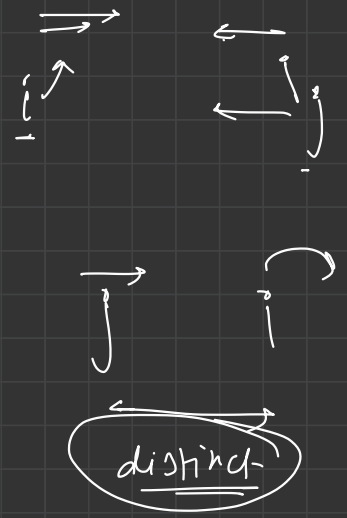
two-pointers

- i → left(0) j → right(n-1)
- i → left(0) j → right(0)



Frequency Array

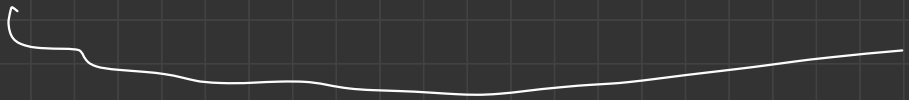
↳ shift

a → ?
b → ?
c → ?
d
⋮

Prefix / Suffix / Cummulative

Values before or after

Sorting and then searching



Strings with all above approach

$n = 6$

$s =$ a a c a b a

$i =$ 0 1 2 3 4 5 valid??

→ how many ways can we split?

we can split after an index i, such that $s_{left} = [0, i]$

$s_{right} = [i+1, n-1]$

can we split for $i = n-1$??

$s_{left} = (0, n-1)$ $s_{right} = (n, n-1)$

split at n-2 ??

$S_{left} = (0, n-2)$ ✓ $S_{right} = (n-1, n-1)$

we can split at any index $i \in [0, n-2]$

loop $i \rightarrow 0 \rightarrow n-2$

- check no. of distinct (loop??) elements from $0 \rightarrow i$??
- check no. of distinct elements from $i+1 \rightarrow n-1$??
- if equal → include in output ✓

a a c a b a

0 n-1

int checkDistinctElement (int l, int r, String s) {

int freq[26] = {0} to store freq of chars (a→z).

```
for (i = l; i <= r; i++){
    freq[s[i] - 'a']++;
}
int count = 0;
for(int i=0; i<26; i++){
    if(freq[i] > 0){
        count++;
    }
}
```

freq

a → 4

b → 1

c → 2

d : } → 0

z

```
}
return count;
}
```

{distinct chars after this index.}

```
if(distinctInRange(0, i, s) == distinctInRange(i+1, n-1, s)){
```

prefix

cummulative

⇒ no of chars distinct upto that index

0 → 0
0 → 1
0 → 2
0 → 3
⋮
0 → n-1

sum of all elements upto that index

	a	a	c	a	b	a
i	0	1	2	3	4	5
distinct upto	1	1	2	2	3	3

distinct = at each index no. of chars from 0 to i

	3	3	2	2	1	0

distinct after

distinct = 0

index ke baad

freq

a →	T
b →	T
c →	T

```
i: 0 → 5
if ( freq[s[i]] == F) {
    distinct++;
    freq[s[i]] = true
}
→ distinctUpto[i] = distinct;
```

i = 0 dist = 3

s[i] = a

freq[a] → 0/F

distUpto

1	1	2	2	3	3
0	1	2	3	4	5

a a c a b a

i = 1; s[i] = a ; freq[a] → T

i = 2; s[i] = c ; freq[c] → F

i = 3; s[i] = a ; freq[a] → T

i = 4; ⇒ b

i = 5; ⇒ a ⇒ freq[a] → T

n = 6

n-2 = 4

freq	
a →	T
b →	T
c →	T

distinctAfter

3	3	2	2	1	0
0	1	2	3	4	5

i == n-2 → 0

0 1 2 3 4 5

a a c a b a.

i=4; ⇒ s[i+1] → s[5] ⇒ a

freq[a] → F

i=3; ⇒ s[i+1] → s[4] ⇒ b

freq[b] → F

i=2; ⇒ s[i+1] → s[3] ⇒ a.

freq[a] → T

i=1; ⇒ s[i+1] → s[2] ⇒ c

freq[c] → F

Prefix

dist=3

// if (freq[s[i+1] == F) {

distinct++

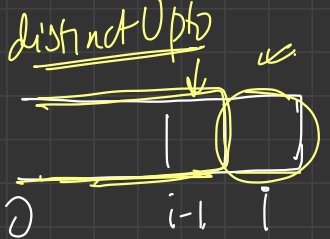
freq[s[i+1]] = T;

}

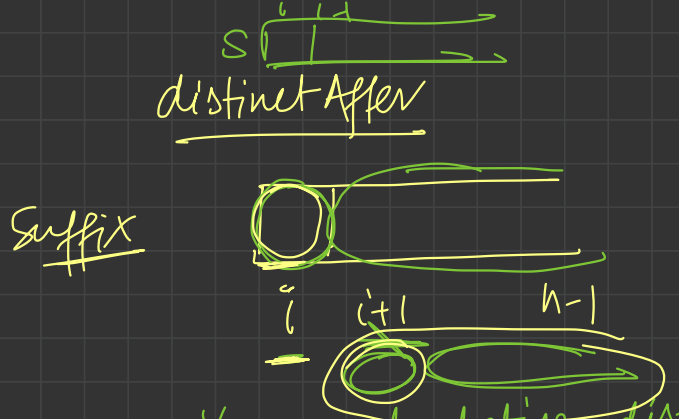
distinctAfter[i] = distinct;

i=0; s[i+1] → s[1] → 'a'

freq[a] → T

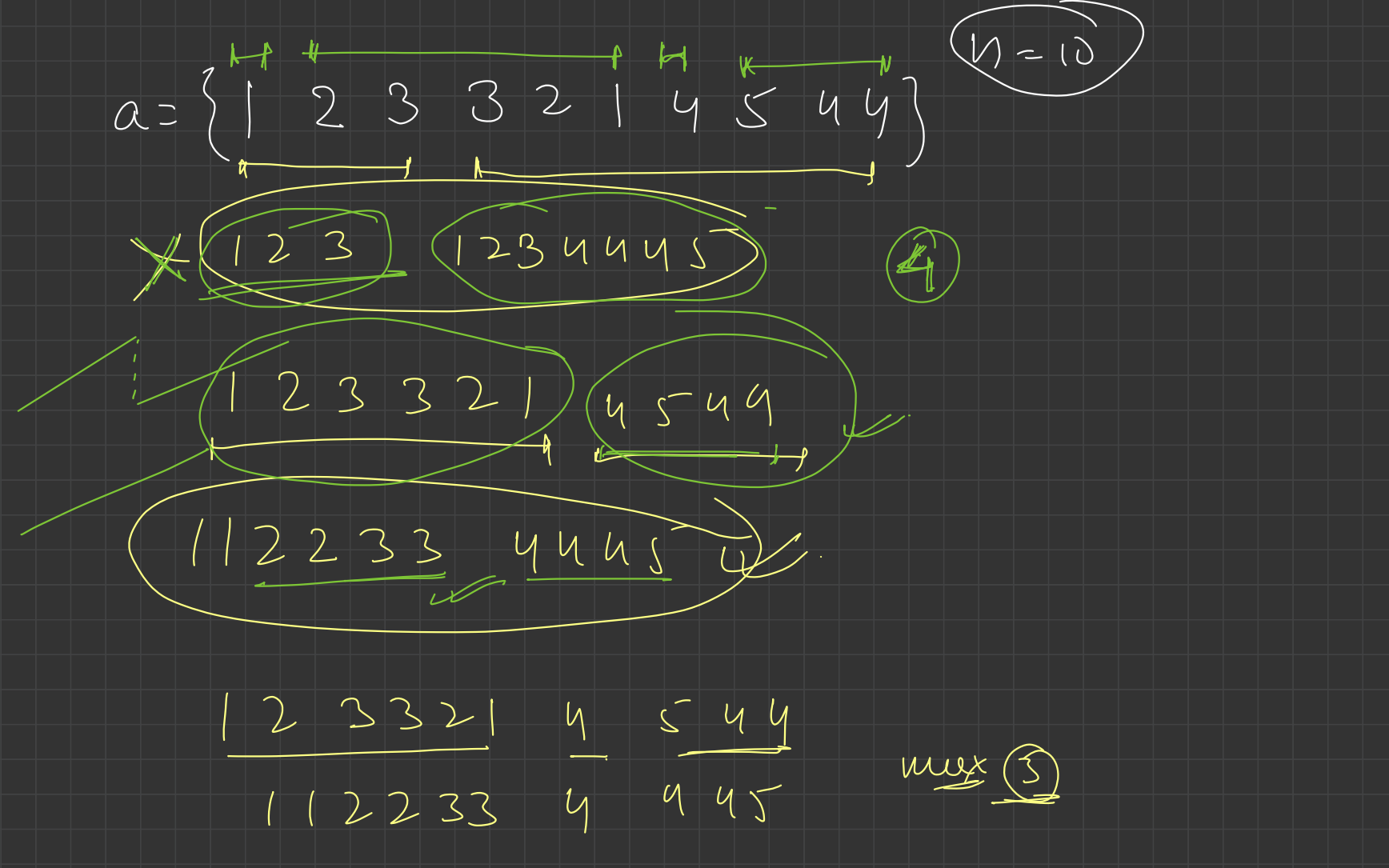


↙ we only need to consider s[i]



when calculating distinctAfter for ith index

we need to consider s[i+1]

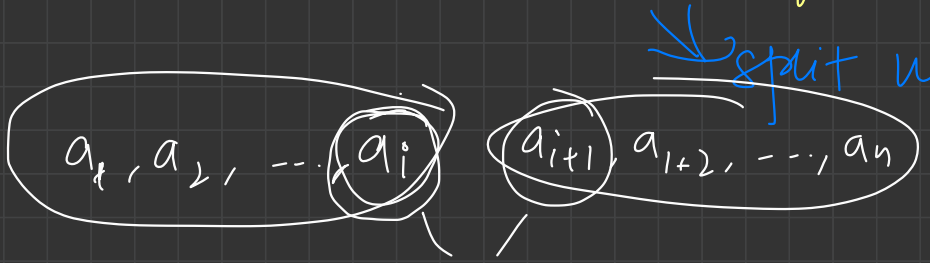

n = 10
a = { 1 2 3 3 2 1 4 5 4 4 }
1 2 3
1 2 3 4 4 4 5
4
1 2 3 3 2 1
4 5 4 4
1 1 2 2 3 3 4 4 4 5
1 2 3 3 2 1 4 5 4 4
1 1 2 2 3 3 4 4 4 5
max 3

1 2 3 3 2 1 4 5 4 4

1, 2, 1, 2, 3, 3, 4, 4, 4, 5

X split not allowed

when the left part has a value which is greater than some element in the right



3 <= 4 not in correct order ?

max (3) min (4)

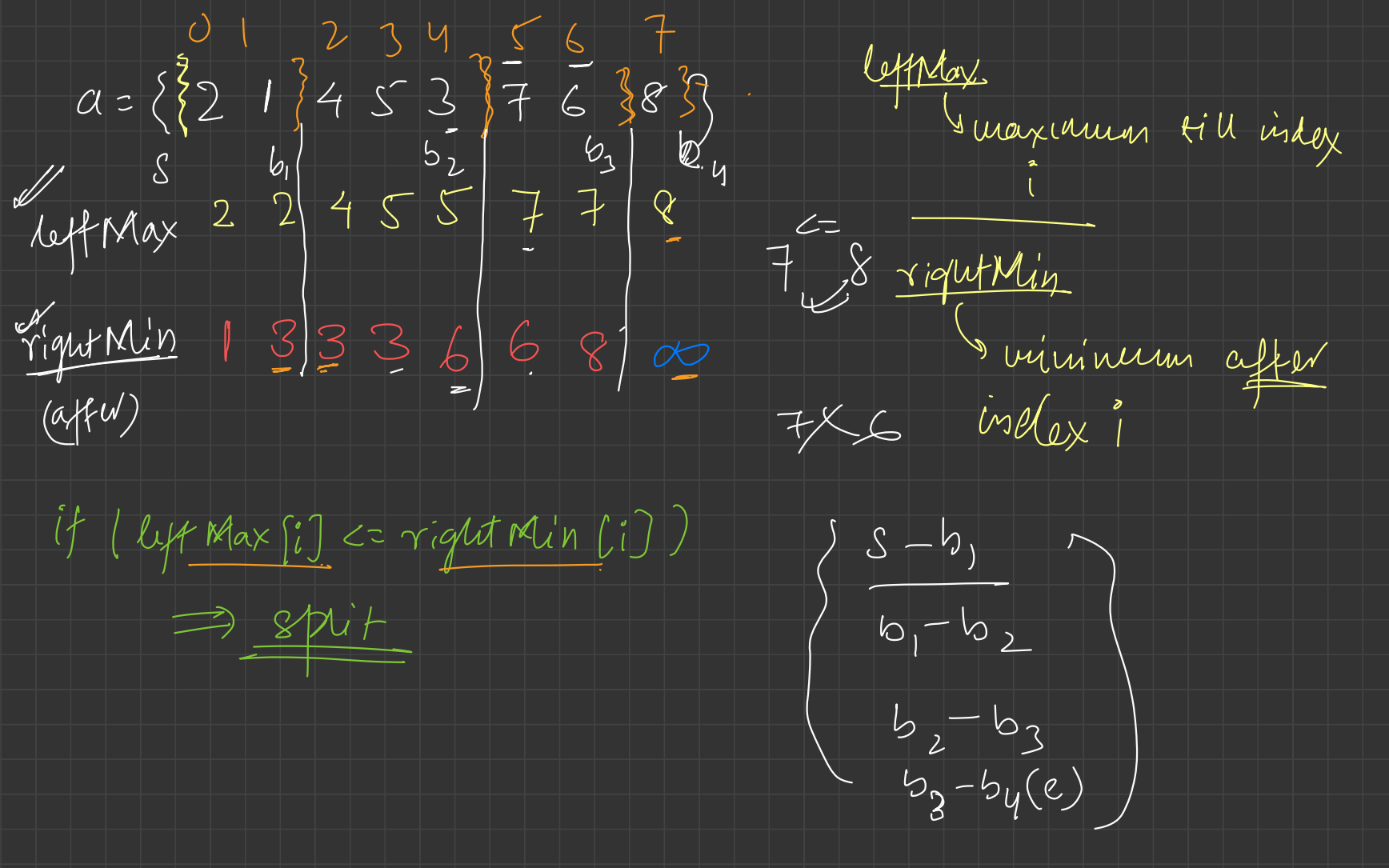
a = { 3 1 2 } 6 4 5

1 2 3 4 5 6

(3) 1, 2, 4, 5, 6

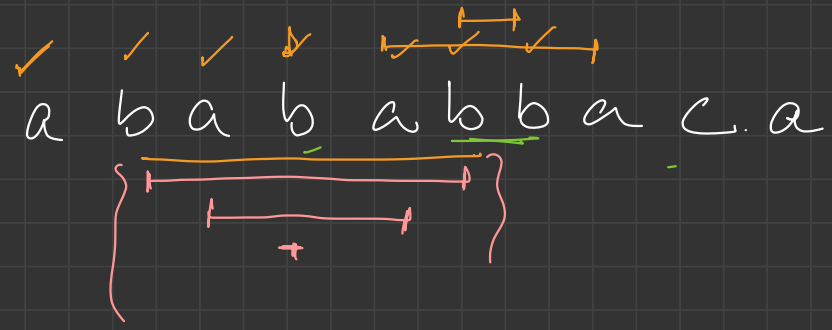
1, 3, 2, 4, 5, 6

⇒ if we want to partition the part of the array then the necessary condn is max of left <= min of right

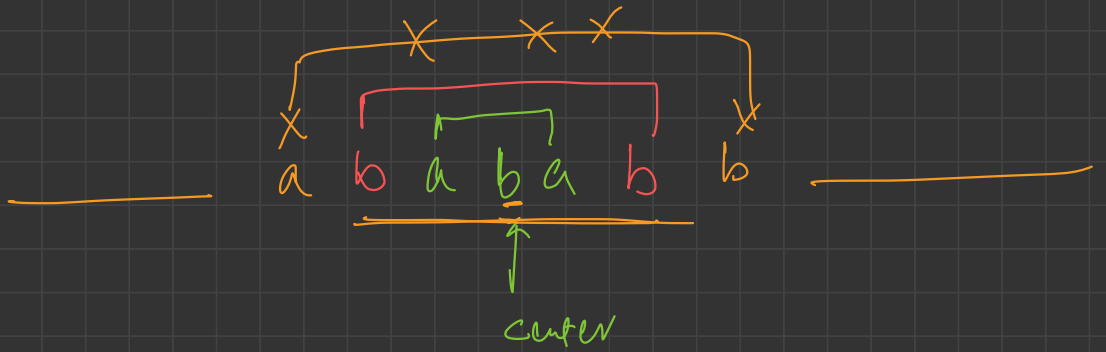

0 1 2 3 4 5 6 7
a = { 2 1 | 4 5 3 | 7 6 | 8 9 }
s b1 b2 b3 b4
leftMax 2 2 4 5 5 7 7 8
rightMin 1 3 3 3 6 6 8 ∞
(after)
leftMax
maximum till index i
<=
7 8
rightMin
minimum after index i
7 X 6
if (leftMax[i] <= rightMin[i])
⇒ split
s − b1
b1 − b2
b2 − b3
b3 − b4 (e)

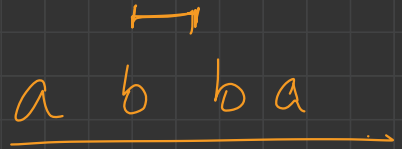
(5)

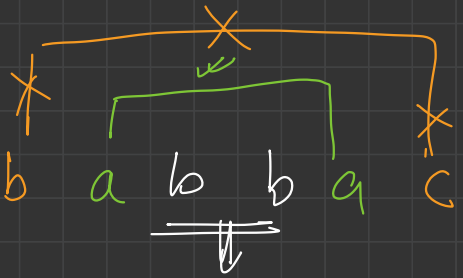
$n = 10$



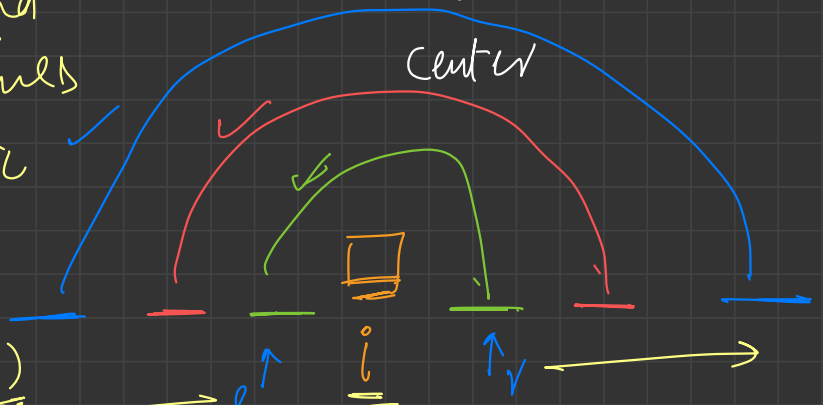
⇒ is centered in a way such that moving out from the centre gives same element at each step







time to find all palindromes for a specific centre

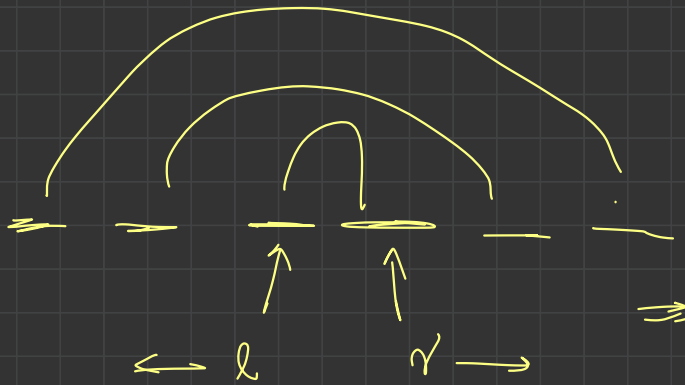


$O(n/2) = O(n)$

odd length palindrome : center → 1 elem

no. of centers possible → (n)

$O(n) \times O(n) \Rightarrow O(n^2)$



⇒ even length pal.

time for pal. check → $O(n)$

no of centers : $n-1$ → $O(n)$

$\Rightarrow O(n^2)$