Day - 5 18/11/24

1. **Quick Sort:**

```
class Solution {
public:
    void quickSort(vector<int>& nums, int low, int high) {
        if (high <= low) return;
        int pivotIndex = partition(nums, low, high);
        quickSort(nums, low, pivotIndex - 1);
        quickSort(nums, pivotIndex + 1, high);
    }
private:
    int partition(vector<int>& nums, int low, int high) {
        int pivotVal = nums[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (nums[j] < pivotVal) {
                i++;
                swap(nums[i], nums[j]);
            }
        }
        i++;
        swap(nums[i], nums[high]);
        return i;
    }
};
```

Time Complexity: O(n log n)

2. **Non Repeating Character:**

```
class Solution {
public:
    char nonRepeatingChar(string& str) {
        unordered_map<char, int> countMap;
        for (int i = 0; i < str.size(); i++) {
            countMap[str[i]]++;
        }
        for (int i = 0; i < str.size(); i++) {
            if (countMap[str[i]] == 1) {
                return str[i];
            }
        }
        return '$';
    }
};
```

Time Complexity: O(n)

3. **Bubble Sort:**

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

Time Complexity: O(n^2)

4. **Kth Largest Element:**

```
class Solution {
public:
    vector<int> kLargest(vector<int>& nums, int k) {
        sort(nums.begin(), nums.end(), greater<>());
        vector<int> result;
        for (int i = 0; i < k; i++) {
            result.push_back(nums[i]);
        }
        return result;
    }
};
```

Time Complexity: O(n log n)

5. **Form Largest Number:**

```
class Solution {
public:
    string printLargest(vector<string>& nums) {
        sort(nums.begin(), nums.end(), compare);
        if (nums[0] == "0") {
            return "0";
        }
        string result = "";
        for (const string& num : nums) {
            result += num;
        }
        return result;
```

```
    }

private:
    static bool compare(const string& a, const string& b) {
        return a + b > b + a;
    }
};
```

Time Complexity: O(n log n)

6. **Edit Distance:**

```
class Solution {
public:
    int editDistance(string str1, string str2) {
        int len1 = str1.length();
        int len2 = str2.length();
        vector<vector<int>> dp(len1 + 1, vector<int>(len2 + 1));
        for (int i = 0; i <= len1; i++) {
            dp[i][0] = i;
        }
        for (int j = 0; j <= len2; j++) {
            dp[0][j] = j;
        }
        for (int i = 1; i <= len1; i++) {
            for (int j = 1; j <= len2; j++) {
                if (str1[i - 1] == str2[j - 1]) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = min({
                        dp[i - 1][j] + 1,
                        dp[i][j - 1] + 1,
                        dp[i - 1][j - 1] + 1
                    });
                }
            }
        }
        return dp[len1][len2];
    }
};
```

Time Complexity: O(m*n)