Nama : Muhammad Anshar

Nim : 2111082024

Sistem Terdisribusi/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ervan.compres;

import java.util.HashMap;

import java.util.Map;

import java.util.PriorityQueue;

/\*\*

\*

\* @author FX506PE

\*/

public class StaticHuffmanCompression {

public static Map<Character, String> buildHuffmanCodes(String text) {

Map<Character, Integer> frequencyMap = new HashMap<>();

for (char c : text.toCharArray()) {

frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);

}

PriorityQueue<HuffmanNode> priorityQueue = new PriorityQueue<>();

for (char c : frequencyMap.keySet()) {

priorityQueue.add(new HuffmanNode(c, frequencyMap.get(c)));

}

while (priorityQueue.size() > 1) {

HuffmanNode left = priorityQueue.poll();

HuffmanNode right = priorityQueue.poll();

HuffmanNode parent = new HuffmanNode('\0', left.frequency + right.frequency);

parent.left = left;

parent.right = right;

priorityQueue.add(parent);

}

HuffmanNode root = priorityQueue.poll();

Map<Character, String> huffmanCodes = new HashMap<>();

buildHuffmanCodesRecursive(root, "", huffmanCodes);

return huffmanCodes;

}

private static void buildHuffmanCodesRecursive(HuffmanNode node, String code, Map<Character, String> huffmanCodes) {

if (node == null) {

return;

}

if (node.data != '\0') {

huffmanCodes.put(node.data, code);

}

buildHuffmanCodesRecursive(node.left, code + "0", huffmanCodes);

buildHuffmanCodesRecursive(node.right, code + "1", huffmanCodes);

}

public static String compress(String text, Map<Character, String> huffmanCodes) {

StringBuilder compressed = new StringBuilder();

for (char c : text.toCharArray()) {

compressed.append(huffmanCodes.get(c));

}

return compressed.toString();

}

public static String decompress(String compressed, Map<Character, String> huffmanCodes) {

Map<String, Character> reverseMap = new HashMap<>();

for (Map.Entry<Character, String> entry : huffmanCodes.entrySet()) {

reverseMap.put(entry.getValue(), entry.getKey());

}

StringBuilder decompressed = new StringBuilder();

int i = 0;

while (i < compressed.length()) {

int j = i + 1;

while (j <= compressed.length() && !reverseMap.containsKey(compressed.substring(i, j))) {

j++;

}

decompressed.append(reverseMap.get(compressed.substring(i, j)));

i = j;

}

return decompressed.toString();

}

public static void main(String[] args) {

String text = "this is an example for huffman encoding";

Map<Character, String> huffmanCodes = buildHuffmanCodes(text);

String compressedData = compress(text, huffmanCodes);

System.out.println("Original Data: " + text);

System.out.println("Compressed Data: " + compressedData);

String decompressedData = decompress(compressedData, huffmanCodes);

System.out.println("Decompressed Data: " + decompressedData);

}

}

Penjelasan dari program di atas adalah?

1. Package dan Komentar:

- Kode dimulai dengan komentar header yang memberikan informasi tentang tujuan dan penggunaan kode tersebut.

2. Import Statements:

- Beberapa pustaka (libraries) dari Java diimpor, termasuk `java.util.HashMap`, `java.util.Map`, dan `java.util.PriorityQueue`.

3. Kelas `StaticHuffmanCompression`:

- Kelas ini merupakan kelas utama yang berisi metode-metode untuk melakukan kompresi dan dekompresi menggunakan algoritma Huffman statis.

4. Metode `buildHuffmanCodes`:

- Metode ini menerima teks sebagai input dan mengembalikan peta (map) yang berisi karakter sebagai kunci dan kode Huffman sebagai nilai.

- Menggunakan `frequencyMap` untuk menghitung frekuensi kemunculan setiap karakter.

- Membangun pohon Huffman menggunakan `PriorityQueue` untuk menggabungkan simpul dengan frekuensi terkecil.

- Memanggil metode rekursif `buildHuffmanCodesRecursive` untuk membangun tabel kode Huffman.

5. Metode `buildHuffmanCodesRecursive`:

- Metode rekursif untuk menyusun tabel kode Huffman dari pohon Huffman.

6. Metode `compress`:

- Menggunakan tabel kode Huffman untuk mengompresi teks input menjadi representasi biner.

7. Metode `decompress`:

- Melakukan dekompresi dari representasi biner ke teks awal menggunakan tabel kode Huffman.

8. Metode `main`:

- Memperlihatkan penggunaan metode-metode di atas.

- Mengonversi teks ke tabel kode Huffman, mengompresi teks, menampilkan teks asli, teks terkompresi, dan hasil dekompresi.