BLG439E COMPUTER PROJECT I

PROJECT 2

TRANSMISSION OF COMPRESSED TEXT FILES WITH HUFFMAN ALGORITHM

BARAN KAYA 150130032

KADİR ENES KARSLIOĞLU 150130047

M. ALİ OSMAN ATİK 150140804

November 2017

HUFFMAN ALGORITHM

Huffman coding is an algorithm for implementing data compression. Basic idea behind this algorithm is file compression. This coding related to fixed length and variable length encoding, prefix rules and Huffman tree.

HUFFMAN ENCODING

This technique works by creating a binary tree of nodes. At the start, all nodes are leaf nodes which include the frequency of the characters. These nodes contain character weight and links to two child nodes. It creates a tree which is Huffman tree. We used priority queue for building Huffman tree where the node with lowest frequency is given highest priority. These are steps:

- For each symbol, create a leaf node and add it to the priority order.
- When there are multiple nodes in the queue:
- To obtain two nodes, remove the node with the highest priority (lowest probability) twice.
- Create a new inner node that is equal to the sum of the two node children and possibly the two node possibilities.
- Add a new node for the queue.
- The remaining node is the root node and the tree is completed.

HUFFMAN DECODING

We would utilize this algorithm to achieve our text message from binary encoded text. In order to this, we should have the Huffman tree. We implemented this algorithm:

- We start from root and do following until a leaf is found.
- If current bit is 0, we move to left node of the tree.
- If next bit is 1, we move to the right node of the tree.
- If we encounter a leaf node during traversal, it would press the character of a particular leaf node.
- Then, proceed to repeat the encoded data starting from step 1.

SERVER

Data transmission simply tried on Bluetooth communication on a Python server. We used BluetoothSocket() methods of Python for connection:

- Server waits for a connection and accepts incoming request.
- Server receives 1024 bytes of data into buffer.
- Data in buffer forwarded to screen.
- In case of connection lost, sockets are closed.

ISSUES

- After compression with Huffman coding, we could not only save as file as bites, but also send as bites.
- Although we can create Huffman tree in encoding, we also could not dispatch this tree to another device in server.

USED LIBRARIES

Android.widget, android.view, android.OS for Android developments.

Java.io and java.util for Java software.

PARTS IMPLEMENTATION PER MEMBER

BARAN KAYA

Implementation of Huffman encoding (compression) of HCoder.

HCoder user interface for logging/file selection.

KADİR ENES KARSLIOĞLU

Implementation of Huffman decoding (compression) of HCoder.

HCoder communication with the HServer.

M. ALİ OSMAN ATİK

Implementation of Huffman encoding/decoding of HServer.

Other server functions that are described above.

RESOURCES

[1] http://www.codemiles.com/java/standard-huffman-coding-t6.html

[2] https://www.codemiles.com/java/huffman-compression-decompression-t96.html

[3] https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Huffman.java.html

[4] https://github.com/tarunsharma1/Text-compression-using-Huffman-coding

[5] https://gist.github.com/ahmedengu/aa8d85b12fccf0d08e895807edee7603

[6]http://sourcecodesforfree.blogspot.com.tr/2013/05/16-text-compression-usinghuffman.html

[7] https://stackoverflow.com/questions/30253118/huffman-decompression

[8] https://stackoverflow.com/questions/43421273/huffman-code-writing-bits-to-a-file-forcompression

[9] https://stackoverflow.com/questions/25843949/sending-string-from-pc-to-android-appusing-bluetooth-dongle

[10] https://github.com/tfg13/LanXchange