Set-A

P0.) Counting frequencies of array elements - GeeksforGeeks

```
void countFreq(int arr[], int n)
{
    unordered_map<int, int> mp;

    // Traverse through array elements and
    // count frequencies
    for (int i = 0; i < n; i++)
        mp[arr[i]]++;

    // Traverse through map and print frequencies
    for (auto x : mp)
        cout << x.first << " " << x.second << endl;
}
```

P1.)Key Pair | Practice | GeeksforGeeks

```
      // Function to check if array has 2 elements
      // whose sum is equal to the given value
     bool hasArrayTwoCandidates(int arr[], int n, int x) {
         unordered_map<int,int>mp;
         for(int i=0;i<n;i++){
             mp.insert({arr[i],i});  //store {arr[i],index(i)} in map
         }
         for(int i=0;i<n;i++){
             int b = x - arr[i];
             if(mp.find(b) != mp.end() && mp.at(b) != i){
//mp.at(b) checks second item(key) in map corresponding to b
//It will take care of Duplicate element
                 return true;
             }
         }
         return false;
     }
```

P2.)Array Subset of another array | Practice | GeeksforGeeks

```
unordered_map<int,int>mp1;
unordered_map<int,int>mp2;
int count = 0;
for(int i=0;i<n;i++){
    //storing frequency of each element of a1 in mp1
    mp1[a1[i]]++;
}
for(int i=0;i<m;i++){
    //storing frequency of each element of a2 in mp2
    mp2[a2[i]]++;
}
 //if frequency of each element in mp1 >= mp2
 //then a2 is subset of a1 else not(assuming a1.size()>a2.size())

for(int i=0;i<m;i++){
    if(mp1[a2[i]] >= mp2[a2[i]]){
       count++;
    }
}
if(count==m){
    return "Yes";
}
else{
    return "No";
}
```

Using <set>(This code will only work for non-duplicate array elements)

```
for(int i=0;i<n;i++)
{
   if(set.find(arr1[i]) == set.end())
       set.insert(arr1[i]);
}
int p=set.size();
for(int i=0;i<m;i++)
{
   if(set.find(arr2[i]) == set.end())
       set.insert(arr2[i]);
}
if(p==set.size())
{then yes otherwise no
```

Insert the elements in the Set of array 1 and then insert the element of array 2 in the same set, if the size of the hashset is same before and after then yes the array 2 is the subset of array 1.

TC -O(m+n) SC: O(m+n) as the insertion would take log n in Set but it is unordered so it doesn't matter.

P3.)Max distance between same elements | Practice | GeeksforGeeks

```
unordered_map<int,int>mp;
      int max_dist = 0;
      for(int i=0;i<n;i++){
          if(mp.find(arr[i]) == mp.end()){
              mp.insert({arr[i],i});
          }
          max_dist = max(max_dist,i-mp.at(arr[i]));
      }
      return max_dist;
```

TC-O(N) SC-O(N)

P4.)Minimum operation to make all elements equal in array - GeeksforGeeks

```
// Insert all elements in hash.
    unordered_map<int, int> hash;
    for (int i=0; i<n; i++)
        hash[arr[i]]++;

    // find the max frequency
    int max_count = 0;
    for (auto i : hash)
        if (max_count < i.second)
            max_count = i.second;

    // return result
    return (n - max_count);
```

TC-O(N) SC-O(N)

P6.)Check if a given array contains duplicate elements within k distance from each other - GeeksforGeeks

```
unordered_set<int> myset(k);
```

```
    for (int i = 0; i < n; i++)
    {
        // If already present n hash, then we found
        // a duplicate within k distance
        if (myset.find(arr[i]) != myset.end())
            return true;

        // Add this item to hashset
        myset.insert(arr[i]);

        // Remove the k+1 distant item
        if (i >= k)
            myset.erase(arr[i-k]);
    }
    return false;
```

P7.)Sum of elements in an array with frequencies greater than or equal to that element - GeeksforGeeks

```
unordered_map<int, int> m;
    for (i = 0; i < n; i++)
        m[arr[i]]++;

    int sum = 0;

    // Traverse the map using iterators
    for (auto it = m.begin(); it != m.end(); it++) {

        // Calculate the sum of elements
        // having frequencies greater than
        // or equal to the element itself
        if ((it->second) >= (it->first)      ) {
            sum += (it->first);
        }
    }

    return sum;
```

P8.)First Unique Character in a String - LeetCode(Very imp)

```
unordered_map<char, pair<int, int>> map;
```

```
    for (int i = 0; i < str.length(); i++)
    {
        map[str[i]].first++;
        map[str[i]].second = i;
    }

    // stores index of the first non-repeating character
    int min_index = -1;

    // traverse the map and find a character with count 1 and
    // a minimum index of the string
    for (auto it: map)
    {
        int count = it.second.first;
        int firstIndex = it.second.second;

        if (count == 1 && (min_index == -1 || firstIndex < min_index)) {
            min_index = firstIndex;
        }
    }

    return min_index;
```

P9.)https://leetcode.com/problems/find-common-characters/submissions/847626656/

Take two Maps → 1) store the character with their frequency count of the first string(0th index) so that we can compare with the others that are present.
2) start comparing from the further string i.e. 1st index

a) Start traversing the string and store the character with their frequency in map2 and calculate the minimum of the frequency of the characters that are common in both map1 and map2.Do this until all the strings in the vector “words” is covered.(form index 1→n)

3) Create vector<string>ans;

```
for(auto it:mp1) {
            string s = "";
            Char x = it.first;
            for(int j=0;j<it.second;j++) {
                ans.push_back(s+x);
            }
        }
```

4.) return ans

P10.)[Count pairs with given sum | Practice | GeeksforGeeks]

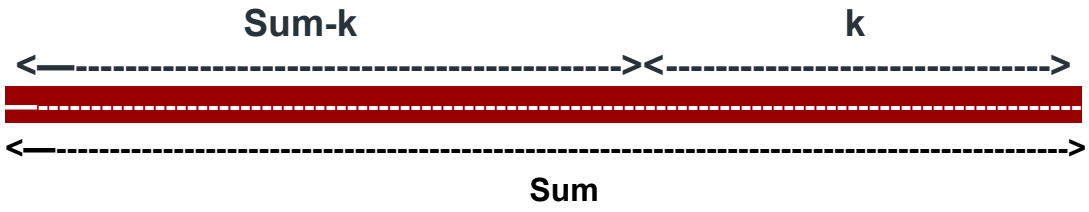
```
unordered_map<int,int>mp;
        int count = 0;

        for(int i=0;i<n;i++){
            int b = k-arr[i];
            if(mp.find(b) != mp.end()){
                count += mp.at(b);
            }
            mp[arr[i]]++;
        }
        return count;
```

Take a unordered_map and store an element an get the difference b =(k-element). If b is present in the map then increment the count as we got a pair for a given sum.
Note→Check whether b count has a value Null in the map then only insert.

Set-B

P1.)[Subarrays with sum K | Practice | GeeksforGeeks](Good Question)




```
  int sum = 0,count =0;
        unordered_map<int,int>mp;

        for(int i=0;i<N;i++){
            sum+= arr[i];     //Prefix Sum
            if(sum == k) count+=1;

            if(mp.find(sum-k) != mp.end()){
                count += mp[sum-k];
            }
            mp[sum]++;
        }
        return count;
```

If for a particular sum if we found sum-k in the map then subarray sum equals k is possible and we increment the count by frequency of <sum-k,freq> in map. And if we found sum == k then we increment the count by 1.

P2.)Smallest Subarray with Sum K from an Array - GeeksforGeeks

```
int sum = 0,minLength =INT_MAX;
int start=0,end=-1;
          unordered_map<int,int>mp;

          for(int i=0;i<N;i++){
              sum+= arr[i];     //Prefix Sum
              if(sum == k){
                 start=0;
                 end=i;
                  minLen = min(minLen,end-start);

               }

              if(mp.find(sum-k) != mp.end()){
                   start= mp[sum-k]+1;
                   end = i;
                   minLen = min(minLen,end-start+1);
              }
             else{
                   mp.insert({sum,i});
                 }
          }
           if(minLen >= INT_MAX) return -1;

           return minLen;
```

P3.)Subarray with given XOR | Interviewbit (Imp Question)

Approach1.) Find all the possible subarrays(Using Kadanes Algorithm)and calculate the XOR of each subarray separately and if it matches with the given number(B) then increment the count.

Optimized Approach 2.)

```
          int xorr = 0;
          int count = 0;
          int n1 = A.size();
          for(int i=0;i<n1;i++){
              xorr = xorr ^ A[i];
             int y = xorr ^ B;
```

```
            if(mp.find(y) != mp.end()){
                count += mp[y];
            }
            if(xorr == B){
                count++;
            }
             mp[xorr]++;
        }
        return count;
```

Use the prefix XOR
Taking the XOR of array means calculating previous element's XOR
Supposedly the given XOR→ k and y is the subarray
a)XORR→ y^k
Can also be written as y→ XORR^k
So now check if that y is present in map
a) if yes then increase the count var with k's frequency
Secondly check that if the prefix XOR == k
a) If yes then increase the count
If no condition is true then simply add the prefix xor in the map with its count as a general way.

TC→ O(NlogN)with map and O(N) with unordered_map SC→O(N)

P4.)[Continuous Subarray Sum - LeetCode](#)(Important Concept)

```
unordered_map<int,int>mp;
        int n = nums.size();
        int sum = 0;
        int start = 0;
        int end = 0;
        int length,rem=1;
        mp[0] = 1;ye isliye kar rhe hai kyuki agar kisi prefixSum 0 aa gaya toh hume dobara rem=0 check naa karna pade aur ek baar me hi 3rd if statement execute ho jaye kyuki hume pata hai ki agar prefixSum=0 hojaye toh wah must divisible by k hoga
        for(int i=0;i<n;i++){
            sum += nums[i];

            if(k!=0) {
                 rem = sum %k;
            }
```

```
            if(i!=0 and rem == 0){
                  return true;
              }

             if(rem<0){    //for taking care of -ve rem
                rem+=k;
            }

            if(mp.find(rem) != mp.end()){
               start = mp[rem]+1;
               end = i;
               length = end-start+1;
               if(length>=2){
                   return true;
               }
            }
            else{
                mp[rem] = i;
            }
        }
        return false;
```

Get the prefix sum

1) Now check if k is not 0 then simply get the remainder by dividing by k.
2) If remainder is not present in the map then insert in into map with its key as index(indicating the start index of the subarray)
3) If remainder is already present then we can conclude that there must me a subarray (in between current subarray and subarray at the time of getting the same remainder for the first time) which is divisible by k. Ex- [23,2,4,6,7] k= 6
   - I.1) Prefix sum = 23 rem = 5
   - I.2) Prefix sum = 25 rem = 1
   - I.3) Prefix sum = 29 rem = 5 (prefSum repeats)

   Start = mp[rem]+1, end = i, length = 2(>1)
   {2,4}=>required prefsum
4) At the start insert mp[0] = 1 so that if remainder come as zero then we do not need to check for another remainder as 0
5) if(rem<0) then rem = rem + k

Video Solution: Continuous Subarray Sum - Leetcode 523 - Python

P5.)Subarray Sums Divisible by K - LeetCode

```
->Similar as previous one

 unordered_map<int,int>mp;
        int n = nums.size();
        int sum = 0;
```

```
        int start = 0;
        int end = 0;
        int count=0,rem=0;
         mp[0] = 1;ye isliye kar rhe hai kyuki agar kisi prefixSum 0 aa gaya toh hume dobara rem=0 check naa karna pade aur ek baar me hi 3rd if statement execute ho jaye kyuki hume pata hai ki agar prefixSum=0 hojaye toh wah must divisible by k hoga

        for(int i=0;i<n;i++){
            sum += nums[i];

            if(k!=0) {
                 rem = sum %k;
            }

            if(rem<0){    //for taking care of -ve rem
                rem+=k;
            }

            if(mp.find(rem) != mp.end()){

                count += mp[rem];
                mp[rem]++;

            }
            else{
                mp[rem] = 1;
            }
        }
        return count;
```

Video Solution:

Subarrays Sums Divisible by K (Leetcode 974) Algorithm Explained

P6.)[Count subarrays with sum equal to its XOR value - GeeksforGeeks]

P7.)[Max Number of K-Sum Pairs - LeetCode]

```
unordered_map<int,int>mp;
        int count = 0;
         int n = nums.size();
```

```
        for(int i=0;i<n;i++){
            int b = k-nums[i];
            if(mp[b]>0){
                count++;
                Mp[b]--;
            }else{
                mp[nums[i]]++;
            }

        }
        return count;
```

Take a map and keep track of the remaining element is in it or not and their freq >0 because then only we can use their freq
Then increase the count and reduce their freq by 1 as they are used once.
If that current element of array is not present then put it in the map in a general way

P8.)Count Number of Pairs With Absolute Difference K - LeetCode

```
 unordered_map<int,pair<int,int>> mp;
        int n= nums.size();
        int count = 0;

        for(int i =0;i<n;i++){
            int diff1 = nums[i]-k;
            int diff2 = nums[i]+k;
            if(mp.find(diff1) != mp.end() && i>mp[diff1].second){
                count += mp[diff1].first;
            }
             if(mp.find(diff2) != mp.end() && i>mp[diff1].second){
                count += mp[diff2].first;
             }

             mp[nums[i]].first++;
            mp[nums[i]].second = i;

        }
        return count;
```

Put all the elements in the map with their respective frequencies .
Now Check whether array[i]-k is present in the map or not ,if yes increase the count .Similarly do for array[i]+k as absolute difference in question means |(2-5 = k)|and |(5-2 = k)| both are same thing.Pretty much straight forward.

P10.)Triplet Sum in Array | Practice | GeeksforGeeks

```
for(int i=0;i<n;i++)
         {
```

```
            unordered_set<int> s;
            int curr_sum= X-A[i];

            for(int j=i+1;j<n;j++)
            {
                if(s.find(curr_sum-A[j])!=s.end())
                {
                    return true;
                }

                s.insert(A[j]);
            }
        }
      return false;
```

P11.)[Count quadruplets with sum K from given array - GeeksforGeeks]

Additional Important Question

//********KADANE'S ALGORITHM*************

```
int sum = 0;
int  maxi = INT_MIN;
for(auto it : nums){
   sum += it;
   maxi = max(sum,maxi);
   if(sum<0){
     sum = 0;
   }
}
return maxi;
```