

FILE COMPRESSOR USING HUFFMEN CODING(JAVA)
BY
slvn_yaswanth

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1	Introduction	3
2	Background Study	4
3	Problem Definition	5
4	Objective	6
5	Methodology/Procedure	7
6	Results and Discussion	10
7	Conclusion and Future Scope	11
	References	12

Chapter -1

Introduction

A file compressor is a zipper which will zip the multiple files by compressing so that the zipped stack size is less than original files size and we can share whole stack once, so that while unzipping they can get same files without missing the sequence of data.

Chapter -2

Background Study

HASHMAP

HashMap class allow *to store key and value pair*, where keys should be unique.

If you try to insert the duplicate key, it will replace the element of the corresponding key. It is easy to perform operations using the key index like updation, deletion, etc. HashMap class is found in the *java. util* package.

HashMap in Java is like the legacy Hashtable class, but it is not synchronized. It allows us to store the null elements as well, but there should be only one null key. Since Java 5, it is denoted as `HashMap<K, V>`, where K stands for key and V for value. It inherits the AbstractMap class and implements the Map interface.

<https://www.javatpoint.com/java-hashmap>

ARRAYLIST

Java ArrayList class uses a *dynamic array* for storing the elements. It is like an array, but there is *no size limit*. We can add or remove elements anytime. So, it is much more flexible than the traditional array. It is found in the *java. util* package. It is like the Vector in C++.

The ArrayList in Java can have the duplicate elements also. It implements the List interface so we can use all the methods of the List interface here. The ArrayList maintains the insertion order internally.

It inherits the AbstractList class and implements List interface.

<https://www.javatpoint.com/java-arraylist>

MIN_PRIORITY_QUEUE

A **priority queue** is a queue which instead of being FIFO is "Best Out." "Best" is defined by a priority. For a typical priority queue, low priority numbers are removed first. That may seem backwards but think of "you are our number one priority!" That is better than being their number two or three priority.

There are hundreds of applications of priority queues. They come up in computer systems (high-priority print jobs, various priority levels of jobs running on a time-sharing system, etc.). They are used in finding shortest paths and other search problems. And a priority queue gives you an effortless way to sort: put everything into the priority queue, then take them out one at a time.

They come out in sorted order.

There are two flavors of priority queues: **min-priority queues** and **max-priority queues**. They vary only in whether a low or high priority number corresponds to "better" priority.

Min Priority Queue is a data structure which manage a list of keys(values). And gives priority to element with minimum value.

<https://www.cs.dartmouth.edu/~scot/cs10/lectures/14/14.html#:~:text=A%20min%2Dpriority%20queue%20implemented%20by%20a%20sorted%20ArrayList&text=java%20gives%20this%20implementation.,size%2D1%20of%20the%20ArrayList%20>.

HUFFMAN_CODING

Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code, and the least frequent character gets the largest code.

<https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/>

Chapter -3

Problem Definition

Now a days we are not getting a constant bandwidth when we need at situation at particular place. Also, even though now a days storage is cheap storage (HDD) compared to SSD, the unused files but which are helpful in future are increasing day by day. So, to store files and share files in efficient manner we need one squeezing tool so that we can store our files for longer and share files faster even in less bandwidth, so that when we un squeeze that we need to get exact data before what we compressed without compromising so that is why introducing file compressor.

Chapter -4

Objective

To share multiple files in less bandwidth

To share the stack of all files at once

To increase the storage efficiency

To store the unused files for long time in a less storage

No compromise of data after extracting the compressed files

Chapter -5

Methodology/Procedure

So, we need to decrease the size of the data so that even when we extract that we will get same data.

Generally, ASCII code for characters is large. So, if we make a bit sequence for every unique character that are present in data so that take less space than ASCII, we can achieve our goal.

So, to generate a sequence here we use Huffman tree so that all unique characters present in leaf nodes, so by traversing this tree we can generate a binary sequence for each character by appending zero for left child and one for right child while traversing.

We will create Huffman tree such that by adding least min frequencies of a characters to become root and we repeat that adding roots to roots up to N Unique characters.

To find every time min frequency in list we use minpriority queue crated using list. So that it will provide all min values in order.

So

first find the frequency off each character and convert them to nodes such that each node represent frequency

of each character.

compress

Getbytes ()

we will create a hash map saying frequency for each character with that as a node we will create minpriorityqueue and return queue

createzip

createHuffmanTree

with that returned minpriorityqueue we create Huffman tree by taking minpriorityqueue as input

gethuffcodes

after creating Huffman tree, we are traversing the Huffman tree up to the leaf node where all leaf nodes represent the unique characters and appending bit sequence of character into HashMap by giving zero for left traverse and one for right traverse.

zipByteWithCodes

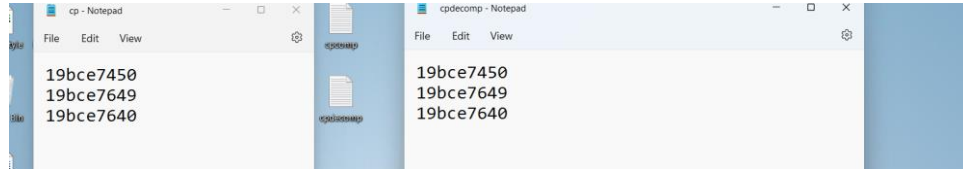
we are appending the original data with respective generated bit sequence respect to their character.

after all that we out streaming the text file with HashMap of Huffman codes and returned data from zipByteWithCodes.

decompress

while decompressing by using the hasp map and appended sequence bytes we will get actual characters and these characters are added to array list and list is printed

Results and Discussion



Here the cp file is the original file and cpcomp is the compressed file and cpdecomp is the extracted file

Chapter -7

Conclusion and Future Scope

So, from this we can compress the file such that we can share multiple files in less bandwidth, share the stack of all files at once, increase the storage efficiency, store the unused files for long time in a less storage without any compromise of data after extracting the compressed files

Future Scope

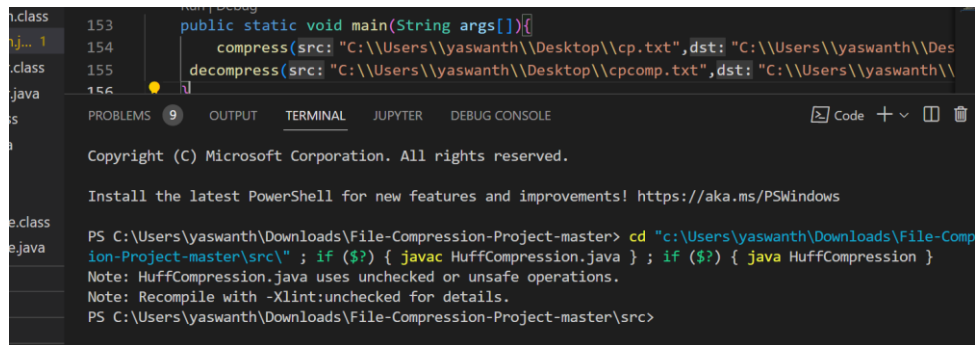
Further Given a string of alphabetic characters, you can compress the size of its bit sequence to obtain a binary sequence that has size significantly lesser than the original, this new sequence then can be sent through a network or saved in file so that it can be decompressed whenever required.

References (Sample)

1. Acharya, Tinku, Lina Karam & Francescomaria Marino 2000, 'Compression of color images based on a 2-dimensional discrete wavelet transform yielding a perceptually lossless image', U.S. Patent no. 6, pp. 154-493.
2. Amir Said & Pearlman, WA 1996, 'A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees', IEEE Trans. Circuit and systems for Video Technology, vol. 6, no. 3, pp. 243- 250.
3. Andreopoulos, I, Karayianris, YA & Stouruaitis, T 2000, 'A hybrid image compression algorithm based on fractal coding and wavelet transform', IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353).
4. Asad Islam & Pearlman 1996, 'An embedded and efficient low-complexity, hierarchical image coder', Visual Communication and Image processing'99proceedings of SPIE, vol. 3653, pp. 294-305.
5. Ates, H & Orchard 2009, 'Spherical Coding Algorithm for Wavelet Image Compression', Proc. IEEE Int. Conf. Image Processing, vol. 18, no. 5, pp. 1015-1024
6. www.ieeeexplore.com
7. www.mathworld.com

Appendix – A

Coding and Snapshot



```

153 public static void main(String args[]) {
154     compress(src: "C:\\Users\\yaswanth\\Desktop\\cp.txt", dst: "C:\\Users\\yaswanth\\Des
155     decompress(src: "C:\\Users\\yaswanth\\Desktop\\cpcomp.txt", dst: "C:\\Users\\yaswanth\\
156

```

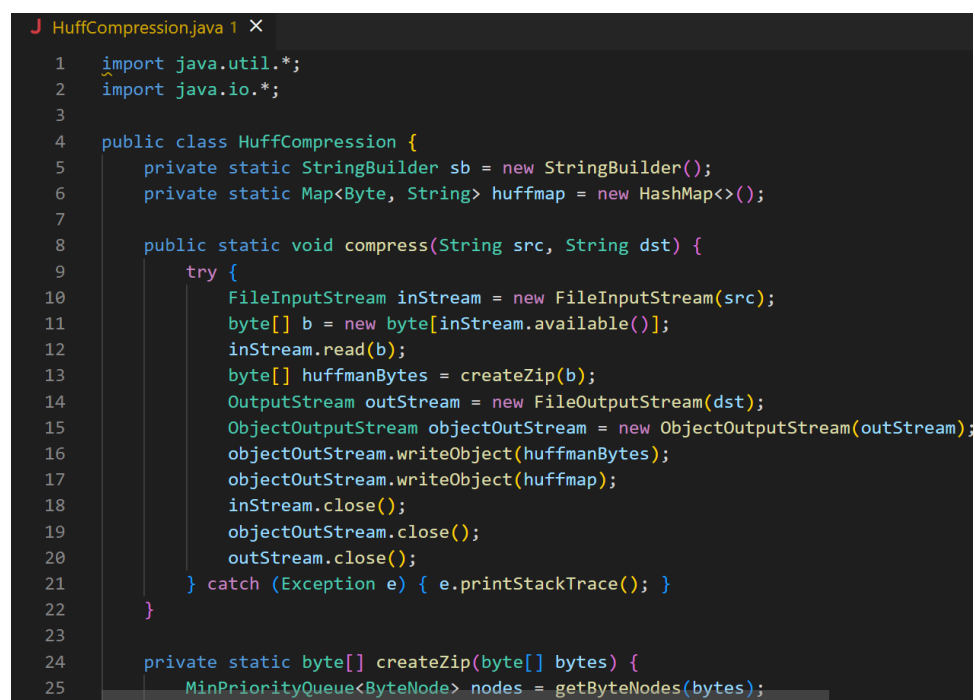
PROBLEMS 9 OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

PS C:\Users\yaswanth\Downloads\File-Compression-Project-master> cd "c:\Users\yaswanth\Downloads\File-Compression-Project-master\src\" ; if (\$?) { javac HuffmanCompression.java } ; if (\$?) { java HuffmanCompression }
Note: HuffmanCompression.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Users\yaswanth\Downloads\File-Compression-Project-master\src>

Here we can add the source and destination files to compress and decompress.



```

J HuffmanCompression.java 1 X
1  import java.util.*;
2  import java.io.*;
3
4  public class HuffmanCompression {
5      private static StringBuilder sb = new StringBuilder();
6      private static Map<Byte, String> huffmanmap = new HashMap<>();
7
8      public static void compress(String src, String dst) {
9          try {
10              FileInputStream inStream = new FileInputStream(src);
11              byte[] b = new byte[inStream.available()];
12              inStream.read(b);
13              byte[] huffmanBytes = createZip(b);
14              OutputStream outStream = new FileOutputStream(dst);
15              ObjectOutputStream objectOutStream = new ObjectOutputStream(outStream);
16              objectOutStream.writeObject(huffmanBytes);
17              objectOutStream.writeObject(huffmanmap);
18              inStream.close();
19              objectOutStream.close();
20              outStream.close();
21          } catch (Exception e) { e.printStackTrace(); }
22      }
23
24      private static byte[] createZip(byte[] bytes) {
25          MinPriorityQueue<ByteNode> nodes = getByteNodes(bytes);

```

The screenshot shows an IDE with a project named 'File-Compression-Project-master'. The Explorer view on the left shows the project structure with files like ByteNode.class, HuffmanEncoder.java, HuffmanTree.java, MinPriorityQueue.class, and Node.java. The main editor displays the code for HuffmanCompression.java, which includes a method to compress a list of characters into a byte array. The terminal window at the bottom shows a PowerShell prompt where the user has navigated to the project's source directory and executed the command 'javac HuffmanCompression.java'. The terminal output shows the compilation was successful, with a note that HuffmanCompression.java uses unchecked or unsafe operations.

```
136 list.add(0);
137 i += count;
138 }
139 byte b[] = new byte[list.size()];
140 for (int i = 0; i < b.length; i++)
141     b[i] = list.get(i);
142 return b;
143 }
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

PS C:\Users\yaswanth\Documents\File-Compression-Project-master\cp\src> cd "c:\Users\yaswanth\Documents\File-Compression-Project-master\cp\src\" ; if (\$?) { javac HuffmanCompression.java } ; if (\$?) { java HuffmanCompression }
Note: HuffmanCompression.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Users\yaswanth\Documents\File-Compression-Project-master\cp\src>