Day - 4 : 14/11/2024

1. **Transition Point:**

```
class Solution {
public:
    int transitionPoint(vector<int>& nums) {
        int transitionIdx = -1;
        int size = nums.size();
        for (int i = 1; i < size; i++) {
            if (nums[i - 1] != nums[i]) {
                transitionIdx = i;
                break;
            }
        }
        if (transitionIdx == -1) {
            return nums[size - 1] == 0 ? -1 : 0;
        }
        return transitionIdx;
    }
};
```

Time Complexity: O(n)

2. **Wave Array:**

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

Time Complexity: O(n)

3. **Stock Buy and Sell:**

```
class Solution {
public:
    int maximumProfit(vector<int>& prices) {
        int buyIndex = 0;
        int maxProfit = 0;
        for (int sellIndex = 0; sellIndex < prices.size(); sellIndex++) {
            if (prices[sellIndex] > prices[buyIndex]) {
                maxProfit = max(maxProfit, prices[sellIndex] - prices[buyIndex]);
```

```
            } else {
                buyIndex = sellIndex;
            }
        }
        return maxProfit;
    }
};
```

TimeComplexity : O(n)

4. **Coin Change:**

```
class Solution {
public:
    int count(vector<int>& denominations, int target) {
        vector<int> ways(target + 1, 0);
        ways[0] = 1;
        for (int coin : denominations) {
            for (int i = coin; i <= target; i++) {
                ways[i] += ways[i - coin];
            }
        }
        return ways[target];
    }
};
```

TimeComplexity : O(n)

5. **Remove Duplicate Elements:**

```
int removeDuplicates(vector<int>& nums) {
    unordered_set<int> uniqueNums;
    int pos = 0;

    for (int i = 0; i < nums.size(); i++) {
        if (uniqueNums.find(nums[i]) == uniqueNums.end()) {
            uniqueNums.insert(nums[i]);
            nums[pos++] = nums[i];
        }
    }

    return uniqueNums.size();
}
```

Time Complexity: O(n)

6. **Maximum Index:**

```
class Solution {
```

```
public:
    int maxIndexDiff(vector<int>& nums) {
        int maxIdx = 0;
        int minIdx = 0;
        int maxElement = nums[0];

        for (int i = 0; i < nums.size(); i++) {
            if (nums[i] > maxElement) {
                maxElement = nums[i];
                maxIdx = i;
            }
        }

        return maxIdx - minIdx;
    }
};
```

Time Complexity: O(n)

7. **First Repeating Element:**

```
class Solution {
public:
    int firstRepeated(vector<int>& nums) {
        int minPosition = INT_MAX;
        unordered_map<int, int> elementPos;
        for (int i = 0; i < nums.size(); i++) {
            if (elementPos.find(nums[i]) != elementPos.end()) {
                minPosition = min(minPosition, elementPos[nums[i]]);
            } else {
                elementPos[nums[i]] = i;
            }
        }
        return minPosition == INT_MAX ? -1 : minPosition + 1;
    }
};
```

Time Complexity: O(n)