18. Longest Subarray With Sum K (CodeStudio)

Monday, March 13, 2023 3:36 PM

arr[] = {1, 2, 3, 1, 1, 1, 1, 4, 2, 3}

k = 3

Subarray ⇒ contigous part of array. eg ⇒ {1, 2, 3} ✓ {4, 2, 3} ✓

This not subarray ⇒ {3, 4, 2} X

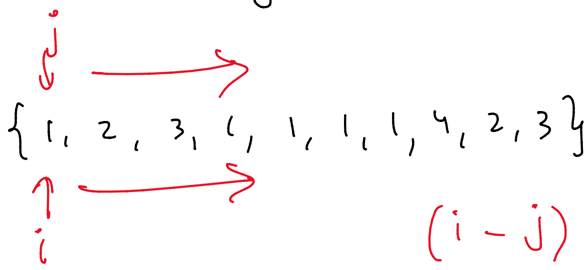
but this is subsequences.

$\{1, 2, 3, \underset{3}{\boxed{1, 1, 1}}, 1, 4, 2, 3\}$ k = 3

len = 3 (longest)

1) Brute force

→ Generate all subarrays



for first i → with whole j

for 2nd i → with whole j

so on until i goes at the end.

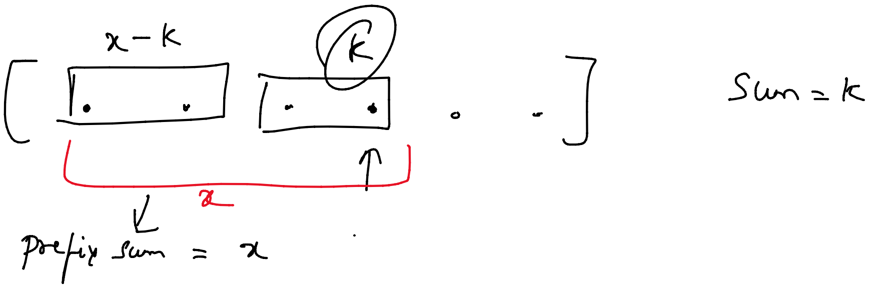
```
len = 0
for(i = 0 → i++)
{  for (j = i → j++)
     { s = 0
       for (k = i → j)                          T.C ⇒ $O(n^3)$
          s+  = a[k]                            S.C ⇒ $O(1)$
       if (s == k)  len = max(len, j-i+1)
     }
  }
  print(k)
```

2) Optimization

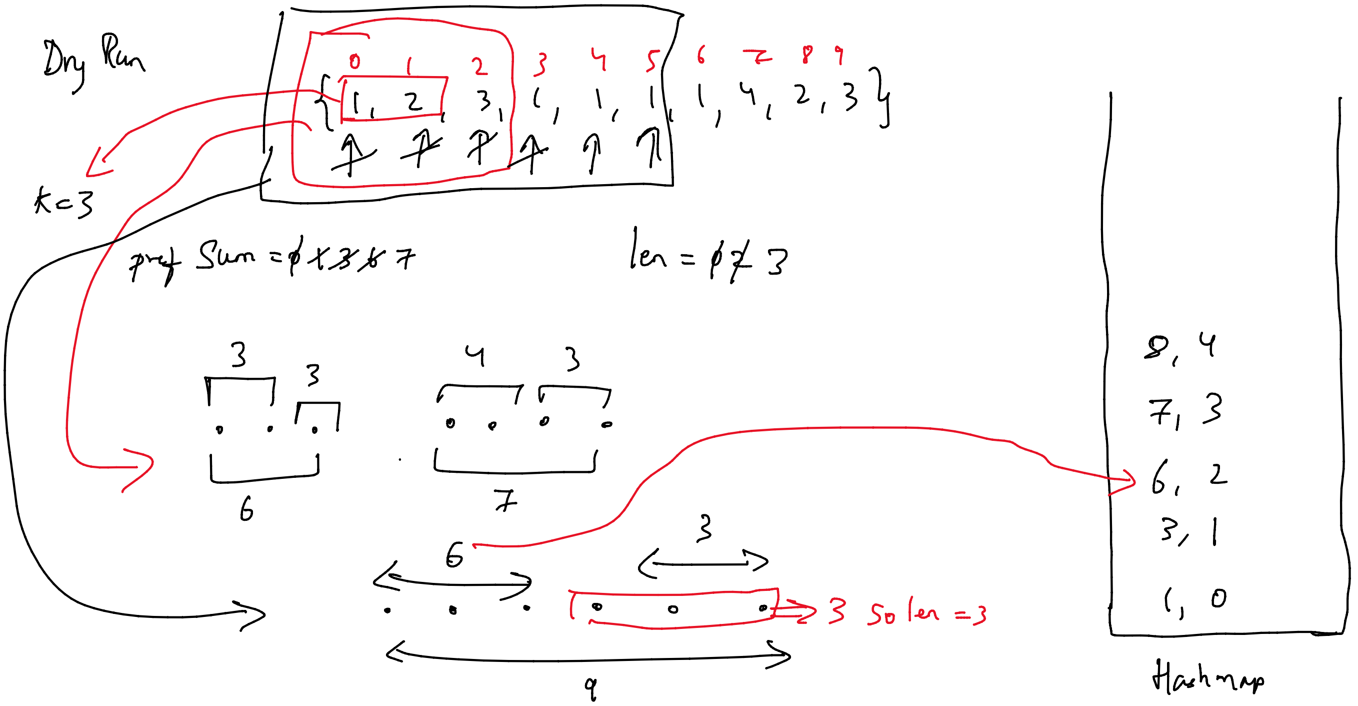
{1, 2, 3, 1, 1, 1, 1, 4, 2, 3}

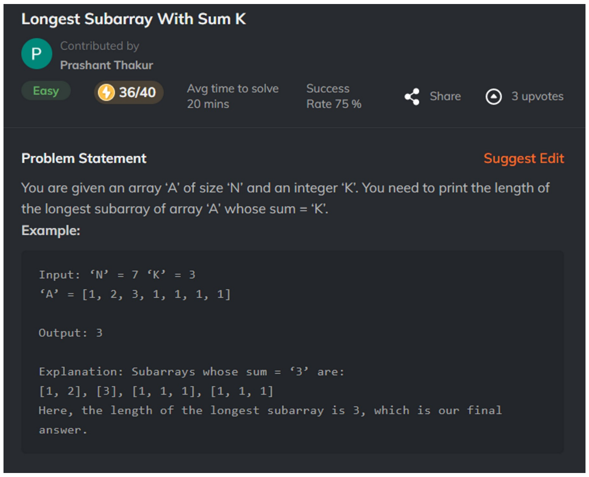
```
len = 0
for(i = 0 → i++)
{ s = 0
  for (j = i → j++)
   {
      s += a[j];                                T.C ⇒ $O(n^2)$
    if (s == k)  len = max(len, j-i+1)
   }
 }
 print(k)
```

Hashing:



if there exists a subarray with sum k as (·) as the last element





Longest Subarray With Sum K

Problem Statement

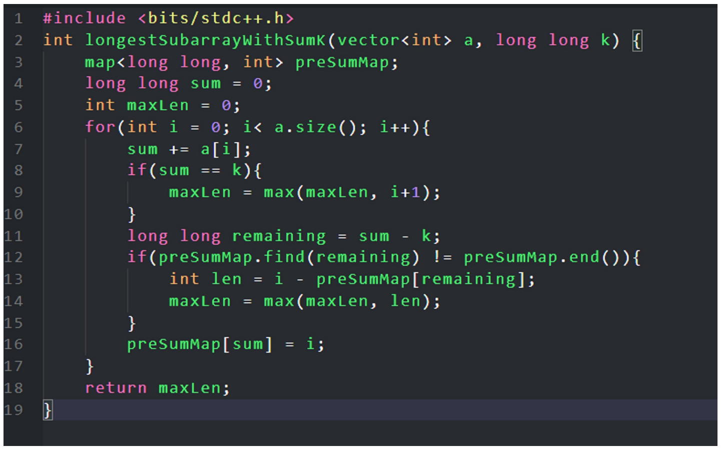
You are given an array 'A' of size 'N' and an integer 'K'. You need to print the length of the longest subarray of array 'A' whose sum = 'K'.

Example:

```
Input: 'N' = 7 'K' = 3
'A' = [1, 2, 3, 1, 1, 1, 1]

Output: 3

Explanation: Subarrays whose sum = '3' are:
[1, 2], [3], [1, 1, 1], [1, 1, 1]
Here, the length of the longest subarray is 3, which is our final
answer.
```



```cpp
#include <bits/stdc++.h>
int longestSubarrayWithSumK(vector<int> a, long long k) {
    map<long long, int> preSumMap;
    long long sum = 0;
    int maxLen = 0;
    for(int i = 0; i< a.size(); i++){
        sum += a[i];
        if(sum == k){
            maxLen = max(maxLen, i+1);
        }
        long long remaining = sum - k;
        if(preSumMap.find(remaining) != preSumMap.end()){
            int len = i - preSumMap[remaining];
            maxLen = max(maxLen, len);
        }
        preSumMap[sum] = i;
    }
    return maxLen;
}
```

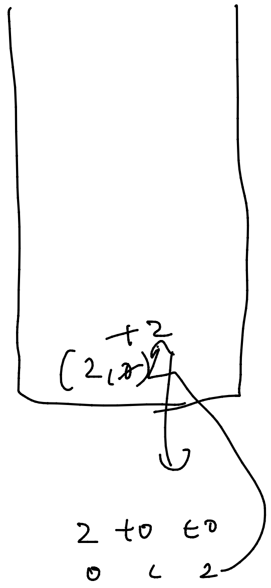
This code will not work for some test case

e.g.

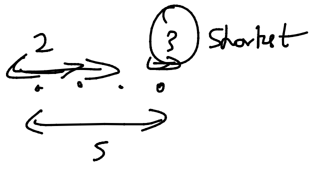
this code will not work for some test case

e.g.

arr[] = { 2, 0, 0, 3 }   k = 3



presum = ~~0~~ ~~2~~ 5   len = 0



we need longest subarray not shortest

T.C ⇒ O(N log N) (ordered map)

S.C ⇒ O(N)

Optimal

Two pointer

arr[] = {1, 2, 3, 1, 1, 1, 1, 3, 3}

k = 6

len = ~~0~~ ~~3~~ 4

sum = ~~1~~ ~~3~~ $\boxed{6}$, ~~7~~ > 6 So we will reduce it, so we will remove 1

~~6~~ ~~7~~ > 6

~~5~~

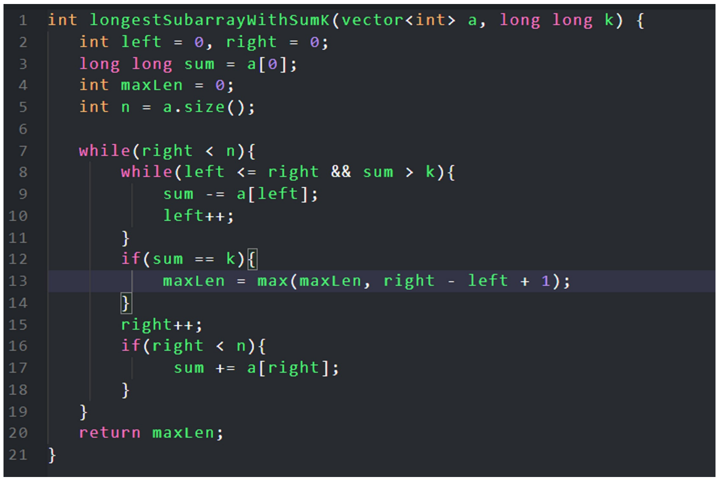
~~6~~ ~~7~~ ≥ 6

~~4~~ ~~7~~ ≥ 6

6 ~~9~~ > 6

~~8~~ 7 > 6

~~6~~



```cpp
int longestSubarrayWithSumK(vector<int> a, long long k) {
    int left = 0, right = 0;
    long long sum = a[0];
    int maxLen = 0;
    int n = a.size();

    while(right < n){
        while(left <= right && sum > k){
            sum -= a[left];
            left++;
        }
        if(sum == k){
            maxLen = max(maxLen, right - left + 1);
        }
        right++;
        if(right < n){
             sum += a[right];
        }
    }
    return maxLen;
}
```