DATA STRUCTURES MINI PROJECT- MESSAGE ENCRYPTION USING HUFFMAN CODING (GREEDY ALGORITHM)

AIM:

This project aims to encrypt the given message using Huffman coding (Greedy algorithm) to securely protect data.

ENVIRONMENT:

This project is developed using C language and GCC has been used to test the application.

DESCRIPTION:

Encryption is the process of changing information in such a way as to make it unreadable by anyone except those possessing special knowledge (usually referred to as a "key") that allows them to change the information back to its original, readable form. This project uses priority queue using min heap and trees to encrypt message using Huffman coding (Greedy Algorithm) thereby protecting data. The encrypted message can be decrypted in future by using the tree that maps each key to a binary value.

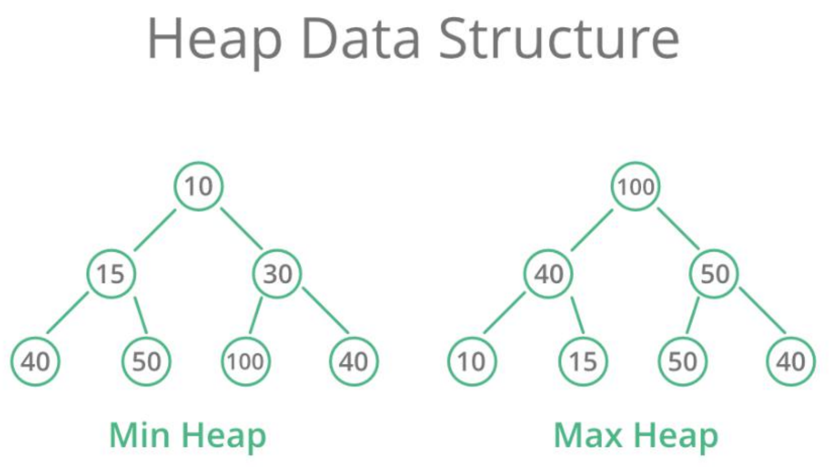
DATA STRUCTURES USED AND THEIR DATA FLOW LAYOUTS:

Binary Heap:

A Binary Heap is a Binary Tree with following properties.

1) It’s a complete tree (All levels are completely filled except possibly the last level and the last level has all keys as left as possible). This property of Binary Heap makes them suitable to be stored in an array.

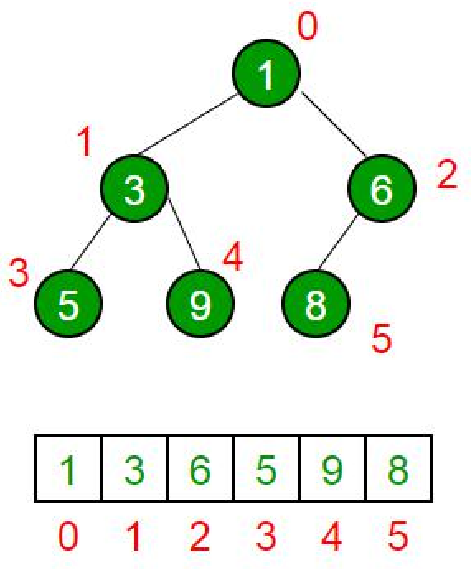
2) A Binary Heap is either Min Heap or Max Heap. In a Min Binary Heap, the key at root must be minimum among all keys present in Binary Heap. The same property must be recursively true for all nodes in Binary Tree. Max Binary Heap is similar to MinHeap.



Priority Queue Using Min Heap:

A priority queue acts like a queue in that items remain in it for some time before being dequeued. However, in a priority queue the logical order of items inside a queue is determined by their “priority”. Specifically, the highest priority items are retrieved from the queue ahead of lower priority items.

A priority queue implemented using a **binary heap**: **min heap**, in which the smallest key is always at the front, and the **max heap**, in which the largest key value is always at the front, will allow to enqueue or dequeue items in the order of *O*(log *n*).



(Priority Queue using Min Heap)

DATA FLOW LAYOUT:

MAIN FUNCTION

-Displays a menu driven program that takes input of characters and its frequency

-calls method to display the Huffman coded values for each character

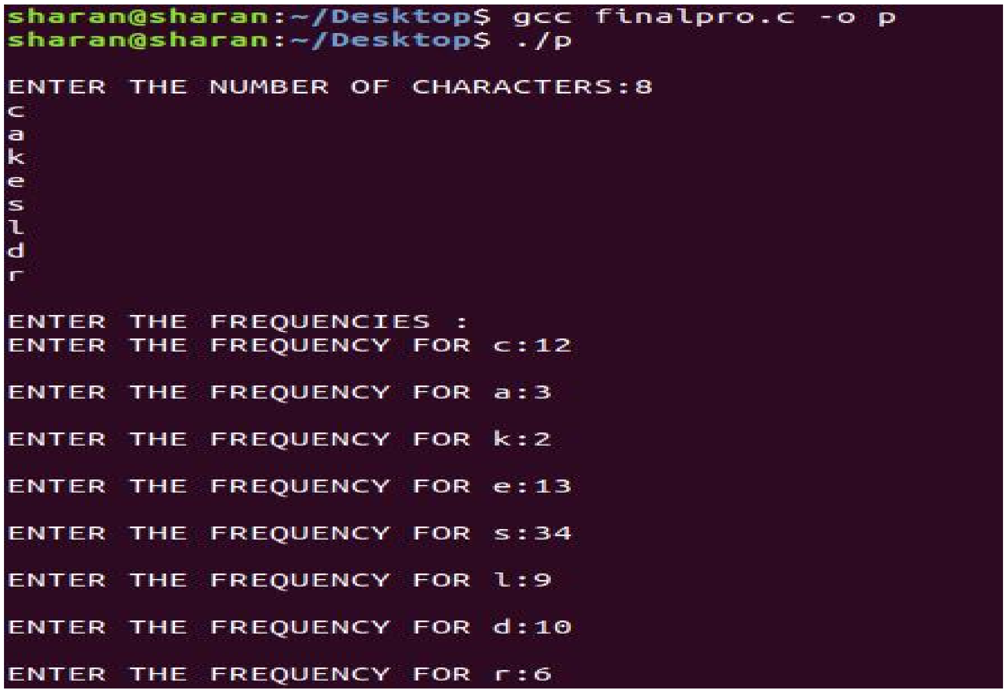
-takes input of the message and prints the encrypted message

HUFFMANIMPL.H

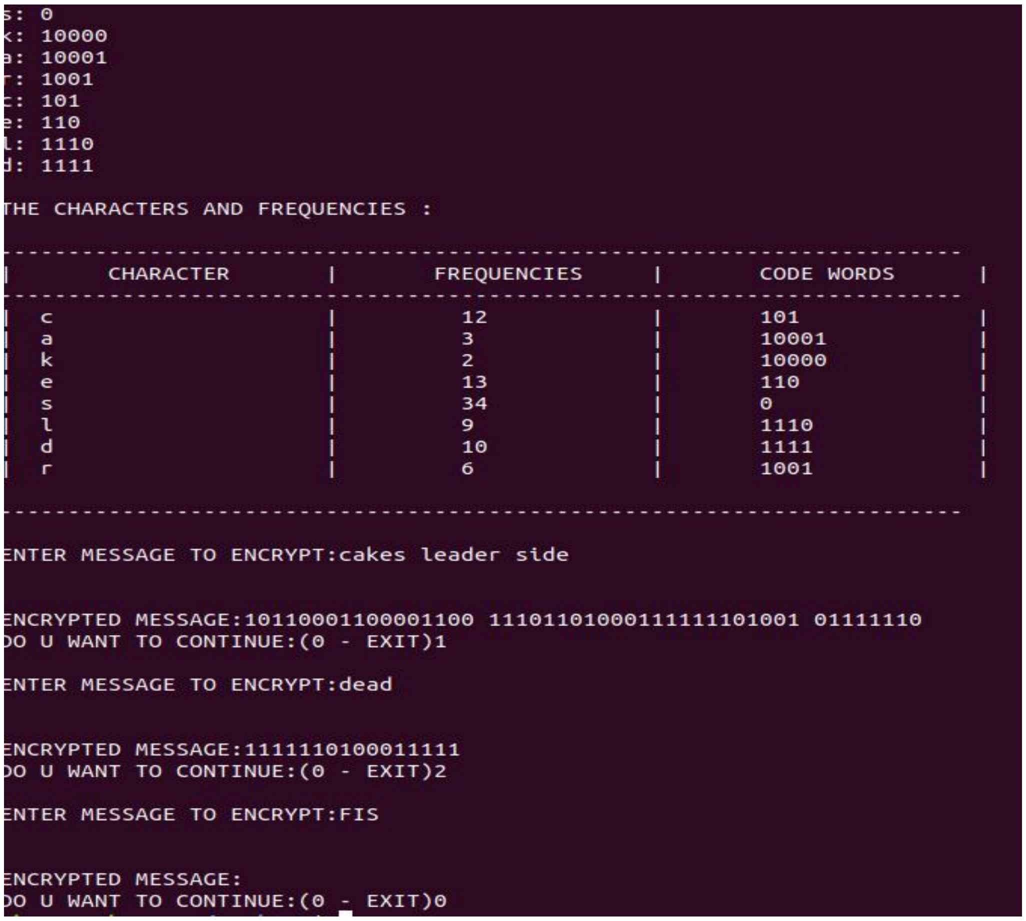
- Takes the frequency of characters from main and passes it to a Min Heap i.e. priority queue ,which is used in the later stages to produce the corresponding code representation for the characters passed (based on the assigned frequencies).

- **Huffmancodes -**will call build Huffman tree ( )
- **BuildHuffmanTree() –** will call createAndBuildMinHeap ( ) , extractMin()& insertMinHeap ( )
- **CreateAndBuildMinHeap ()-**will call createMinHeap and RearrangeMinHeap ()
- **CreateMinHeap( )-** will create a min heap node
- **RearrangeHeap()-**will call itself as long as there is a violation in the property of the minheap and would swap the values using the swapNode().
- **extractMin()-** removes the smallest element from the heap and rearranges it.
- **InsertMinHeap ( ) –** insert the nodes into minheap
- **printCodes() -**traverses through the minheap to produce the codes and calls printArr ( )
- **printArr ()-**prints the codes and saves it into a twoD array

SNAPSHOTS OF OUTPUT:



```
sharan@sharan:~/Desktop$ gcc finalpro.c -o p
sharan@sharan:~/Desktop$ ./p

ENTER THE NUMBER OF CHARACTERS:8
c
a
k
e
s
l
d
r

ENTER THE FREQUENCIES :
ENTER THE FREQUENCY FOR c:12

ENTER THE FREQUENCY FOR a:3

ENTER THE FREQUENCY FOR k:2

ENTER THE FREQUENCY FOR e:13

ENTER THE FREQUENCY FOR s:34

ENTER THE FREQUENCY FOR l:9

ENTER THE FREQUENCY FOR d:10

ENTER THE FREQUENCY FOR r:6
```



```
s: 0
k: 10000
a: 10001
r: 1001
c: 101
e: 110
l: 1110
d: 1111

THE CHARACTERS AND FREQUENCIES :

-----------------------------------------------------------------------
|        CHARACTER        |        FREQUENCIES       |        CODE WORDS       |
-----------------------------------------------------------------------
|  c                      |          12              |        101              |
|  a                      |          3               |        10001            |
|  k                      |          2               |        10000            |
|  e                      |          13              |        110              |
|  s                      |          34              |        0                |
|  l                      |          9               |        1110             |
|  d                      |          10              |        1111             |
|  r                      |          6               |        1001             |

-----------------------------------------------------------------------

ENTER MESSAGE TO ENCRYPT:cakes leader side


ENCRYPTED MESSAGE:10110001100001100 11101101000111111101001 01111110
DO U WANT TO CONTINUE:(0 - EXIT)1

ENTER MESSAGE TO ENCRYPT:dead


ENCRYPTED MESSAGE:1111110100011111
DO U WANT TO CONTINUE:(0 - EXIT)2

ENTER MESSAGE TO ENCRYPT:FIS


ENCRYPTED MESSAGE:
DO U WANT TO CONTINUE:(0 - EXIT)0
```

LEARNING OUTCOMES:

1. Learnt to organize data using priority queue.

2. A structure node containing data, frequency, left and right pointers to the node is created and other structure minHeap containing the size capacity and array of pointers to node structure is created.

3.Operations carried on the data are creation of min heap involving insertion, deletion and extract minimum, building of Huffman tree and finding Huffman code.

4.The above data and operations together are used to create HuffmanCodeADT.

5. Message encryption app using Huffman code has been developed using HuffmanCodeADT and has been tested.