Lesson:

Arrays - 3
(Basic Problem Solving)

PW SKILLS

Pre-Requisites

- Concept of arrays
- Vectors

List of Concepts Involved

- Problem solving using arrays and vectors

Pattern: Target sum (Brute Force Approach)

In this approach, we try to find the sum of the elements of an array to yield a particular target value. In the brute force approach, we use the nested for loop and try to match the value at every point.

Important Note: The number of loops depends on the number of elements we have to take in the array. For example, for pair sum, we use 2 nested for loops. And for triple sum, we use triple nested for loop.

Let us go through the problems discussed below for better understanding.

Problem 1: Find the total number of pairs in the array whose sum is equal to the given value x.

Code:

```
int pairSum(vector<int> &a, int x) {
      int count = 0;
      int n = a.size();
      for(int i=0;i<n;i++)
          for(int j=i+1;j<n;j++)
              if(a[i]+a[j]==x)
                  count++;
      return count;
  }
```

Explanation: Traverse through the array using a nested for loop (i.e. one for loop within another for loop), and add/consolidate the sum of the elements at each iteration. Compare that sum with the target value 'x' and increment the count if found equal. Return the value of count in the end.

Problem 2: Count the number of triplets whose sum is equal to a given value x.

Code:

```
int findTriplet(vector<int> &a, int x) {
      int n = a.size();
      int count = 0;
      for(int i=0;i<n;i++){
          for(int j=i+1;j<n;j++){
              for(int k=j+1;k<n;k++){
                  if(a[i]+a[j]+a[k]==x){
                      count++;
                  }
              }
          }
      }
      return count;
  }
```

Explanation: Traverse through the array using a triple nested for loop, and just add the sum of the elements at each iteration, then compare that sum with target value 'x' and increment the count if they are found to be equal. Return the value of count in the end.

Pattern: Basic Miscellaneous Problems
Here, we will discuss array manipulation problems where we perform some operations on the array in order to solve the problem.

Problem 3: Find the unique number in a given array where all the elements are repeated twice with one value being unique.

Code:

```
int findUnique(vector<int> &a){
      int n = a.size();
      for(int i=0;i<n;i++){
          for(int j=i+1;j<n;j++){
              if(a[i]==a[j]){
                  a[i] = a[j] = -1;
              }
          }
      }
      for(int i=0;i<n;i++)
          if(a[i]>0)
              return a[i];
      return -1;
  }
```

Explanation: Traverse through the array. In the first traversal, take the current element and find if it appears again in the array using a nested for loop. If it appears again in the array, just mark it is as -1. In the end, use a single traversal to find if any element is positive, that means it has no duplicate and hence it is our answer.

Problem 4: Find the second largest value in a given array.

Code:

```
int secondLargestValue(vector<int> &a) {
    int n = a.size();
    int max = INT_MIN;
    int smax = INT_MIN;
    for(int i=0;i<n;i++){
        if(max<a[i])
            max = a[i];
    }
    for(int i=0;i<n;i++){
        if(a[i]≠max)
        {
            if(a[i]>smax)
                smax = a[i];
        }
    }
    return smax;
}
```

Explanation: First find the max using a single traversal. It can be found by comparing every element while traversing through the array. In the second traversal, check at every point if the current element is not the max element, it can be compared to find the second max element.

Problem 5: Rotate the given array 'a' by k steps, where k is non-negative.
Note: k can be greater than n as well where n is the size of array 'a'.

Code :

```
// with extra space
vector<int> rotate(vector<int>& a, int k) {
    int n = a.size();

    k %= n; // k can be greater than n
    vector<int> ans(n,-1);

    for(int i = n-k; i ⩽ n-1; i++) {
        ans[i-(n-k)] = a[i];
    }

    for(int i = 0; i < n-k; i++) {
        ans[i+k] = a[i];
    }

    return ans;
}
```

Explanation: Create a new array 'ans' and add the last k elements first in the ans array and the rest of the elements after that. We can do that by traversing the array twice as shown. Note: Make sure to take modulus of k by n, as value of k can be greater than n so it will become less than n as :

k = constant * n + reminder
Here reminder = k % n
If we rotate an array n times it will not be of any use as the array will come back to the same position, so remove that from k and take the remaining part only.

Problem 6: For Q inputs, check if the given number is present in the array or not.
Note: Value of all the elements in the array is less than 10^5.

Code :

```
 int n;
  cin >> n;
  vector<int> a(n);
  for (int i = 0; i < n; i++) {
    cin >> a[i];
  }

  const int N = 1e5 + 10;
// creating a frequency array as max(a[i]) < 10^5
  vector<int> freq(N, 0);
  for (int i = 0; i < n; i++) {
    freq[a[i]]++;
  }

  int q;
  cin >> q;
  while (q--) {
    int val;
    cin >> val;
    if (freq[val] > 0) {
      cout << "YES" << endl;
    } else cout << "NO" << endl;
  }
```

```
Input :
4
1 1 2 7

4
1
2
3
4
```

```
Output:
YES
YES
NO
NO
```

Explanation: Create an array 'freq' to store frequency of element and increment the count of values that the user is entering. This way we have the count of all the values that the user has entered and the answer/output would be YES whenever the count of the values will be positive; when the count is 0, the answer will be NO.

```
That is all for this lesson! See you in the next one. Happy Learning !
```

Upcoming Class Teasers:

- Advance problem solving using arrays