Lesson:

# Problem on Arrays

SKILLS

## Pre-Requisites

- Arrays
- Vectors

## List of Concepts Involved

- Array Problems based on 2 pointer approach

## Pattern: Two Pointers

In this approach, we declare two pointers and perform operations on the array. The two-pointers are usually declared, one at the start and the other at the end. We increment and decrement them as per the problem. Look at the problems discussed below to have a better understanding.

**Problem 1:** Sort an Array consisting of only 0s and 1s.

**Explanation of approach:** Take two pointers, one from start and the other from the end. If the start pointer's element has a value 1 and the end pointer's element has a value 0, then we swap them (as we want to sort the array). Check this condition, at every iteration, and interchange the values if the condition is satisfied, else increment the starting pointer and decrement the ending pointer as done in the program.

**Code:**

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

      }
      return;
  }
```

**Problem 2:** Given an array of integers 'a', move all the even integers at the beginning of the array followed by all the odd integers. The relative order of odd or even integers does not matter. Return any array that satisfies the condition.

**Input :**
```
[1,2,3,4,5]
```
**Output :**
```
[4,2,3,1,5]
```

**Explanation of approach:**
We need to keep all the even parity values at the beginning followed by the odd values so whenever we encounter the case of an even value at the end pointer and odd value at the start pointer, we swap them. Also, we keep on incrementing the pointers if the values are already at the correct position.

**Code:**
```
     vector<int> sortArrayByParity(vector<int>& a) {
        int i = 0, j = a.size()-1;

        while(i < j) {
            if(a[i] % 2 == 1 && a[j] % 2 == 0) {
                swap(a[i], a[j]);
                i++, j--;
            }
            if(a[i] % 2 == 0) i++;
            if(a[j] % 2 == 1) j--;
        }

        return a;
    }
```

**Problem 3:** Given an integer array ‘a’ sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing/increasing order.

**Input :**
```
[-10,-3,2,5,6]
```

**Output :**
```
[4,9,25,36,100]
```

**Explanation of approach:**
Note that the square of a negative number is always positive so we can just compare the absolute values. Given that the array is sorted, we can just take the higher absolute value from the start or end of the array and move our pointers accordingly. In the end, we will be required to reverse the answer as we are taking the higher values first so it will be in decreasing order and we have to return our answer in non-decreasing/increasing order.

**Code:**
```
vector<int> sortedSquares(vector<int>& a) {
        int n = a.size(), i = 0, j = n-1;
        vector<int> ans;

        while(i <= j) {
            if(abs(a[i]) < abs(a[j])) {
                a[j] *= a[j];
                ans.push_back(a[j--]);
            } else {
                a[i] *= a[i];
                ans.push_back(a[i++]);
            }
        }

        reverse(ans.begin(), ans.end());

        return ans;
    }
```

**That is all for this lesson! Do not miss the next one !!**

## Upcoming class teaser :

- Basic problems based on arrays.