Agenda

- Longest Substring Without Repeat
- First Non Repeating Element
- Check if there exits a subarray with sum = 0
- Check if there exits a subarray with sum = K

Q. a b c a b c b b

abc
bca
cab
abc
} 3

b b b b b = 1

p w w k e w = 3

Brute force:

generate all substrings $O(N^2)$

$O(N^3)$

↳ check if all characters are unique

$O(N)$ ↳ Keep track of length of longest

substring

5 mins

4

a b c d b b c

s ..... s

expand till repeat happens

b

a
b
c
d

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

a b c d e a f g h i b j k l m n

s e

i
f
a
g k
c l
d m
e n
a
g
h
b

biggest possible length — index e → repeating char

e - s

16 - 2 = 14

start = 0

end = 0

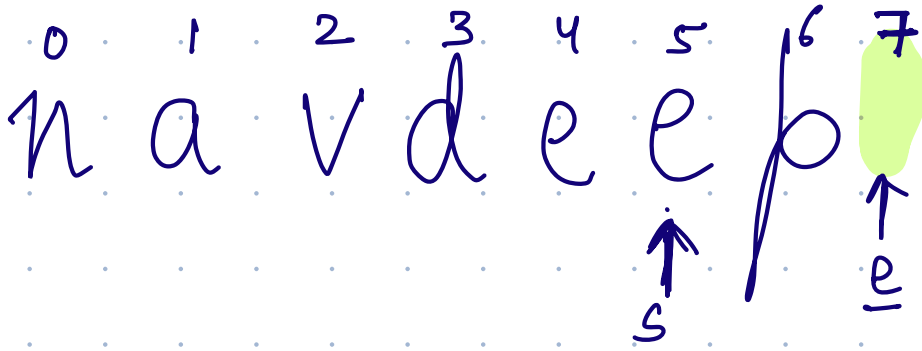
maxL = 0

hashset = empty set

str

```
while ( end < N ) {
        end < N &&
    while (hashset.search
                (str.charAt(end)) ==
                                false) {
        hashset.add(str.charAt(end))
        end++
    }
    [ mylen = end - start .
    if (mylen > maxL) {maxL = mylen}
    if (end == N) break;
    [ while (start != end &&
          hashset.search(str.charAt(end)) == true) {
          hashset.remove(str.charAt(start))
          start++
    }
```

remove from start

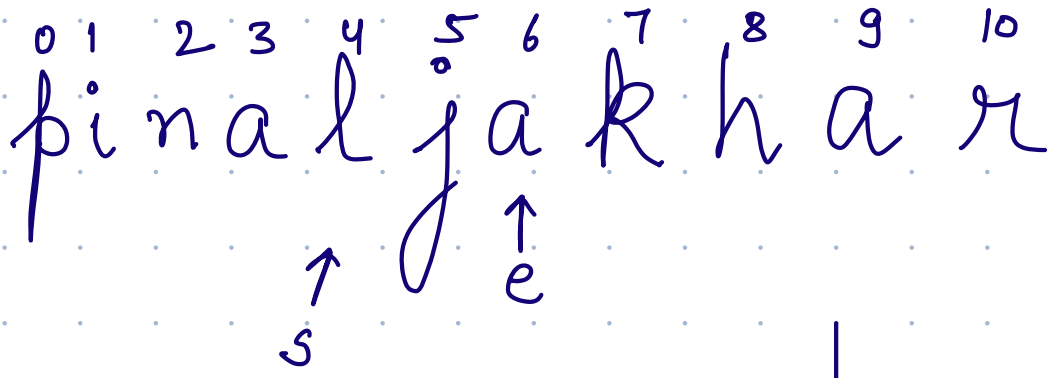


mylen = e - s = 5 e - s = 2

maxL = 5

e

p



mylen = 6 - 0 = 6

maxL = 6

l

j

I

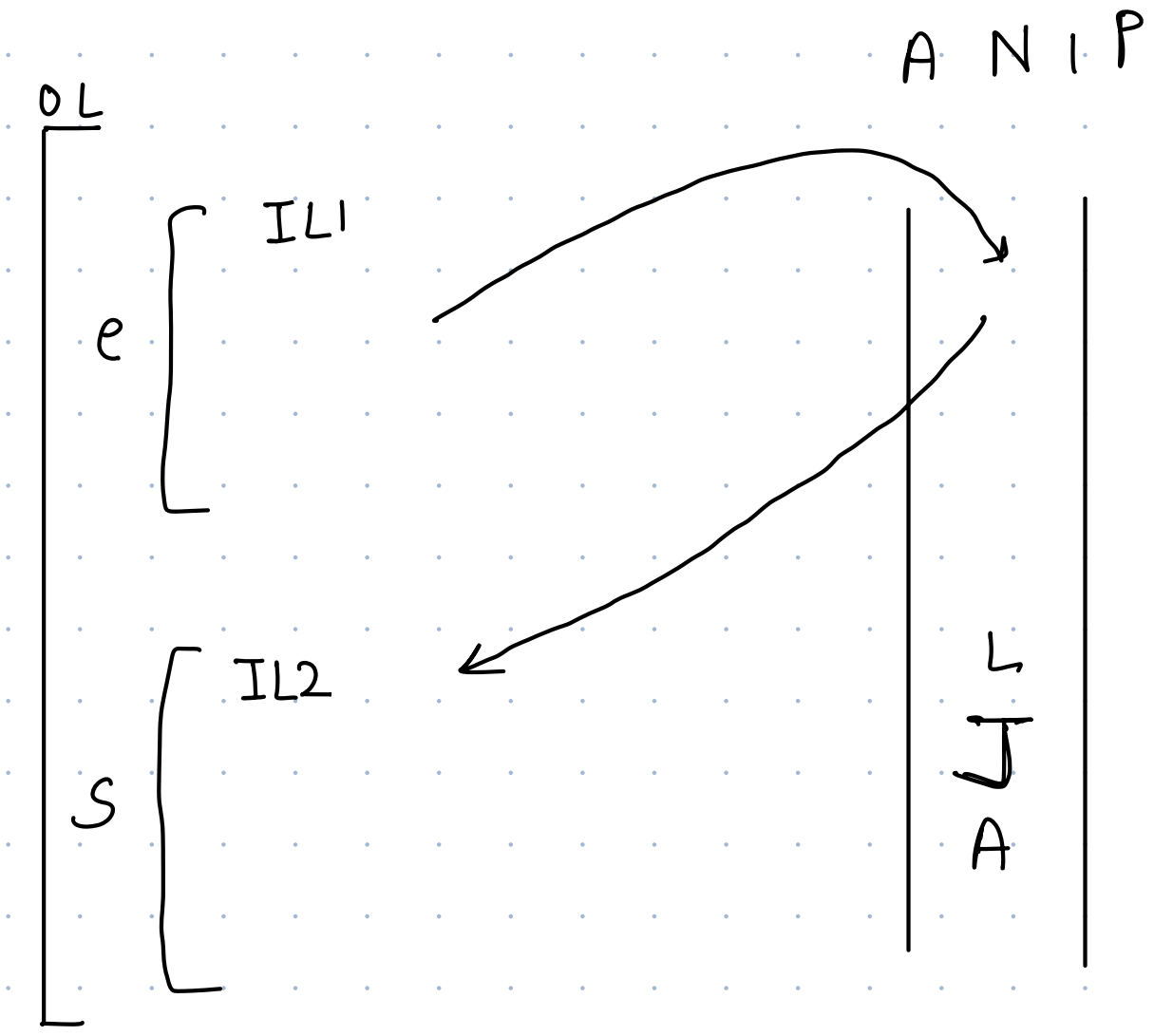
s == e

s != e && hs.search(a) == true

II

hashset.search(a) == false

[expanding ——— window]
[shrinking]



arr[i] ↻ = 2N = O(N)

Q2

1 2 3 1 2 5

ans = 3

4 3 3 2 5 6 4 5

ans = 2

arr = [2 6 8 4 7 2 9]

ans

S1 freq on map

==1 No

==1 Yes

8	1
6	1
4	1
2	2
7	1
9	1

elements are added in random order

S2: Iterate using array on hm

=> ==1 Yes

Problem Statement

Given an array of N elements, check if there exists a subarray with a sum equal to 0.

Example

if subarray with sum = 0?

Input:

N = 10

2 2 1 -3 4 3 1 -2 -3 2 true

Output:

if we add elements from index 1 to 3, we get 0; therefore, the answer is **true**.

1) ⇒ N = 6

2, 2, 1, -3, 4, 3 Yes

2) N = 6

4 5 -6 -3 8 9 Yes

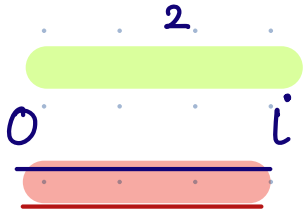
N = 6

2, 2, 1, -3, 4, 3

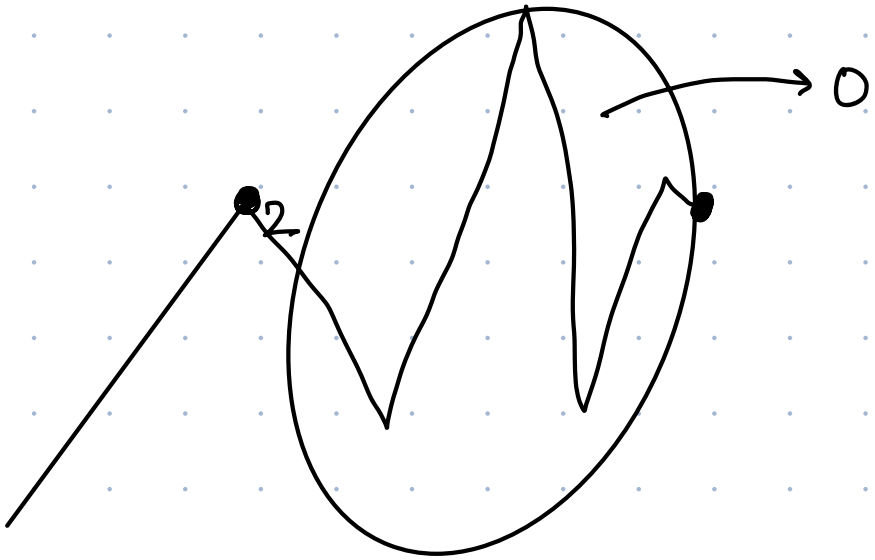
2 4 5 2 6 9

ar[0] + ar[1] + ar[2] + ar[3]

= 2 = psum[3]



ar[0] = 2 psum[0]



$j > i$

$0 - i = 0 - j$

$\underbrace{0+1+2 \ldots i} = \underbrace{0+1+2 \ldots i \ldots j}$

Sum $i+1$ to j

$\Rightarrow$ if psum value repeats, then yes $\rightarrow$ subarray sum = 0

psum[0]	psum[1]	psum[2]	psum[3]	psum[4]
0-0	0-1	0-2	0-3	0-4

psum[1] = sum(0 + 1)

psum[4] = sum(0 + 1 + 2 + 3 + 4)

0

S1: find psum array

S2: keep on adding psum values in hashset → if repetition comes return true.

⇒ [-1, 1]

psum = -1, 0 repetition of psum ✗

↳ directly psum = 0

S1: find psum array

S1.1 - check if psum[i] == 0 ✓ return true

S2: keep on adding psum values in hashset → if repetition comes ✓ return true

return false.

Problem Statement

Given an array arr[n] check if there exists a subarray with sum = K

Example:

We have the following array

Index 0 1 2 3 4 5 6 7 8

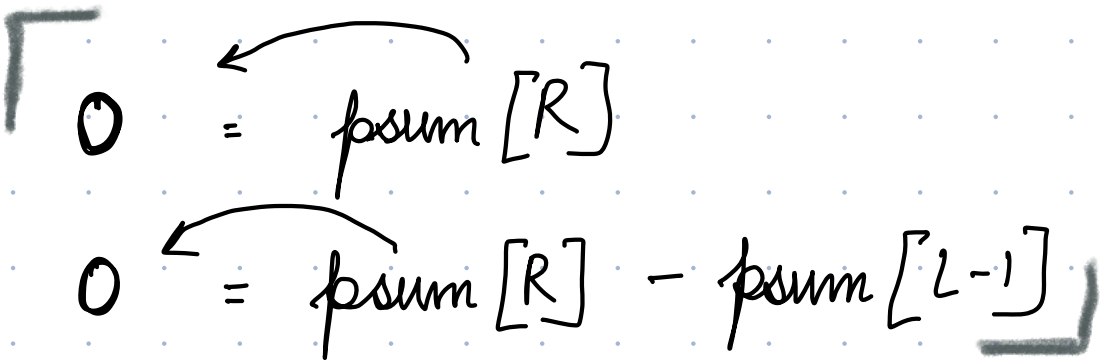
arr[7] 2 3 9 -4 1 5 6 2 5

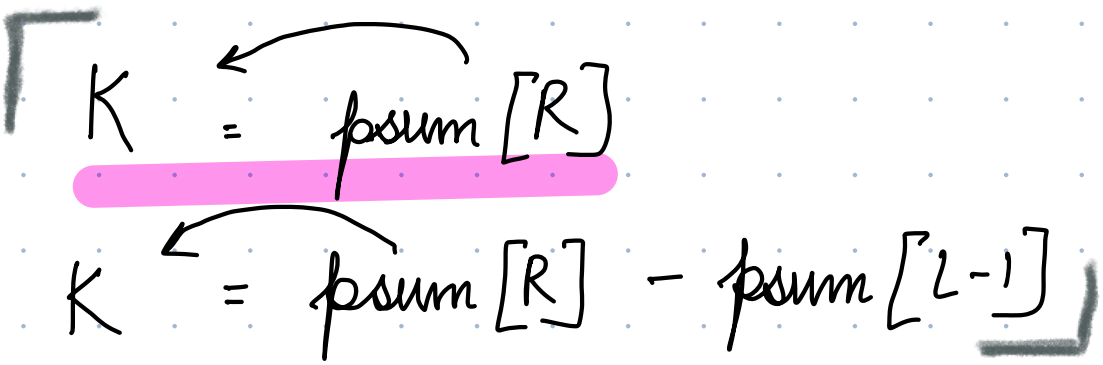
Check if it exists

Possible subarrays for the following values of K are,

- k = 11: {2 3 9 -4 1}, {5, 6} Yes
- k = 10: {2 3 9 -4} Yes
- k = 15: {-4, 1, 5, 6, 2, 5} = 15 Yes

K = 22 [2 3 9 -4 1 5 6] Yes





K = 11

2 3 9 -4 1 5 6 2 5

2 5 14 10 11 16 22 24 29

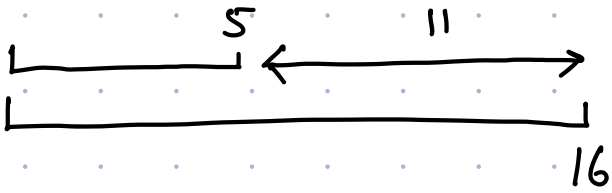
pair = 2 - K.
= -9

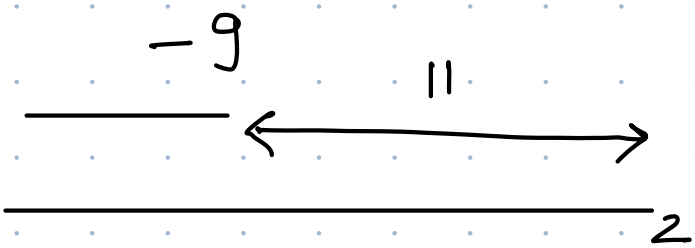
5-11 = -6
14-11 = 3
10-11 = -1
11-11 = 0
16-11 = 5

S_K = psum[R] - psum[L-1]

psum[R] = K + psum[L-1]
22 = 11 + 11

or psum[L-1] = psum[R] - K
= 22 - 11
= 11





Element traversal - one by one

psum[i]

pair psum[i] - K

Count

arr =	2	3	9	-4	1	5	6	2	5
psum =	2	5	14	10	11	16	22	24	29
	2-11	5-11	14-11	10-11	11-11	16-11	22-11	24-11	29-11
Pair	-9	-6	3	-1	0	5	11	13	18

Count sub sum = K, K = 11

2	3	9	-4	1
9	-4	1	5	
5	6			

count = 0 + 1 + 1 + 1 + ...

3

29	1
24	1
0	1
	=
2	1
11	(1)
5	=
22	1
14	1
16	1
10	1

0	1	2	3	4	5
2	3	-5	-4	9	11

0	2	5	0	-4	5	16
1		=				16-11
pair	-9	-6	-11	-15	-6	5

K = 11

2	1
5	2
0	2
-4	1

11

16 — 5

2 pairs

11 [5-5]

-5 -4 9 11 [2-5]

count = count + freq[pair]

psum ✓

hm.add(0, 1)

for every psum value →

pair = psum - K

count = count + freq[pair]

hm.add(psum, +1)

return count