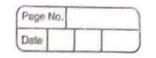
TITE BISSE			
72017912C			
PAWAR	GROUP A: ASSIGNMENT No. 2 Page No. Date		
	Title: Huffman Encoding		
	= Se Harriot Encoding		
	Objective: To implement Huffman Encoding using		
	a greedy stoategy.		
_	Problem Statement:		
	Write a program to implement Huffman		
	Encoding using a greedy strategy.		
	5 5 7 50		
	Software & Harodware Requirement:		
	1. Any Operating System		
	2. Desktop/Laptop		
	3. Compilers/Intempreters		
	4. IDE (any)		
<u></u>			
	Theory:		
	Huffman Coding:		
-	1) Huffman coding is a lossless data compression		
ACCES TO THE PARTY OF THE PARTY	algorithm		
	2) The idea is to design (assign variable - length		
-	codes to input characters, lengths of the assigned		
	coders are based on the frequencies of		
	3> The most frequent character gets the		
and the same	smallest code & the least frequent characters		
The state of the s	gets the largest code.		
	4> The variable-length codes assigned to input		
	characters are Profix Codes, means the codes		
	(bit sequences) are assigned in such a way that		



the code assigned to one characters is not the prefix of code assigned to any other characters.

5) This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

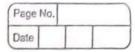
Greedy Algorithm:

- 1) Grocedy is an algorithmic paradigm that builds up a solution piece by piece , always choosing the next piece that offers the most obvious & immediate benifit.
- 2) Problems where choosing locally optimal also leads to global solution are the best fit for greedy.

 3) It greedy algorithm is an approach for

solving a problem by selecting the best option available at the moment.

- 4) It doesn't worry whether the cyrrent best will bring the overall optimal result.
- decision even if the choice is wrong.
- 6) It wooder in top-down approach
- 7) This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.



Algorithm: Step 1: Calculate the forquerry of each character in the string Step 2: Soot the characters in increasing orders of the Frequency. There are stored priority queue 9 Step 3: Make each unique characters as a leaf node Step 4: Crocate an empty node 2. Assign the min. Freq. to the left child of 2 & assign the second min foreq. to the right child at z. Set the value of I as the sum of the above two min. Forg. Step 5: Remove there two min freq. from 9 & add the Sum into the list of Forg. Step 6: Insert/node Z into the toce Step 7: Repeat step 3 to 5 for all the characters Steps: For each non-leaf node, assign 0 to the Very edge & I to the right edge. Nathematical Model: Time Complexity:

O (nlogn) where n is the no. of unique characters. If there are needed,

If the input array is sorted, there exists que linear time algorithm.

Page No.

Test Cases:

				1
No	Input	Expected	Actual	Result
		Output	Octput	_
1	{'a', 'b',	F:0	f : 0	
	'c', 'd',	C 2100	(=100	Pass
	`e',`f'}	d=10)	d=101	
	{s,g,	a = 1100	a = 1100	
	12,13	b=110)	b=1101	
	16,453	e = 1	e=111	
	5'6'.b',	9 2 1 1	Q=11	
2.	'c','d'3	b = 100	b=100	P055
	{ S,1,	C = 0	(=0	
1	6,33	d=10)	d=10)	
		w 4-66 a.c.		

Conclusion: Successfully implemented Hulfman Encoding using a greedy stoategy.

The way

Code:

```
#include <bits/stdc++.h>
using namespace std;
struct MinHeapNode {
    char data;
    unsigned freq;
    MinHeapNode *left, *right;
    MinHeapNode(char data, unsigned freq)
        left = right = NULL;
        this->data = data;
        this->freq = freq;
};
struct compare {
    bool operator()(MinHeapNode* 1, MinHeapNode* r)
    {
        return (1->freq > r->freq);
    }
};
void printCodes(struct MinHeapNode* root, string str)
    if (!root)
        return;
    if (root->data != '$')
        cout << root->data << ": " << str << "\n";</pre>
    printCodes(root->left, str + "0");
    printCodes(root->right, str + "1");
void HuffmanCodes(char data[], int freq[], int size)
    struct MinHeapNode *left, *right, *top;
    priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> minHeap;
    for (int i = 0; i < size; ++i)
        minHeap.push(new MinHeapNode(data[i], freq[i]));
    while (minHeap.size() != 1) {
```

```
left = minHeap.top();
    minHeap.pop();

    right = minHeap.top();
    minHeap.pop();

    top = new MinHeapNode('$', left->freq + right->freq);

    top->left = left;
    top->right = right;

    minHeap.push(top);
}

printCodes(minHeap.top(), "");
}

int main()
{

    char arr[] = { 'a', 'b', 'c', 'd', 'e', 'f' };
    int freq[] = { 5, 9, 12, 13, 16, 45 };

    int size = sizeof(arr) / sizeof(arr[0]);

    HuffmanCodes(arr, freq, size);
    return 0;
}
```

Output:

```
f: 0
c: 100
d: 101
a: 1100
b: 1101
e: 111
```