Programming Project – Huffman encoding-decoding

Advanced Data Structures – COP5536
Princepreet Singh Bhatti | UFID: 31116116

Problem Statement: Software giant Toggle recently bought video streaming site MyTube. Now MyTube needs to send its data to Toggle server. As enormous amount of data will be transferred, they decided to use Huffman coding to reduce data size. The aim is to implement the working model of Huffman encoder and decoder.

1. **Huffman Coding:** In this part of the project, I implemented the three types of data structures that can store the input data in form of a heap. The idea is to be able to get the minimum element from the heap data structure and using it to build the Huffman tree.

   i) Binary Heap: I implemented the binary heap data structure using a Node data structure. The node data structure has six fields: parent, leftchild, rightchild, isleaf, key and value; where key is the input number and the value is the frequency of that number. Following functions are included in the code. Following table explains the prototype of functions used for implementing the binary heap

FUNCTION	CLASS	PROTOTYPE *(returnType FunctionName(arguments))*
BinaryHeap	BinaryHeap	Class Constructor
getHeap	BinaryHeap	BinaryNode[] getHeap ()
insert	BinaryHeap	void insert (BinaryNode n)
findSmallestChild	BinaryHeap	int findSmallestChild (int s, int e)
restoreHeap	BinaryHeap	void restoreHeap (int i)
removeMin	BinaryHeap	BinaryNode removeMin ()
getSize	BinaryHeap	int getSize()
swap	BinaryHeap	void swap (int p, int l)

ii) 4-way cache optimized heap: I implemented the 4-way heap data structure using a DNode data structure. The DNode data structure has six fields: parent, leftchild, rightchild, isleaf, key and value; where key is the input number and the value is the frequency of that number. Following functions are included in the code. Following table explains the prototype of functions used for implementing the 4-way heap:

FUNCTION	CLASS	PROTOTYPE *(returnType FunctionName(arguments))*
DHeap	DHeap	Class Constructor
getdHeap	DHeap	DNode[] getdHeap()
insert	DHeap	void insert (DNode n)
findSmallestChild	DHeap	int findSmallestChild(int s, int e)
restoreHeap	DHeap	void restoreHeap (int i)
removeMin	DHeap	DNode removeMin()
getSize	DHeap	int getSize()

iii) Pairing Heap: I implemented the 4-way heap data structure using PNode data structure. The PNode data structure has five fields: nextSibling, backLink, lChild, leftChild, rightChild, isLeaf, key and value; where key is the input number and the value is the frequency of that number. Following functions are included in the code. Following table explains the prototype of functions used for implementing the 4-way heap:

FUNCTION	CLASS	PROTOTYPE *(returnType FunctionName(arguments))*
DHeap	PairingHeap	Class Constructor
insert	PairingHeap	PNode insert (PNode newNode)
meldOperation	PairingHeap	PNode meldOperation (PNode n1, PNode n2)
twoPassSchemeJoin	PairingHeap	PNode twoPassSchemeJoin(PNode sibling)
increaseMinTreeArraySize	PairingHeap	PNode[] increaseMinTreeArraySize (PNode[ ] arr, int pos)
removeMin	PairingHeap	PNode removeMin ( )

Performance Analysis:

2. **Encoder:** encoder.java class contains the main class for the encoder program.

FUNCTION	CLASS	PROTOTYPE *(returnType FunctionName(arguments))*
encodeInput	encoder	void encodeInput(LinkedHashMap<Integer, String> codeMap, File file)
huffmanCodeGenerator	encoder	LinkedHashMap<Integer,String> huffmanCodeGenerator(DNode n, String hCode, LinkedHashMap m, BufferedWriter w)
runEncoder	encoder	void runEncoder(String path)
main	encoder	void main(String [] args)

3. **Decoder:** decoder.java contains the main function for decoder program.

FUNCTION	CLASS	PROTOTYPE *(returnType FunctionName(arguments))*
generateTree	decoder	Node generateTree(File file)
generateTreefromCode	decoder	Node generateTreefromCode(Node root, String codePair, int pos, int posMax)
getDecodedMessageString	decoder	void getDecodedMessageString(File file2, File file3, Node htree)
decodeData	decoder	String decodeData(Node htree, String message, BufferedWriter bw)
runDecoder	decoder	void runDecoder(String[] arr)
main	decoder	void main(String[] args)