Day - 1 : 11/11/2024

1. **Equal Array:**

```
class Solution {
  public:
    bool check(vector<int>& arr1, vector<int>& arr2) {
        sort(arr1.begin(),arr1.end());
        sort(arr2.begin(),arr2.end());
        return arr1 == arr2;
    }
};
```

Time Complexity: O(N log N)

2. **Linked List Palindrome:**

```
class Solution {
public:
    bool isPalindrome(Node *head) {
        vector<int> arr;
        Node* current = head;
        while (current != nullptr) {
            arr.push_back(current->data);
            current = current->next;
        }
        int n = arr.size();
        for (int i = 0; i < n / 2; i++) {
            if (arr[i] != arr[n - i - 1]) {
                return false;
            }
        }
        return true;
    }
};
```

Time Complexity: O(N)

3. **Floor in Sorted Array:**

```
class Solution {
  public:
    int findFloor(vector<int>& arr, int k) {
        int maxRes = -1;
        int res = -1;
        for (int i = 0; i < arr.size(); i++) {
            if (arr[i] <= k && arr[i] > maxRes) {
                maxRes = arr[i];
```

```
                res = i;
            }
        }
        return res;
    }
};
```

Time Complexity: O(N)

4. **<u>Triplet Sum:</u>**

```
class Solution {
public:
    bool find3Numbers(int arr[], int n, int x) {
        sort(arr, arr + n);

        for (int i = 0; i < n - 2; i++) {
            int left = i + 1;
            int right = n - 1;

            while (left < right) {
                int currentSum = arr[i] + arr[left] + arr[right];

                if (currentSum == x) {
                    return true;
                } else if (currentSum < x) {
                    left++;
                } else {
                    right--;
                }
            }
        }

        return false;
    }
};
```

Time Complexity: O(N^2)

5. **<u>0/1 Knapsack Problem:</u>**

```
class Solution {
public:
    int knapSack(int capacity, vector<int>& val, vector<int>& wt) {
        int n = val.size();
        vector<vector<int>> dp(n + 1, vector<int>(capacity + 1, 0));

        for (int i = 1; i <= n; i++) {
            for (int j = 0; j <= capacity; j++) {
                dp[i][j] = dp[i - 1][j];
```

```
            if (j >= wt[i - 1]) {
                dp[i][j] = max(dp[i][j], dp[i - 1][j - wt[i - 1]] + val[i - 1]);
            }
        }
    }

    return dp[n][capacity];
  }
};
```

Time Complexity: O(N * Capacity)
Output:

6. **<u>Balanced tree check</u>**

```
class Solution {
public:
  bool isBalanced(TreeNode* root) {
    return checkHeight(root) != -1;
  }

private:
  int checkHeight(TreeNode* node) {
    if (!node) return 0;

    int leftHeight = checkHeight(node->left);
    if (leftHeight == -1) return -1;

    int rightHeight = checkHeight(node->right);
    if (rightHeight == -1) return -1;

    if (abs(leftHeight - rightHeight) > 1) return -1;

    return max(leftHeight, rightHeight) + 1;
  }
};
```

Time Complexity: O(N)