Day - 3 13/11/24

1. **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 ans = heights[n - 1] - heights[0];

        for (int i = 1; i < n; i++) {
            int maxH = max(heights[i - 1] + k, heights[n - 1] - k);
            int minH = min(heights[0] + k, heights[i] - k);
            ans = min(ans, maxH - minH);
        }

        return ans;
    }
};
```

Time Complexity: O(n log n)

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

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

Time Complexity: O(n log k)

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

```
class Solution {
public:
    int findEquilibrium(vector<int>& nums) {
        int sum = 0, left = 0;
        for (int x : nums) sum += x;
        for (int i = 0; i < nums.size(); i++) {
            sum -= nums[i];
            if (left == sum) return i;
            left += nums[i];
        }

        return -1;
    }
};
```

Time Complexity: O(n)

4. **<u>Binary Search:</u>**

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

Time Complexity: O(n)

5. **<u>Parenthesis Checker:</u>**

```
class Solution {
public:
    bool isValidParenthesis(string str) {
        stack<char> stk;
        for (char ch : str) {
            if (ch == '(' || ch == '{' || ch == '[') stk.push(ch);
            else {
                if (stk.empty() ||
                    (ch == ')' && stk.top() != '(') ||
                    (ch == '}' && stk.top() != '{') ||
                    (ch == ']' && stk.top() != '[')) return false;
                stk.pop();
            }
        }
        return stk.empty();
    }
};
```

Time Complexity: O(n)

6. **Next Greater Element:**

```
class Solution {
public:
    vector<int> nextGreaterElements(vector<int>& nums) {
        vector<int> res(nums.size(), -1);
        stack<int> stk;

        for (int i = 0; i < nums.size(); i++) {
            while (!stk.empty() && nums[stk.top()] < nums[i]) {
                res[stk.top()] = nums[i];
                stk.pop();
            }
            stk.push(i);
        }

        return res;
    }
};
```

Time Complexity: O(n)

7. **Union of Two Arrays:**

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

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

Time Complexity: O(n+m)