Day - 5 18/11/24

1. **Bubble Sort**

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

Time Complexity: O(n^2)

2. **Quick Sort:**

```
Quick Sort
class Solution {
  public:
    void quickSort(vector<int>& arr, int low, int high) {
        if(high<=low) return;
        int pivot = partition(arr,low,high);
        quickSort(arr,low,pivot-1);
        quickSort(arr,pivot+1,high);
    }
  public:
    int partition(vector<int>& arr, int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
        for(int j=low;j<high;j++){
            if(arr[j]<pivot){
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        i++;
        int temp = arr[i];
        arr[i] = arr[high];
        arr[high] = temp;
        return i;
    }
};
```

Time Complexity: O(n log n)

3. **Non Repeating Character:**

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

Time Complexity: O(n)

4. **Edit Distance:**

```
class Solution {
public:
    int editDistance(string s1, string s2) {
        int m = s1.length();
        int n = s2.length();
        vector<vector<int>> dp(m + 1, vector<int>(n + 1));
        for (int i = 0; i <= m; i++) {
            dp[i][0] = i;
        }
        for (int j = 0; j <= n; j++) {
            dp[0][j] = j;
        }
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (s1[i - 1] == s2[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[m][n];
    }
};
```

Time Complexity: O(m*n)

5. **Kth Largest Element:**

K Largest Elements

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

Time Complexity: O(n log n)

6. **Form Largest Number:**

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

Time Complexity: O(n log n)