|  |  |  |
| --- | --- | --- |
| +animatedLOGO | **Data Structures (CS301)**  **Assignment # 03**  **Semester Spring 2021** | **Total Marks: 20**  **Due Date: 12/07/2021** |
| **Solution**  **NAME: TAMKEEN SAJJAD**  **ID: MC200400003**  **Course: MIT** | | |

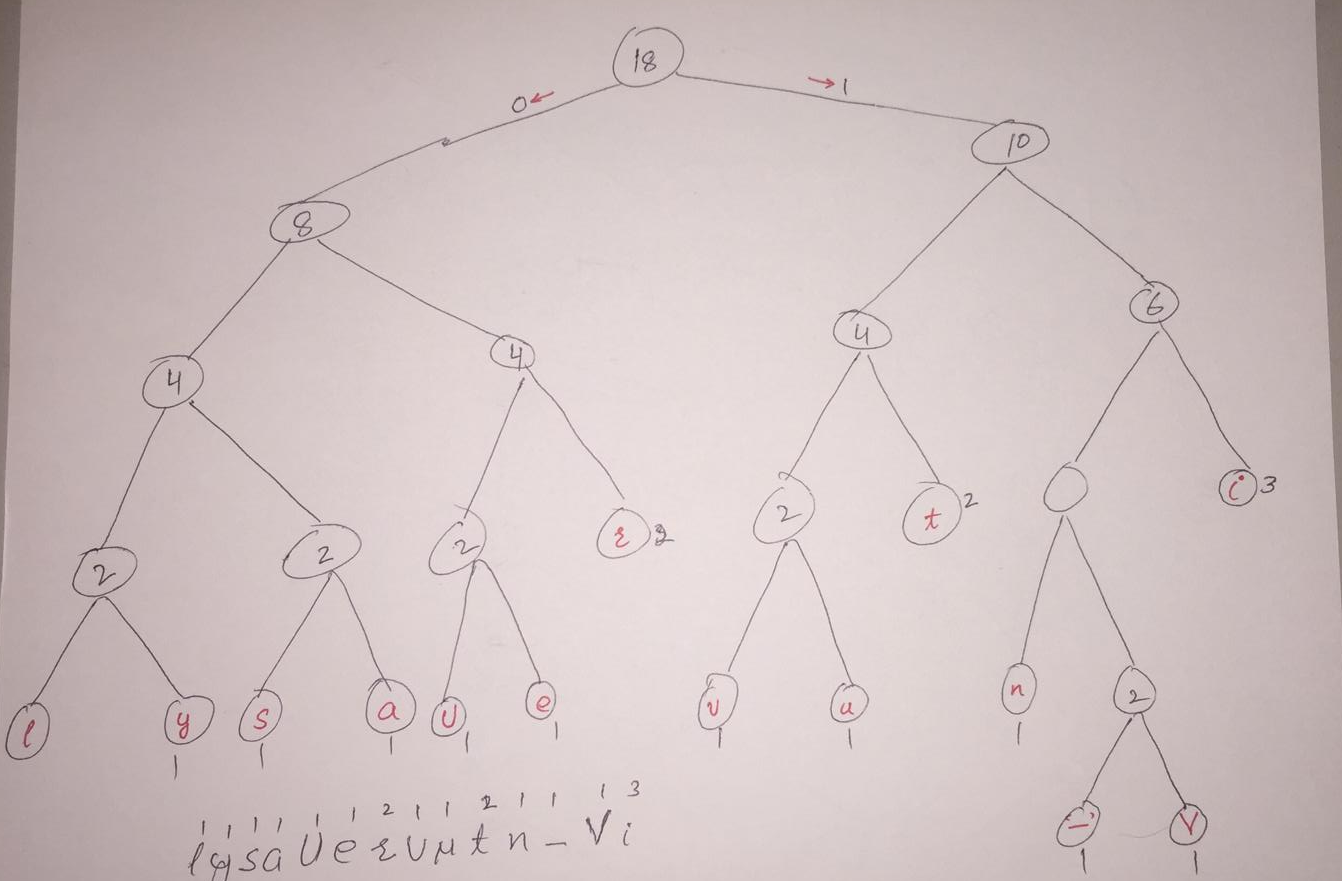
**Tree**

**V i r t u a l U n i v e r i s t y**

1 3 2 2 1 1 1 1 1 1 1 1 2 1 1 => Total Distinct Characters = 14

1 3 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 => Total Characters = 18

The tree is given as:



**Table Calculations:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Character | ASCII | Encoded | Encoded Length l | Frequency F | Probability P | L =l\*P | Entropy E |
| Space | 00100000 | 11010 | 5 | 1 | 0.06 | 0.30 | 0.243534 |
| U | 01010101 | 0100 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
| V | 01010110 | 11011 | 5 | 1 | 0.06 | 0.30 | 0.243534 |
| a | 01100001 | 0011 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
| e | 01100101 | 0101 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
| i | 01101001 | 111 | 3 | 3 | 0.17 | 0.51 | 0.434587 |
| l | 01101100 | 0000 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
| n | 01101110 | 1100 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
| r | 01110010 | 011 | 3 | 2 | 0.11 | 0.33 | 0.350287 |
| s | 01110011 | 0010 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
| t | 01110100 | 101 | 3 | 2 | 0.11 | 0.33 | 0.350287 |
| u | 01110101 | 1001 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
| v | 01110110 | 1000 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
| y | 01111001 | 0001 | 4 | 1 | 0.06 | 0.24 | 0.243534 |
|  |  |  |  |  |  | 3.93 | 3.81403 |
|  |  |  |  |  |  |  |  |

**Encoded Huffman data:**

11011 111 011 101 1001 0011 0000 11010 0100 1100 111 1000 0101 011 0010 111 101 0001

**V i r t u a l - U n i v e r i s t y**

**Storage:**

Total Encoded Bits: 67

Total Original Bits: 144

Total Bits Saved: 77

Therefore, the amount of memory saved is about 53.47% as compared to the original bits.

**Efficiency:**

Efficiency in terms of averaged bits and entropy = E/L \*100 = 3.93/3.8 \*100 = 97.04911

Efficiency is terms of toral original vs encoded bits: 144/67\*100=214.92%

**Code:**

#include <bits/stdc++.h>

#define MAX\_TREE\_HT 256

using namespace std;

map<char, string> codes;

map<char, int> freq;

struct MinHeapNode

{

char data;

int freq;

MinHeapNode \*left, \*right;

MinHeapNode(char data, int freq)

{

left = right = NULL;

this->data = data;

this->freq = freq;

}

};

struct compare

{

bool operator()(MinHeapNode\* l, MinHeapNode\* r)

{

return (l->freq > r->freq);

}

};

void printCodes(struct MinHeapNode\* root, string str)

{

if (!root)

return;

if (root->data != '$')

cout << root->data << ": " << str << "\n";

printCodes(root->left, str + "0");

printCodes(root->right, str + "1");

}

void storeCodes(struct MinHeapNode\* root, string str)

{

if (root==NULL)

return;

if (root->data != '$')

codes[root->data]=str;

storeCodes(root->left, str + "0");

storeCodes(root->right, str + "1");

}

priority\_queue<MinHeapNode\*, vector<MinHeapNode\*>, compare> minHeap;

void HuffmanCodes(int size)

{

struct MinHeapNode \*left, \*right, \*top;

for (map<char, int>::iterator v=freq.begin(); v!=freq.end(); v++)

minHeap.push(new MinHeapNode(v->first, v->second));

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);

}

storeCodes(minHeap.top(), "");

}

void calcFreq(string str, int n)

{

for (int i=0; i<str.size(); i++)

freq[str[i]]++;

}

string decode\_file(struct MinHeapNode\* root, string s)

{

string ans = "";

struct MinHeapNode\* curr = root;

for (int i=0;i<s.size();i++)

{

if (s[i] == '0')

curr = curr->left;

else

curr = curr->right;

if (curr->left==NULL and curr->right==NULL)

{

ans += curr->data;

curr = root;

}

}

return ans+'\0';

}

int main()

{

string str = "Virtual University";

string encodedString, decodedString;

calcFreq(str, str.length());

HuffmanCodes(str.length());

cout << "Character With there Frequencies:\n";

for (auto v=codes.begin(); v!=codes.end(); v++)

cout << v->first <<' ' << v->second << endl;

for (auto i: str)

encodedString+=codes[i];

cout << "\nEncoded Huffman data:\n" << encodedString << endl;

decodedString = decode\_file(minHeap.top(), encodedString);

cout << "\nDecoded Huffman Data:\n" << decodedString << endl;

return 0;

}