Day - 2 : 12/11/2024

1. **Anagram:**

```
class Solution {
public:
    bool isAnagram(string s, string t) {
        if (s.length() != t.length()) return false;
        sort(s.begin(), s.end());
        sort(t.begin(), t.end());
        for (int i = 0; i < s.length(); i++) {
            if (s[i] != t[i]) return false;
        }
        return true;
    }
};
```

Time Complexity: O(n log n)

2. **Row with Max 1's:**

```
class Solution {
public:
    int rowWithMax1s(vector<vector<int>>& mat) {
        int row = -1, max1s = 0;
        int n = mat.size(), m = mat[0].size();
        int j = m - 1;

        for (int i = 0; i < n; i++) {
            while (j >= 0 && mat[i][j] == 1) {
                row = i;
                j--;
            }
        }
        return row;
    }
};
```

Time Complexity: O(m *n)

3. **Longest Consecutive Subsequence:**

```
class Solution {
public:
    int findLongestConseqSubseq(vector<int>& arr) {
        if (arr.empty()) return 0;
```

```
        sort(arr.begin(), arr.end());
        int count = 1, maxi = 1;

        for (int i = 1; i < arr.size(); i++) {
            if (arr[i] == arr[i - 1]) continue;
            if (arr[i] == arr[i - 1] + 1) {
                count++;
            } else {
                maxi = max(maxi, count);
                count = 1;
            }
        }

        return max(maxi, count);
    }
};
```

Time Complexity: O(n log n)

4. **<u>Longest Palindromic Substring:</u>**

```
class Solution {
public:
    string longestPalindrome(string s) {
        int start = 0, maxLen = 1;

        for (int i = 0; i < s.length(); i++) {
            expandFromCenter(s, i, i, start, maxLen);
            expandFromCenter(s, i, i + 1, start, maxLen);
        }

        return s.substr(start, maxLen);
    }

private:
    void expandFromCenter(const string& s, int left, int right, int& start, int& maxLen) {
        while (left >= 0 && right < s.length() && s[left] == s[right]) {
            if (right - left + 1 > maxLen) {
                start = left;
                maxLen = right - left + 1;
            }
            left--;
            right++;
        }
    }
};
```

Time Complexity: O(N^2)

5. **Rat in a Maze:**

```
class Solution {
public:
    bool solveMaze(vector<vector<int>>& maze) {
        vector<vector<int>> solution(maze.size(), vector<int>(maze[0].size(), 0));
        return solve(maze, 0, 0, solution);
    }
private:
    bool solve(vector<vector<int>>& maze, int x, int y, vector<vector<int>>& solution) {
        int n = maze.size();
        int m = maze[0].size();
        if (x == n - 1 && y == m - 1 && maze[x][y] == 1) {
            solution[x][y] = 1;
            printSolution(solution);
            return true;
        }

        if (isSafe(maze, x, y)) {
            solution[x][y] = 1;
            if (solve(maze, x + 1, y, solution)) return true;
            if (solve(maze, x, y + 1, solution)) return true;
            solution[x][y] = 0;
        }
        return false;
    }
    bool isSafe(vector<vector<int>>& maze, int x, int y) {
        return (x >= 0 && x < maze.size() && y >= 0 && y < maze[0].size() && maze[x][y] == 1);
    }
    void printSolution(vector<vector<int>>& solution) {
        for (const auto& row : solution) {
            for (int cell : row) {
                cout << cell << " ";
            }
            cout << endl;
        }
    }
};
```

Time Complexity: O(2^n*m)