← reverse ✓
Ans = Yes/True ✓

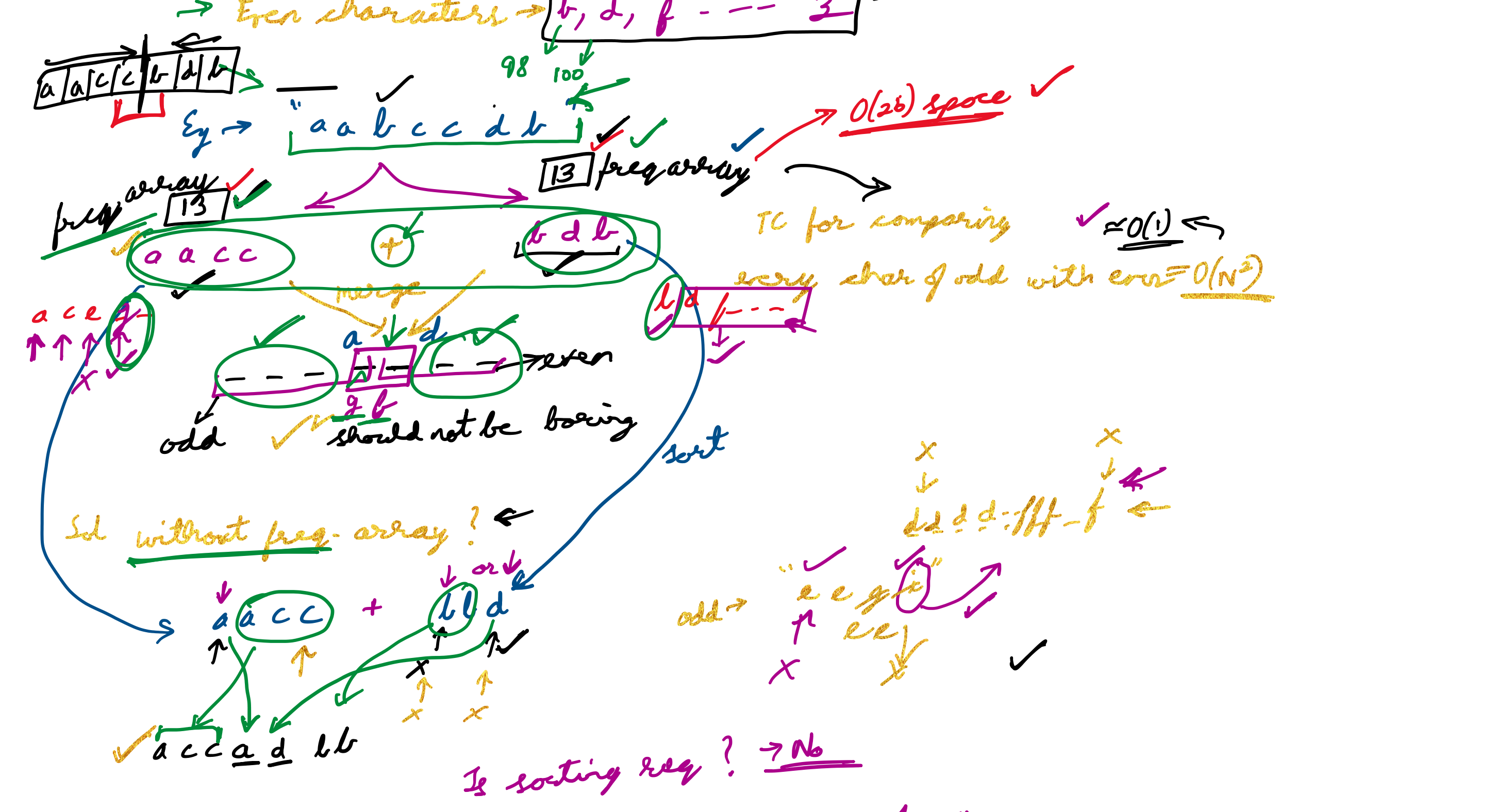
cadb
↓ reverse
bdac

Brute → check all permutations. TC = O(N!)

Observation → 1) If a rearrangement is possible then it's reverse is also valid.

2) "aab" Ans = No/False ✓

3) → Odd characters → a, c, e - - - y (97 99 101) → can never form boring substring
→ Even characters → b, d, f - - - z (98 100)



O(26) space ✓

TC for comparing every char of odd with even = O(N²) ✓ ≈ O(1)

Is sorting req? → No
∵ we just need min & max of both.

Sol → I/P → get min odd, max odd, min even & max even char.
If it is possible to get a non-boring pair return true.
else return false. TC = O(N) SC = O(1) ✓

Eg → "aabccdb" → min odd = 'a', max odd = 'c', min even = 'b', max even = 'd'

c = 'a' → (int) c % 2 == 0 / == 1
→ (c − 'a') % 2 == 0 or 1

ad ✓

ASCII 97 97 98 99 100 98
a a b c c d b

Q → Find the smallest period of a string/char array.
→ K if ∀i s[i] = s[i % K]

Eg → "abca abca abc" K = 4
0 1 2 3 4 5 6 7 8 9 10

s[i] = s[i % 4] ∀i ✓
7 7 % 4 = 3
2 2 % 4 = 2
5 5 % 4 = 1

"aaaa" K = 1

largest K = N
Smallest K = 1
1 <= K <= N

"abcd" K = 4
0 1 2 3
s[i] = s[i % 4] ✓
i < 4

Brute → ∀K check if it is period of a string. TC = O(N²)
O(N) O(N) → ∀i is s[i] == s[i % K]

Binary Search on K
0 1 2 3 4 5 6 7 8 9 10
a b c a a b c a a b c

X 1 — N
mid = 5 Is K = 5 possible? → No ✓ → can't decide to go left or right. ✓
mid = 3 If K = 3 possible? → No ✓

Z[i] = length of longest substring of s array starting from index i ✓ which is also a prefix (substring starting from index 0).
→ TC = O(N) ✓ SC = O(N) ✓

s = a b c a | a b c a | a b c
Z = − 0 0 1 | 4 0 0 1 | 3 0 0
array

∀i → Z[i % K] == Z[i]

TC = O(N)

Use Z function to calculate period of a string (K).

0 1 2 3 | 4 5 6 7 | 8 9 10 11 | 12 13 | 14 15 | 16 17
a b c a | a b c a | a b c a | a b | c a | a b
Z → 1 4 1 2 2 3 3 4

□ = ▱

(i + Z[i] == N)
↓
period of string

K = N ✓

K = 4
"abcd" K = 4
∀i s[i] == s[i % 4] ✓

Z Function

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
x	x	y	a	x	x	y	a	z	x	x	y	a	x	x	y	a	x	z
−	1	0	0	4	1	0	0	0	8	1	0	0	5	1	0	0	1	0

prefix

s[i] == s[j] ✓ len = 1 2 3 4 5 6 7 8 ✗

cur
Z[cur − l]
10 Z[10 − 9]
= Z[1] = 1
10 + 1 <= 16 ✓

1) Bruteforce (2 Pointer) index 0, current location
2) if (cur + Z[cur − l] <= r) → 16
13 Z[cur] = Z[cur − l];
4
13 + 4 > 16
3) if (cur + Z[cur − l] > r)
start comparing from r+1;

```
✓ int l=0, r=0;
for(cur = 1; cur < n; cur++) {
    if(cur > r) { // 1
        l = r = cur;
        while(r < n && s[r-l] == s[r]) r++;
        z[cur] = r-l; r--;
    } else {
        ✓ if( cur + z[cur-l] <= r)  z[cur] = z[cur-l];  //2
        else {   //3
            l=cur;
            while( r < n && s[r-l] == s[r] ) r++; ✓
            z[cur] = r-l;  r--;
        }
    }
}
```

TC = O(N)

Z[9] = 17 − 9 = 8
Z[13] = 18 − 13 = 5
Z[17 − 13] = Z[4]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
x	x	y	a	x	x	y	a	z	x	x	y	a	x	x	y	a	x	z
	1	0	0	4	1	0	0	0	8	1	0	0	5	1	0	0	1	0

13 + Z[13 − 9] = 13 + 4 = 17 ≤ r ✗ (16)
10 + Z[10 − 9] = 10 + 1 = 11 <= r ✓

pattern = "abc" ✓ s = "axbcacabcxza"

Find pattern in s?

abc # a x b c a c a b c x z a
Z → − 0 0 0 1 0 0 0 1 0 3 0 0 0 0 1

If Z[i] == len(pattern) ✓