|  |  |  |  |
| --- | --- | --- | --- |
| A logo of a university  Description automatically generated | **School of Engineering & Technology** | | |
| **Department: SOET** | **Session: 2023-2024** |
| **Program: B. Tech Computer Science Engineering** | **Semester: III** |
| **Course Code: ENCS253** | **Number of students:** |
| **Course Name:** **Data Structure** | **Faculty: Ms. Suman** |
| **Lab Date:** | **Batch:** |

**Lab Sheet 2**

**Aim: Implementing Advanced Array and String Operations with Complexity Analysis**

**Objective:**

The objective of this assignment is to delve deeper into array and string data structures, perform advanced operations, and analyze their time and space complexities.

**Problem Description**

1. Multi-dimensional Arrays:
   * Implement a two-dimensional array and perform operations such as row-wise and column-wise insertion, deletion, and traversal.
   * Implement a program to find the transpose of a given matrix.
2. Advanced String Operations:
   * Implement string pattern matching algorithms (e.g., Knuth-Morris-Pratt algorithm).
   * Write a program to perform string compression using Run Length Encoding (RLE).
3. Complexity Analysis:
   * Analyze the time and space complexity of the multi-dimensional array operations and the string algorithms.
   * Provide best, average, and worst-case analyses for each implemented algorithm.

**Instructions**

1. Multi-dimensional Array Implementation:
   * Create a class TwoDimensionalArray with methods for row-wise and column-wise insertion, deletion, and traversal.
   * Write a function transposeMatrix that takes a matrix as input and returns its transpose.
2. String Pattern Matching and Compression:
   * Create a class StringAlgorithms with methods for the Knuth-Morris-Pratt pattern matching algorithm.
   * Write a function runLengthEncoding that takes a string as input and returns its RLE compressed form.
3. **Complexity Analysis**:
   * Write a report analyzing the time and space complexity of the implemented algorithms using Big O, Omega, and Theta notations.
   * Include best, average, and worst-case analyses for the algorithms.

**Submission Guidelines**

1. Your code should be well-commented and adhere to standard coding practices.
2. Submit your code as a GitHub repository. Include a README.md file with instructions on how to run your code.
3. Ensure your repository includes:
   * Implementation files
   * Test cases
   * A report on complexity analysis
4. Submit the GitHub repository link on Moodle by the due date.

**Test Cases**

| Test Case | Input | Expected Output | Desired Output |
| --- | --- | --- | --- |
| 2D Array Operations | insertRow([1, 2]), insertColumn([3, 4]), deleteRow(0), traverse() | [[3, 4]] | [[3, 4]] |
| Transpose Matrix | [[1, 2], [3, 4]] | [[1, 3], [2, 4]] | [[1, 3], [2, 4]] |
| KMP Pattern Matching | "abxabcabcaby", "abcaby" | 6 | 6 |
| Run Length Encoding | "aaabbbcccaaa" | "3a3b3c3a" | "3a3b3c3a" |

**Expected Outcomes**

* Proficiency in handling multi-dimensional arrays and performing advanced operations.
* Understanding and implementation of string pattern matching algorithms.
* Ability to compress strings using RLE.
* Skill in analyzing algorithm complexities and providing comprehensive analysis reports.

**Evaluation Rubrics**

| Criteria | Excellent (4) | Good (3) | Satisfactory (2) | Needs Improvement (1) |
| --- | --- | --- | --- | --- |
| Implementation | Correct and efficient | Mostly correct | Partially correct | Incorrect |
| Code Quality | Well-commented, clean | Some comments, mostly clean | Few comments, some issues | Poorly commented, messy |
| Analysis | Thorough and accurate | Mostly accurate | Some inaccuracies | Inaccurate |
| Submission | On time, complete | Slightly late, complete | Late, incomplete | Not submitted |