BLG 372E

ANALYSIS OF ALGORITHMS II

CRN: 22853

REPORT OF HOMEWORK #2

Submission Date: 26.04.2014

STUDENT NAME: TUĞRUL YATAĞAN

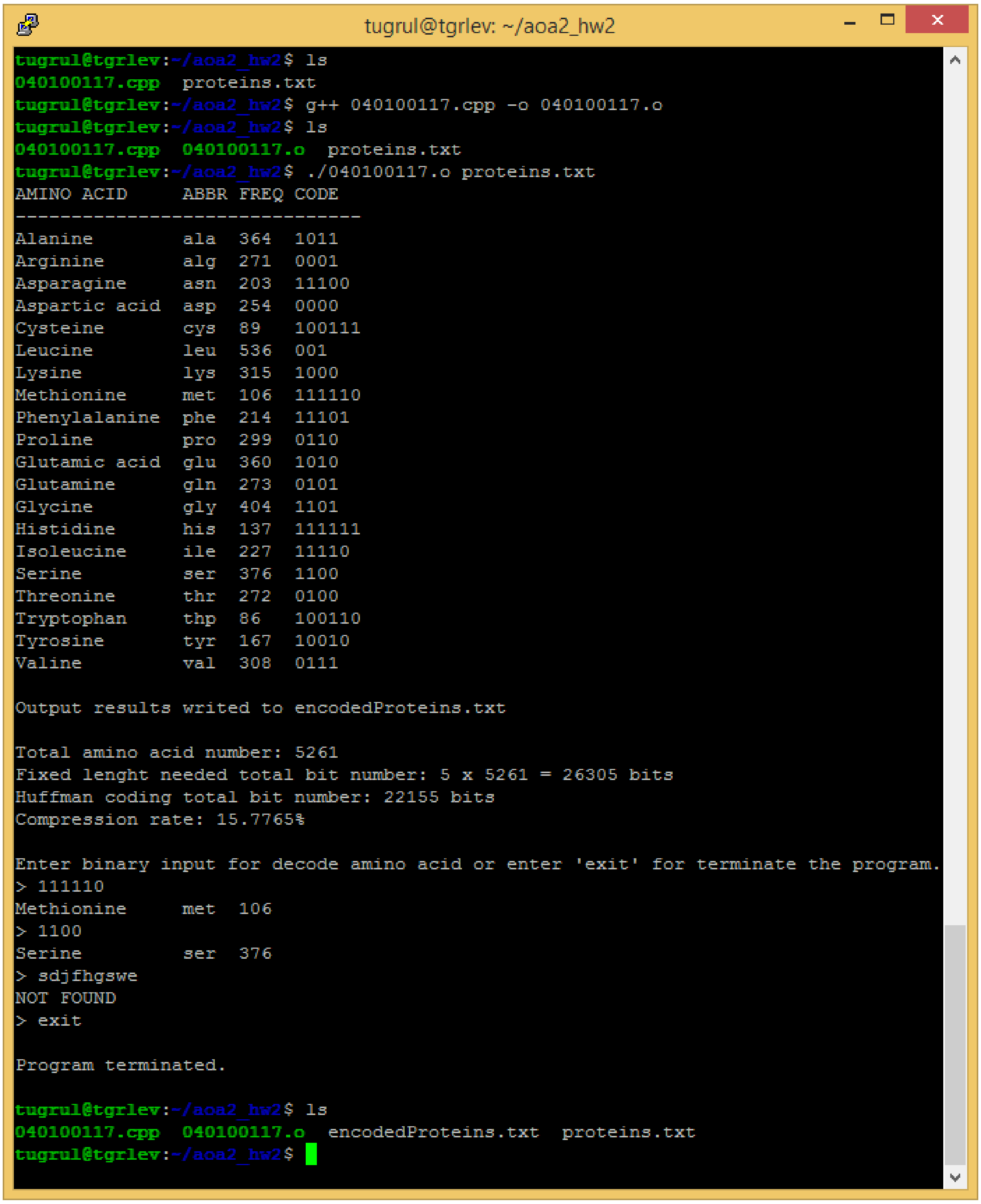
STUDENT NUMBER: 040100117

1. Building and Running

The program built and compiled without any warning or error under g++ and the program executed with commands:

```
g++ 040100117.cpp -o Huffman

./Huffman proteins.txt
```

Sample output is below:



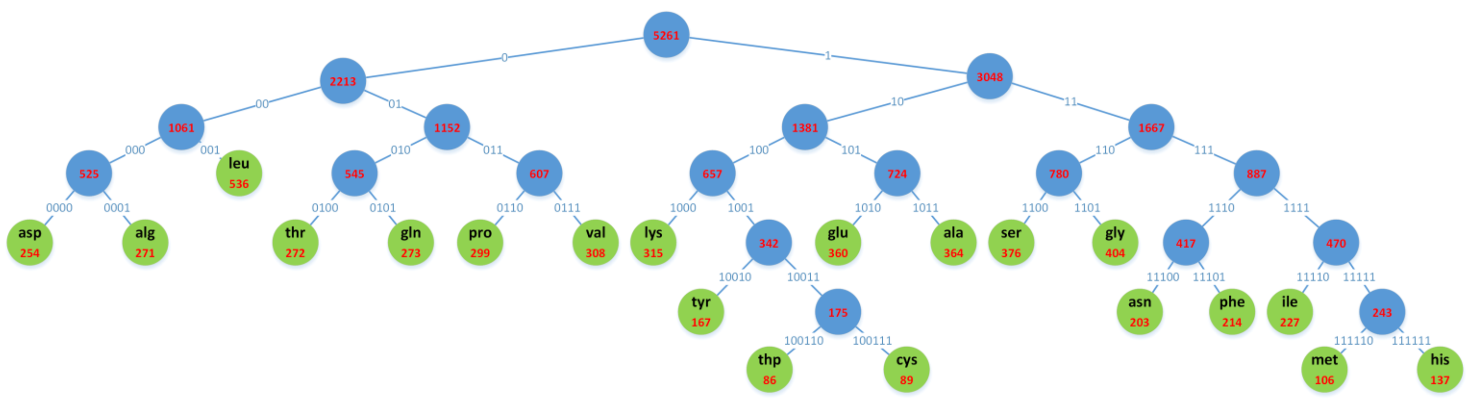
```
tugrul@tgrlev:~/aoa2_hw2$ ls
040100117.cpp  proteins.txt
tugrul@tgrlev:~/aoa2_hw2$ g++ 040100117.cpp -o 040100117.o
tugrul@tgrlev:~/aoa2_hw2$ ls
040100117.cpp  040100117.o  proteins.txt
tugrul@tgrlev:~/aoa2_hw2$ ./040100117.o proteins.txt
AMINO ACID     ABBR FREQ CODE
-------------------------------
Alanine        ala  364  1011
Arginine       alg  271  0001
Asparagine     asn  203  11100
Aspartic acid  asp  254  0000
Cysteine       cys  89   100111
Leucine        leu  536  001
Lysine         lys  315  1000
Methionine     met  106  111110
Phenylalanine  phe  214  11101
Proline        pro  299  0110
Glutamic acid  glu  360  1010
Glutamine      gln  273  0101
Glycine        gly  404  1101
Histidine      his  137  111111
Isoleucine     ile  227  11110
Serine         ser  376  1100
Threonine      thr  272  0100
Tryptophan     thp  86   100110
Tyrosine       tyr  167  10010
Valine         val  308  0111

Output results writed to encodedProteins.txt

Total amino acid number: 5261
Fixed lenght needed total bit number: 5 x 5261 = 26305 bits
Huffman coding total bit number: 22155 bits
Compression rate: 15.7765%

Enter binary input for decode amino acid or enter 'exit' for terminate the program.
> 111110
Methionine     met  106
> 1100
Serine         ser  376
> sdjfhgswe
NOT FOUND
> exit

Program terminated.

tugrul@tgrlev:~/aoa2_hw2$ ls
040100117.cpp  040100117.o  encodedProteins.txt  proteins.txt
tugrul@tgrlev:~/aoa2_hw2$
```

2. Data Structures and Variables

Purpose of the classes and methods explained in the source code as comment lines.

- `AminoAcid aminoAcids[20];`
  array of amino acids which keeps; name, abbreviated name, occurrence number and Huffman code of amino acids.
- `HuffmanTree Tree;`
  Keeps tree data structure of Huffman tree and linked list.
- `HuffmanNode;`
  Node class for keeping right, left child pointer for tree and next pointer for linked list.
- `HuffmanNode *listHead;`
  Head of the linked list to be converted to Huffman tree. Linked list generated from `aminoAcids` array and it uses for tree building operation.
- `HuffmanNode *treeRoot;`
  Root of the Huffman tree.
- `LookupTable Table;`
  Lookup table class which keeps trie data structure for lookup operation.
- `LookupNode;`
  Node of lookup table trie data structure which keeps letter and children information.
- `LookupNode *lookupRoot;`
  Root of the trie data structure of lookup table

Final Huffman tree is below:



3. Analysis

Total amino acid number in the input file is 5261. If we use fixed length codes for these 20 amino acid items, we need to denote each one of them with 5 bits ($2^4<20<2^5$). As a result, we need 5261 x 5 = 26305 bits to represent the input. However, Huffman coding uses 22155 bits. Compression rate is:

$$\frac{(\text{Fixed Length Bits} - \text{Huffman Coding Bits})\text{x}100}{\text{Fixed Length Bits}}$$

$$\frac{(26305 - 22155)\text{x}100}{26305} = 15.7765\%$$

If the input file consists of elements which has much higher frequency over other elements and elements which has smaller frequency, the compression rate will be higher. So if distribution of occurrence number of elements are scattered then the compression rate will be higher, if distribution of occurrence number of elements are close to between them (exp: all elements has same occurrence number) then the compression rate will lower.