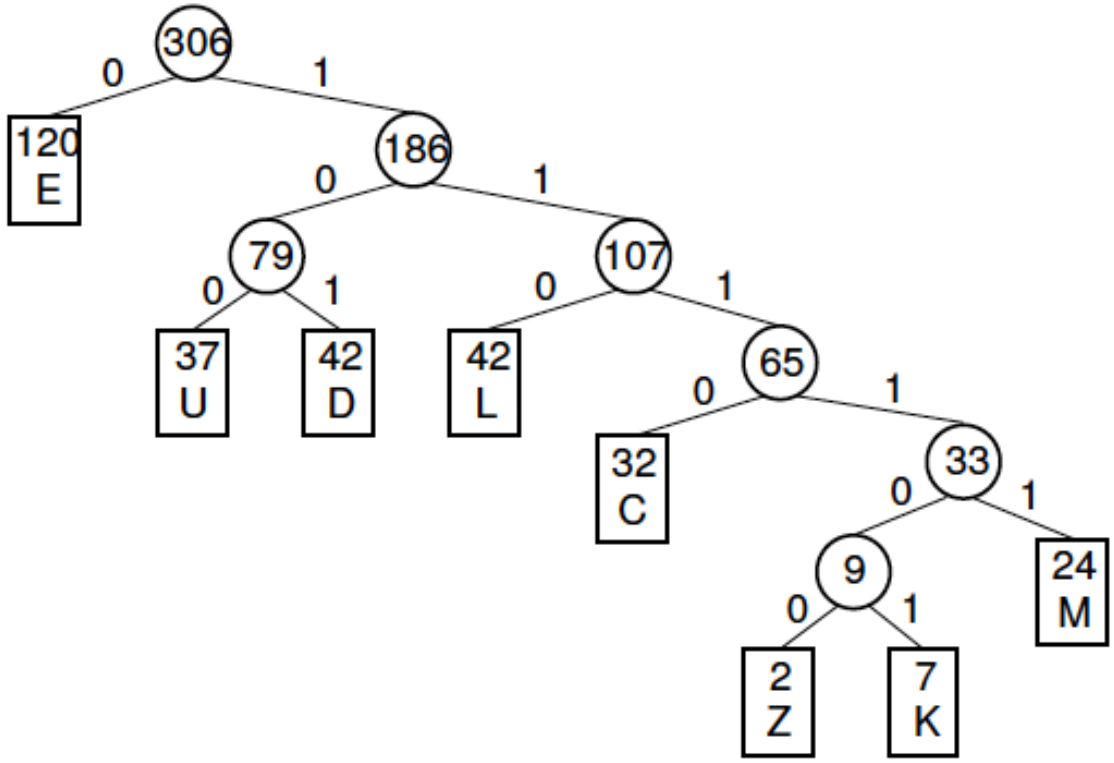
Description:

Huffman coding is a method of data compression that is independent of the data type, that is, the data could represent an image, audio or spreadsheet. This compression scheme is used in JPEG and MPEG-2. Huffman coding works by looking at the data stream that makes up the file to be compressed. Those data bytes that occur most often are assigned a small code to represent them (certainly smaller then the data bytes being represented). Data bytes that occur the next most often have a slightly larger code to represent them. This continues until all of the unique pieces of data are assigned unique code words. For a given character distribution, by assigning short codes to frequently occurring characters and longer codes to infrequently occurring characters, Huffman's minimum redundancy encoding minimizes the average number of bytes required to represent the characters in a text. Static Huffman encoding uses a fixed set of codes, based on a representative sample of data, for processing texts. Although encoding is achieved in a single pass, the data on which the compression is based may bear little resemblance to the actual text being compressed. .

Data Structures:

The data structure of Huffman codes and its application to efficient encoding and decoding of Huffman codes are studied in detail. The tree structure is *by class for each node to collect the node char and its code ,then we use a data structure "priorityqueue" as a sorting technique to*

get min two nodes .another data structure using hash map and treemap to store codes of huffman and its frequencies.



Algorithms ,Complexity and Header Format:

Algorithm:

Complexity:

The time complexity of the Huffman algorithm is **O(nlogn)**. Using a heap to store the weight of each tree, each iteration requires **O(logn)** time to

determine the cheapest weight and insert the new weight. There are **O(n)** iterations, one for each item.

Header Format:

You must store some initial information in the compressed file that will be used by the decompression. Basically you must store the tree used to compress the original file. This tree is used by the decompression program.

There are several alternatives for storing the tree :

encoding technique:

1. Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap. Initially, the least frequent character is at root)

2. Extract two nodes with the minimum frequency from the min heap.

3. Create a new internal node with frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.

4. Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

- **header file as following :**
- store the character then huffman code and its frequency (frequency/length of the string) so the result between 0 and 1.