Day - 3 13/11/24

1. **K-th Smallest Element in an Array:**

```
class Solution {
public:
    int kthSmallest(vector<int>& arr, int k) {
        priority_queue<int> maxHeap;
        for (int num : arr) {
            maxHeap.push(num);
            if (maxHeap.size() > k) maxHeap.pop();
        }
        return maxHeap.top();
    }
};
```

Time Complexity: O(n log k)

2. **Minimize the Maximum Difference Between Heights:**

```
class Solution {
public:
    int minimizeDifference(vector<int>& heights, int k) {
        int n = heights.size();
        sort(heights.begin(), heights.end());
        int result = heights[n - 1] - heights[0];

        for (int i = 1; i < n; i++) {
            int maxVal = max(heights[i - 1] + k, heights[n - 1] - k);
            int minVal = min(heights[0] + k, heights[i] - k);
            result = min(result, maxVal - minVal);
        }

        return result;
    }
};
```

Time Complexity: O(n log n)

3. **Parenthesis Checker:**

```
class Solution {
public:
    bool isValidParenthesis(string s) {
        stack<char> st;
        for (char ch : s) {
            if (ch == '(' || ch == '{' || ch == '[') st.push(ch);
            else {
                if (st.empty() ||
                    (ch == ')' && st.top() != '(') ||
```

```
                (ch == '}' && st.top() != '{') ||
                (ch == ']' && st.top() != '[')) return false;
            st.pop();
        }
    }
    return st.empty();
  }
};
```

Time Complexity: O(n)

4. **Equilibrium Point in an Array:**

```
class Solution {
public:
  int findEquilibrium(vector<int>& arr) {
    int totalSum = 0, leftSum = 0;
    for (int num : arr) totalSum += num;

    for (int i = 0; i < arr.size(); i++) {
      totalSum -= arr[i];
      if (leftSum == totalSum) return i;
      leftSum += arr[i];
    }

    return -1;
  }
};
```

Time Complexity: O(n)

5. **Binary Search:**

```
class Solution {
public:
  int binarySearch(vector<int>& arr, int target) {
    int left = 0, right = arr.size() - 1;
    while (left <= right) {
      int mid = left + (right - left) / 2;
      if (arr[mid] == target) return mid;
      else if (arr[mid] < target) left = mid + 1;
      else right = mid - 1;
    }
    return -1;
  }
};
```

Time Complexity: O(n)

6. **Next Greater Element:**

```
class Solution {
public:
    vector<int> nextGreaterElements(vector<int>& arr) {
        vector<int> result(arr.size(), -1);
        stack<int> st;

        for (int i = 0; i < arr.size(); i++) {
            while (!st.empty() && arr[st.top()] < arr[i]) {
                result[st.top()] = arr[i];
                st.pop();
            }
            st.push(i);
        }

        return result;
    }
};
```

Time Complexity: O(n)

7. **Union of Two Arrays:**

```
class Solution {
public:
    vector<int> unionOfArrays(vector<int>& arr1, vector<int>& arr2) {
        set<int> resultSet(arr1.begin(), arr1.end());
        resultSet.insert(arr2.begin(), arr2.end());

        return vector<int>(resultSet.begin(), resultSet.end());
    }
};
```

Time Complexity: O(n+m)