Design and Analysis of Algorithms
Experiential Learning

D.Veera Harsha Vardhan Reddy
210C2030193

Problem 1

Tom is a computer security expert working in XYZ Ltd. Recently a group of computers of XYZ Ltd. has been hacked by a malicious virus. All the files residing in those computers are encrypted and therefore are not useful. However, those files are very important, and the organization has no backup copies. Tom is to get the key
(password) to decrypt those files.
After a thorough run through all the computers in the network, Tom has found the malicious program that did all the mischief. He debugged the code and concluded that the key(password) can be formed with the following steps:

i) There are two log files (text files) residing in a local server. Find the most frequently occurring three words that are common in both the log files.

ii) From the above three words, find the string representing the longest common subsequence (pairwise strings to be taken).

iii) The binary string representing the optimal encoding of the longest common subsequence will form the key.

Tom wants to write a program that will generate the key. Put yourself in Tom's shoes and try to solve the problem.

Take two text files as input for the log files. Make sure there are some common words in the files.

-> pseudocode

Having a look at the above problem, it is mainly divided into three parts.

- Firstly, we need to identify the most frequently repeating three common words from the two text files.
- Going into second step we need to input the three words that are obtained in the first step as three strings in the form of pairwise combination of the three words obtained

. • And then we need to find the longest common subsequence of these three strings using the LCS algorithm.

- Finally, we need to find the binary string representing the optimal encoding of the longest common subsequenceobtained in the previous step using Huffman encoding

->Time and space :

Time complexity for Counter is O(n)

Time complexity for lcs is O(mn)

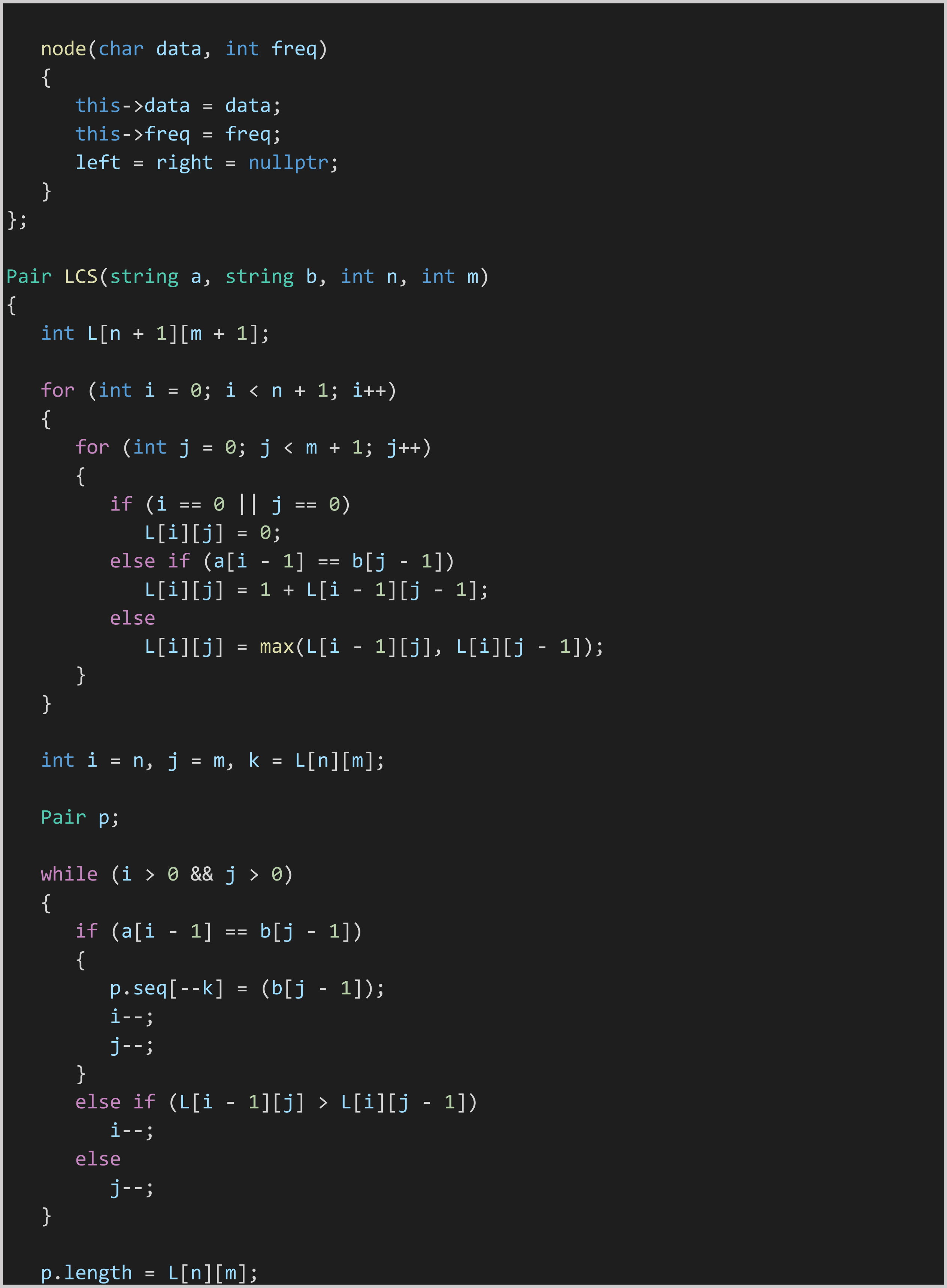
Time complexity for Huffman is O(nlogn)

Overall time complexity is O(mn)

->Code:

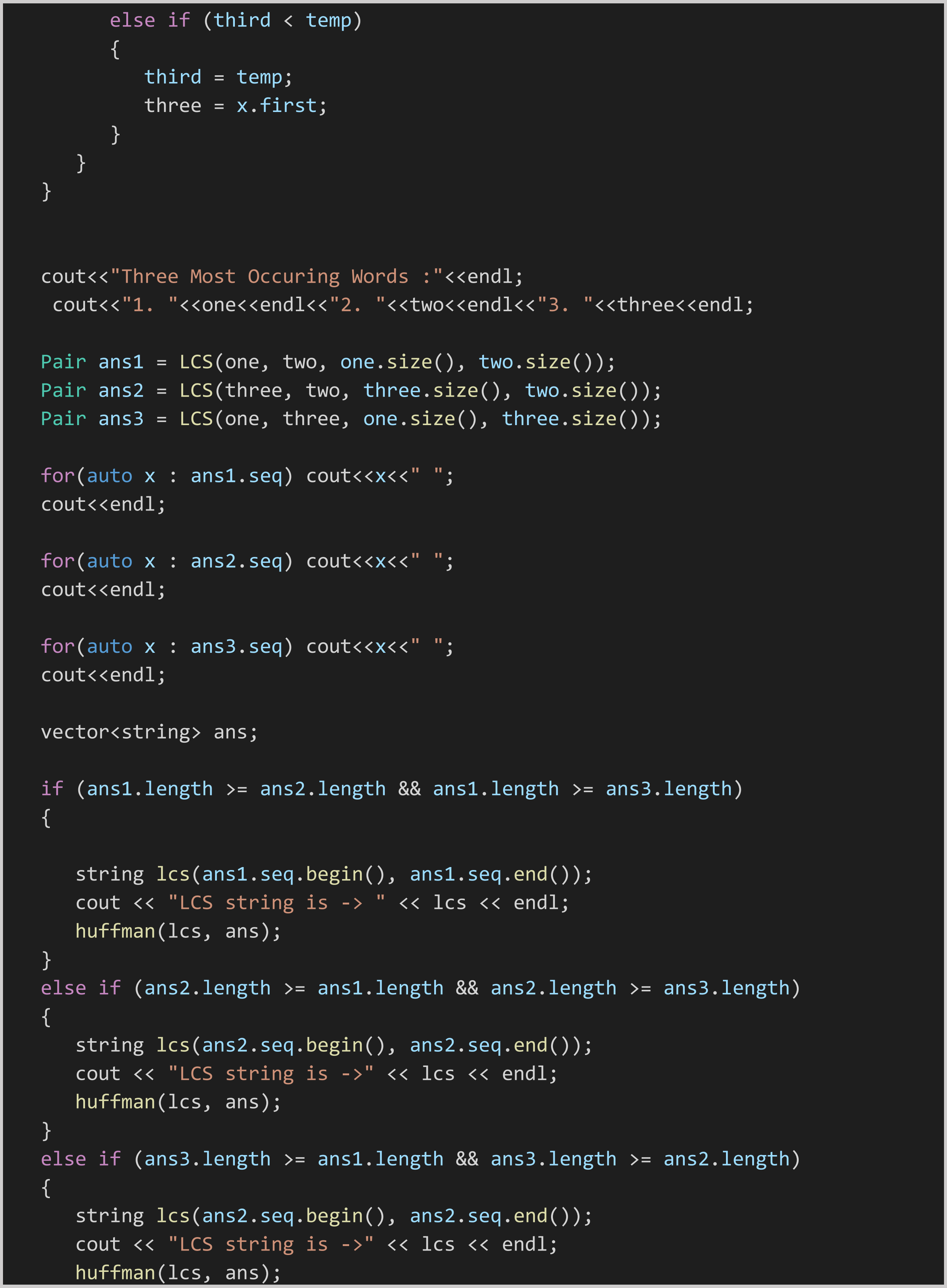
The three parts of the solution are:

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <vector>
#include <queue>
#include<climits>

using namespace std;

struct Pair
{
   int length;
   vector<char> seq;

   Pair()
   {
      seq = vector<char>(100);
   }
};

class node
{
public:
   char data;
   int freq;
   node *left;
   node *right;
```



```
   node(char data, int freq)
   {
      this->data = data;
      this->freq = freq;
      left = right = nullptr;
   }
};

Pair LCS(string a, string b, int n, int m)
{
   int L[n + 1][m + 1];

   for (int i = 0; i < n + 1; i++)
   {
      for (int j = 0; j < m + 1; j++)
      {
         if (i == 0 || j == 0)
            L[i][j] = 0;
         else if (a[i - 1] == b[j - 1])
            L[i][j] = 1 + L[i - 1][j - 1];
         else
            L[i][j] = max(L[i - 1][j], L[i][j - 1]);
      }
   }

   int i = n, j = m, k = L[n][m];

   Pair p;

   while (i > 0 && j > 0)
   {
      if (a[i - 1] == b[j - 1])
      {
         p.seq[--k] = (b[j - 1]);
         i--;
         j--;
      }
      else if (L[i - 1][j] > L[i][j - 1])
         i--;
      else
         j--;
   }

   p.length = L[n][m];
   return p;
}

class comp
{
public:
   bool operator()(node *a, node *b) { return a->freq > b->freq; }
};

void print(node *root, string str, unordered_map<char, string> &map)
{

   if (!root)
      return;

   if (root->data != '@')
      map[root->data] = str;

   print(root->left, str + "0", map);
   print(root->right, str + "1", map);
}

void huffman(string s, vector<string> &ans)
{
   priority_queue<node *, vector<node *>, comp> minheap;

   node *left;
   node *right;
   node *root;

   vector<int> count(26, 0);

   for (char x : s)
      count[x - 'a']++;

   for (int i = 0; i < 26; i++)
   {
      if (count[i] > 0)
      {
         minheap.push(new node(char('a' + i), count[i]));
      }
   }

   while (minheap.size() > 1)
   {
      left = minheap.top();
      minheap.pop();

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

      root = new node('@', (left->freq + right->freq));
      root->left = left;
      root->right = right;

      minheap.push(root);
   }

   unordered_map<char, string> map;

   print(minheap.top(), "", map);

   cout << "code is\n";

   for (auto x : map)
   {
      cout << x.first << ": " << x.second << endl;
   }

   // cout<<count['l' - 'a']<<endl;

   for (char x : s)
   {
      ans.push_back(map[x]);
   }
}

int main()
{
   string line1, line2;
   fstream file1, file2;

   unordered_map<string, int> map1, map2;

   file1.open("log1.txt");
   file2.open("log2.txt");

   if (file1.fail() || file2.fail())
      return 0;
```

```
while (!file1.eof())
{
   file1 >> line1;
   map1[line1]++;
   // cout<<line1<<"\n";
}

while (!file2.eof())
{
   file2 >> line2;
   map2[line2]++;
}

file1.close();
file2.close();

int first = 0, seq, third;
seq = third = INT_MIN;

string one, two, three;

for (auto x : map1)
{
   if (map2.find(x.first) != map2.end())
   {
      int temp = max(x.second, map2[x.first]);
      if (first < temp)
      {
         third = seq;
         seq = first;
         first = temp;

         three = two;
         two = one;
         one = x.first;
      }
      else if (seq < temp)
      {
         third = seq;
         seq = temp;

         three = two;
         two = x.first;
      }
```



```
      else if (third < temp)
      {
         third = temp;
         three = x.first;
      }
   }
}


cout<<"Three Most Occuring Words :"<<endl;
 cout<<"1. "<<one<<endl<<"2. "<<two<<endl<<"3. "<<three<<endl;

Pair ans1 = LCS(one, two, one.size(), two.size());
Pair ans2 = LCS(three, two, three.size(), two.size());
Pair ans3 = LCS(one, three, one.size(), three.size());

for(auto x : ans1.seq) cout<<x<<" ";
cout<<endl;

for(auto x : ans2.seq) cout<<x<<" ";
cout<<endl;

for(auto x : ans3.seq) cout<<x<<" ";
cout<<endl;

vector<string> ans;

if (ans1.length >= ans2.length && ans1.length >= ans3.length)
{

   string lcs(ans1.seq.begin(), ans1.seq.end());
   cout << "LCS string is -> " << lcs << endl;
   huffman(lcs, ans);
}
else if (ans2.length >= ans1.length && ans2.length >= ans3.length)
{
   string lcs(ans2.seq.begin(), ans2.seq.end());
   cout << "LCS string is ->" << lcs << endl;
   huffman(lcs, ans);
}
else if (ans3.length >= ans1.length && ans3.length >= ans2.length)
{
   string lcs(ans2.seq.begin(), ans2.seq.end());
   cout << "LCS string is ->" << lcs << endl;
   huffman(lcs, ans);
   }

   for (string x : ans)
      cout << x;
   cout << endl;

   return 0;
}
```

Text:

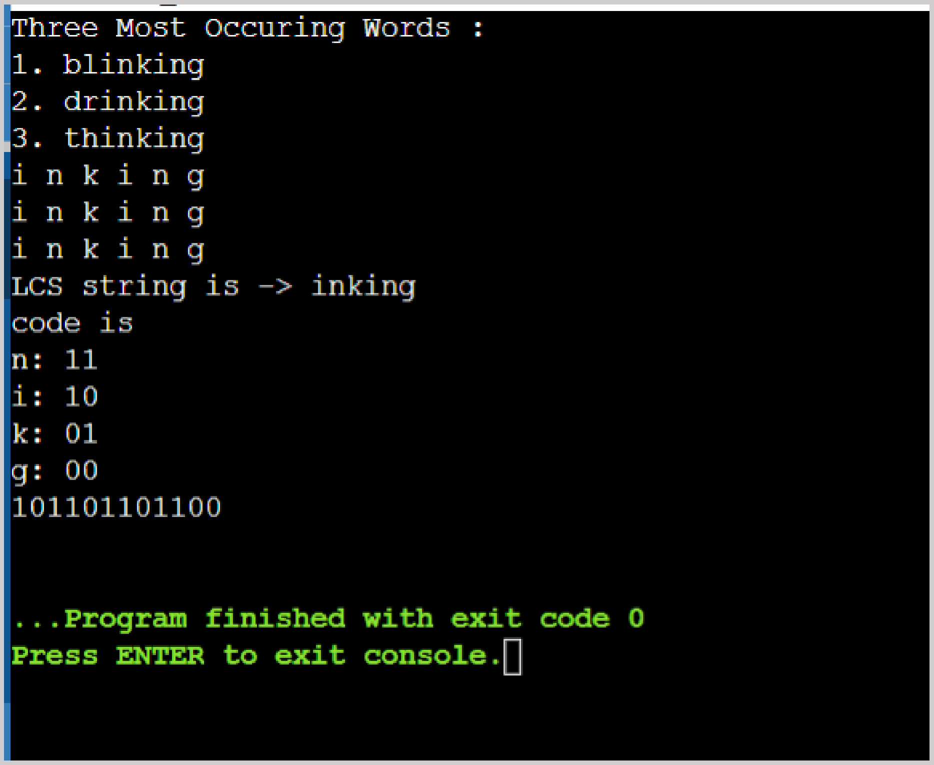
In file1:

Lorem ipsum dolor sit amet consectetuer adipiscing elit Aenean commodo ligula eget dolor Aenean massa Cum sociis natoque penatibus et magnis dis parturient montes nascetur ridiculus mus Donec quam felis ultricies nec pellentesque eu pretium quis sem Nulla consequat massa quis enim Donec pede justo fringilla vel aliquet nec vulputate eget arcu In enim justo rhoncus ut imperdiet a venenatis vitae justo Nullam dictum felis eu pede mollis pretium Integer tincidunt Cras dapibus Vivamus elementum semper nisi Aenean vulputate eleifend tellus Aenean leo ligula porttitor eu consequat vitae eleifend ac enim Aliquam lorem ante dapibus in viverra quis feugiat a tellus Phasellus viverra nulla ut metus varius laoreet Quisque rutrum Aenean imperdiet Etiam ultricies nisi vel augue Curabitur ullamcorper ultricies nisi Nam eget dui Etiam rhoncus Maecenas tempus tellus eget condimentum rhoncus sem quam semper libero sit amet adipiscing sem neque sed ipsum Nam quam nunc blandit vel luctus pulvinar hendrerit id lorem Maecenas nec odio et ante tincidunt tempus Donec vitae sapien ut libero venenatis faucibus Nullam quis ante Etiam sit amet orci eget eros faucibus tincidunt Duis leo Sed fringilla mauris sit amet nibh drinking drinking drinking drinking drinking drinking Donec sodales sagittis drinking drinking drinking drinking drinking drinking magna drinking thinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking thinking thinking thinking thinking thinking drinking drinking drinking drinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking drinking Sed consequat leo eget thinking thinking thinking thinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking blinking thinking thinking drinking drinking drinking drinking drinking drinking thinking thinking thinking thinking thinking thinking bibendum sodales augue velit cursus nunc

In file2:

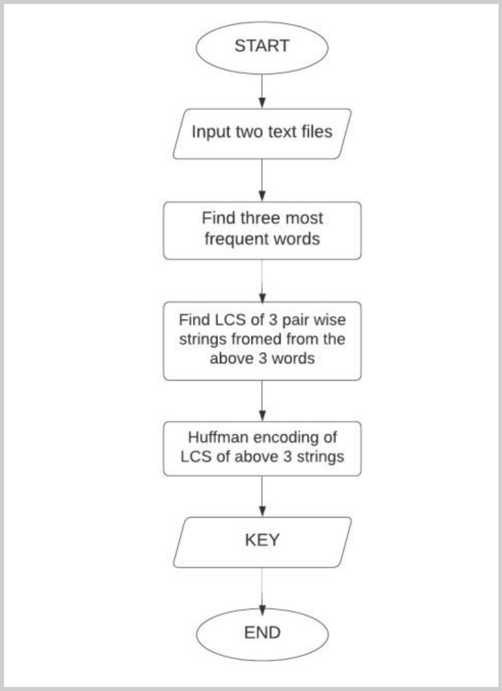
bcdaac bcdaac bcdaac bcdaac Lorem ipsum dolor sit amet consectetuer adipiscing elit Aenean commodo ligula eget dolor Aenean massa Cum sociis natoque penatibus et magnis dis parturient montes nascetur ridiculus mus Donec quam felis ultricies nec pellentesque eu pretium quis sem Nulla consequat massa quis enim Donec pede justo fringilla vel aliquet nec vulputate eget arcu In enim justo rhoncus ut imperdiet a venenatis vitae justo Nullam dictum felis eu pede mollis pretium Integer tincidunt Cras dapibus Vivamus elementum semper nisi Aenean vulputate eleifend tellus Aenean leo ligula porttitor eu consequat vitae eleifend ac enim Aliquam lorem ante dapibus in viverra quis feugiat a tellus Phasellus viverra nulla ut metus varius laoreet Quisque rutrum Aenean imperdiet Etiam ultricies nisi vel augue Curabitur ullamcorper ultricies nisi Nam eget dui Etiam rhoncus Maecenas tempus tellus eget condimentum rhoncus sem quam semper libero sit amet adipiscing sem neque sed ipsum Nam quam nunc blandit vel luctus pulvinar hendrerit id lorem Maecenas nec odio et ante tincidunt tempus Donec vitae sapien ut libero venenatis faucibus Nullam quis ante Etiam sit amet orci eget eros faucibus tincidunt Duis leo Sed fringilla mauris sit amet nibh Donec sodales sagittis magna Sed consequat leo eget bibendum sodales augue velit cursus nunc

->output:



```
Three Most Occuring Words :
1. blinking
2. drinking
3. thinking
i n k i n g
i n k i n g
i n k i n g
LCS string is -> inking
code is
n: 11
i: 10
k: 01
g: 00
101101101100


...Program finished with exit code 0
Press ENTER to exit console.
```

->Implementation Diagram


START
Input two text files
Find three most frequent words
Find LCS of 3 pair wise strings fromed from the above 3 words
Huffman encoding of LCS of above 3 strings
KEY
END

->Algorithm

1. Import Counter class from collections module.
2. Split the string into a list using split (), it will return the lists of words.
3. Now pass the list to the instance of Counter class
4. The function 'most-common ()' inside Counter will return the list of most frequent words from the list and its count.
5. Apply the same for the second list as well.
6. Then add the frequencies of same words in two lists and the top three words with highest frequency sum are returned
7. Then we apply the LCS algorithm on the most common words.
8. Create a priority queue Q consisting of each unique character. sort then in ascending order of their frequencies. for all the unique characters: create a newNode extract minimum value from Q and assign it to leftChild of newNode extract minimum value from Q and assign it to rightChild of newNode calculate the sum of these two minimum values and assign it to the value of newNode insert this newNode into the tree return rootNode .