**National University of Computer and Emerging Sciences**



Laboratory Manual

for

Data Structures Lab

|  |  |
| --- | --- |
| Lab Instructor(s) | Anosha Khan |
| Section | BCS-3G |
| Semester | Fall 2024 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

* **Huffman Coding**

### ****Lab Question: Huffman Coding Implementation****

**Objective:**  
Implement the basic Huffman Coding algorithm to compress a given string based on character frequency.

**Problem Statement:**  
You are tasked with implementing a Huffman Coding algorithm to compress a given text. Follow these steps:

1. **Input:**  
   Accept a string from the user.  
   Example: "compression"
2. **Frequency Count:**  
   Scan the string and count the frequency of each character. Use this information to prioritize characters for building the Huffman code tree.
3. **Tree Construction:**  
   Implement the following steps to build the Huffman tree:
   * Create a minHeap (priority queue) to store nodes of the Huffman tree.
   * For each character wi in the frequency table:
     + Create a tree T for wi where the node stores the character and its frequency.
     + Insert the tree T into the minHeap.
   * While the minHeap has more than one node:
     + Extract the two nodes with the smallest frequencies (T1 and T2).
     + Merge these two trees into a new tree T3 where the root contains the sum of their frequencies.
     + Insert T3 back into the minHeap.
4. **Code Generation:**  
   Perform a traversal of the final Huffman tree to generate binary code words for each character.
5. **Compression:**  
   Use the Huffman codes to encode the input string into a compressed binary string.
6. **Output:**
   * Display the frequency table of characters.
   * Display the generated Huffman codes for each character.
   * Display the compressed binary string.

**Tasks:**

1. **Implement the minHeap Class:**
   * Write a class minHeap to handle insertion, deletion, and extraction of the minimum element.
2. **Build the Huffman Tree:**
   * Implement the tree construction steps described above.
3. **Generate Huffman Codes:**
   * Write a function to traverse the tree and generate Huffman codes for each character.
4. **Test Your Implementation:**
   * Run your program with the input string "compression".
   * Verify the frequency table, Huffman codes, and the compressed binary string.