Longest Subarray zero sum

BRUTE FORCE

Generate all subarrays and calculate sum and check if it is equal to 0 if yes then store maxlen.

```
#include <bits/stdc++.h>

int LongestSubsetWithZeroSum(vector < int > arr) {

  // Write your code here
  int n=arr.size();
  int sum=0,len=0,maxlen=0;
  for(int i=0;i<n;i++)
  {

    for(int j=i;j<n;j++)
    {
      sum=0;
      for(int k=i;k<=j;k++)
      {
        sum+=arr[k];

      }
      if(sum==0)
        maxlen=max(j-i+1,maxlen);

    }
  }
  return maxlen;
}
```

- Time Complexity : O(N^3)
- Space Complexity : O(1).

Improved Brute Force

Using two loops to calculate sum of subarrays and compare.

```
#include <bits/stdc++.h>

int LongestSubsetWithZeroSum(vector < int > arr) {
  // Write your code here
  int n=arr.size();
  int sum=0,len=0,maxlen=0;
  for(int i=0;i<n;i++)
  {
    sum=0;
    for(int j=i;j<n;j++)
    {
      sum+=arr[j];
      if(sum==0)
        maxlen=max(j-i+1,maxlen);

    }
  }
  return maxlen;
}
```

- Time Complexity : O(N^2)
- Space Complexity : O(1)

Optimal Approach :

Will work only when sum is k not for zero sum.

We will use the concept of prefix sum here. We look for sum ==k if it is found then we compare maxlen and i+1 len whichever is maximum. Then moving forward if we have sum x then we find whether x-k exist in already stored sum in the map , we will be storing each sum in each iteration in the map if the sum is found in map this means k is also found. We will store sum as well as index at which that sum was found subtracting x-k sum index with current sum index will give us the len of array whose sum is k.

```
class Solution {
public:
    int subarraySum(vector<int>& arr, int k) {
  // Write your code here
  map<int,int> m;
  int cnt=0,sum=0;
  for(int i=0;i<arr.size();i++)
  {
    sum+=arr[i];
    if(sum==k)
    {
        cnt++;
    }
    int remain=sum-k;
    if(m.find(remain)!=m.end())
    {
      cnt+=m[remain];
    }
     m[sum]++;

  }
  return cnt;

    }
};
```

- Time complexity : O(N)
- Space Complexity : O(N)

Optimal Approach :

Using two pointer if sum exceeds then we push left pointer and keep on moving right pointer and adding to sum. len is measured using right-left+1.

```
#include <bits/stdc++.h>

int LongestSubsetWithZeroSum(vector < int > arr,int k) {

  // Write your code here
    int left=0,right=0,n=arr.size(),maxlen=0;
    long long sum=arr[0];
    while(right<n)
    {

      while(left<=right && sum>k)
      {
        sum-=arr[left];
        left++;

      }
      if(sum==k)
      {
        maxlen=max(maxlen,right-left+1);
      }
      right++;
      if(right<n) sum+=arr[right];

    }
    return maxlen;

}
```