Day - 4 : 14/11/2024

1. **Transition Point:**

```
class Solution {
  public:
        int transitionPoint(vector<int>& arr) {
        int transition_point = -1;
        int n = arr.size();
        for(int i = 1;i < n; i++){
        if(arr[i-1] != arr[i]){
        transition_point = i;
        }
        }
        if(transition_point == -1 && arr[n-1] == 0){
        return -1;
        }
        else if(transition_point == -1 && arr[n-1] == 1){
        return 0;
        }
        return transition_point;
        }
};
```

Time Complexity: O(n)

2. **Wave Array:**

```
class Solution {
  public:
        void convertToWave(vector<int>& arr) {
        int n = arr.size();
        for(int i=0; i<n-1; i+=2){
        swap(arr[i],arr[i+1]);
        }
        }
};
```

Time Complexity: O(n)

3. **First Repeating Element:**

```
class Solution {
  public:
        int firstRepeated(vector<int> &arr) {
        int minIndex = INT_MAX;
        unordered_map<int, int> firstOccur;
        for(int i=0;i<arr.size();i++){
```

```
            if(firstOccur.find(arr[i]) != firstOccur.end()){
            minIndex = min(firstOccur[arr[i]],minIndex);
            }
            else{
            firstOccur[arr[i]] = i;
            }
            }
            if(minIndex == INT_MAX){
            return -1;
            }
            return minIndex+1;
            }
};
```

Time Complexity: O(n)

4. **Stock Buy and Sell:**

```
class Solution {
  public:
    int maximumProfit(vector<int> &prices) {
        int i=0;
        int j=0;
        int result =0;
        while(j < prices.size())
        {
            if(prices[j] > prices[i])
            {
                result = max(result, prices[j]-prices[i]);
            }
            else{
                i= j;

            }
            ++j;
        }
        return result;
    }
};
```

TimeComplexity : O(n)

5. **Coin Change:**

```
class Solution {
  public:
    int count(vector<int>& coins, int sum) {
        vector<int> dp(sum + 1, 0);
        dp[0] = 1;
```

```
        for (int coin : coins) {
            // Update dp array for all sums >= coin
            for (int i = coin; i <= sum; i++) {
                dp[i] += dp[i - coin];
            }
        }
        return dp[sum];
    }
};
```

TimeComplexity : O(n)

6. **Remove Duplicate Elements:**

```
int removeDuplicates(vector<int>& arr) {
     unordered_set<int> s;
    int idx = 0;

    for (int i = 0; i < arr.size(); i++) {
        if (s.find(arr[i]) == s.end()) {
            s.insert(arr[i]);
            arr[idx++] = arr[i];
        }
    }

      return s.size();
}
```

Time Complexity: O(n)

7. **Maximum Index:**

```
class Solution {
  public:
    int maxIndexDiff(vector<int>& arr) {
        int maxIndex = 0;
        int prevIndex = 0;
        int maxNum = 0;
        for(int i = 0; i<arr.size(); i++){
            if(arr[i] > maxNum){
                maxNum = arr[i];
                maxIndex = i;
            }
        }
        return maxIndex - prevIndex;
    }
};
```

Time Complexity: O(n)