CSCI – B505 Applied Algorithms

Lab Assignment – 5

Submitted by – Saiabhinav Chekka

Text File used-

http://www.gutenberg.org/ebooks/69308 - The call from beyond by Clifford D.Simak

Description-

The Huffman Coding was implemented in the following steps-

1. A class 'Heap_Node' is defined to construct heap object, a heap object has following features–
    a. Char – the character at the node
    b. Freq – The frequency of the character
    c. Left – The character on the left branch of the node
    d. Right – The character on the right branch of the node
2. We then create a frequency dictionary for all the characters in the text of the book using the method 'calculate_frequency'
3. Next, we create a heap node object for every character using the method 'make_heap'
4. Further, we combine the characters with the lowest frequencies subsequently using the method 'merge_nodes'
5. Codes are assigned to all the characters based on their frequency of occurrence using the method 'char_codes'
6. We remove the characters which don't have ASCII values between 31 and 128
7. Next, we calculate the number of bits taken by doing Huffman Coding by multiplying the frequency of a character with the length of codes for that character
8. We compare the number of bits calculated above with the number of bits it would've taken to encode every character in a seven-bit character.
9. The results for the comparison for the given book are as follows-

    Size of encoding using Huffman coding -> 310157 bits

    Number of characters -> 82465

    Size of encoding using 7-bit fixed length encoding -> 82465 * 7 = 577255 bits

    Number of bits saved -> 577255 – 310157 = 267098 bits

Output-

Character codes :

```
t 000
s 0010
r 0011
h 0100
' 01010000
R 010100010
1 0101000110
x 0101000111
A 01010010
```

I 01010011
v 0101010
j 010101100
B 010101101
G 010101110
U 0101011110
? 0101011111
u 01011
e 011
i 1000
n 1001
y 101000
W 10100100
C 101001010
H 101001011
T 10100110
M 1010011100
) 101001110100
(101001110101
4 1010011101100
6 1010011101101
: 101001110111
z 10100111100
9 1010011110100
V 1010011110101
0 101001111011
q 10100111110
X 1010011111110000
 1010011111100010
1010011111100011
7 10100111111001
8 1010011111101
2 1010011111110
5 1010011111111
p 101010
f 101011
a 1011
l 11000
. 110010
" 1100110
N 110011100
P 110011101
E 110011110
- 110011111
o 1101
w 111000
g 111001
d 11101
k 1111000
Y 1111001000
D 1111001001
S 111100101
L 1111001100
O 1111001101
F 1111001110
! 1111001111000
3 1111001111001
_ 111100111101
J 111100111110

```
] 111100111111000
[ 111100111111001
% 1111001111110100
$ 1111001111110101
; 1111001111110110
Q 1111001111110111
/ 11110011111110
K 11110011111111
m 111101
c 111110
b 1111110
 , 1111111
```

Number of characters which are encoded: 80

Number of characters which are encoded between ASCII values 31 and 128: 79

Character Frequency: `{'t': 3, 's': 4, 'r': 4, 'h': 4, "'": 8, 'R': 9, '1': 10, 'x': 10, 'A': 8, 'I': 8, 'v': 7, 'j': 9, 'B': 9, 'G': 9, 'U': 10, '?': 10, 'u': 5, 'e': 3, 'i': 4, 'n': 4, 'y': 6, 'W': 8, 'C': 9, 'H': 9, 'T': 8, 'M': 10, ')': 12, '(': 12, '4': 13, '6': 13, ':': 12, 'z': 11, '9': 13, 'V': 13, '0': 12, 'q': 11, 'X': 15, '#': 16, '7': 14, '8': 13, '2': 13, '5': 13, 'p': 6, 'f': 6, 'a': 4, 'l': 5, '.': 6, '"': 7, 'N': 9, 'P': 9, 'E': 9, '-': 9, 'o': 4, 'w': 6, 'g': 6, 'd': 5, 'k': 7, 'Y': 10, 'D': 10, 'S': 9, 'L': 10, 'O': 10, 'F': 10, '!': 13, '3': 13, '_': 12, 'J': 12, ']': 15, '[': 15, '%': 16, '$': 16, ';': 16, 'Q': 16, '/': 14, 'K': 14, 'm': 6, 'c': 6, 'b': 7, ',': 7}`

The text was encoded using 310157 bits

The text had 82465 valid characters

Using a 7-bit fixed length encoding, this would have been 577255 bits long

So we saved 267098 bits!

1. How many bits were you able to save by using Huffman encoding, compared to an n-bit fixed-length code?

Ans: We were able to save 267098 bits when compared to 7-bit fixed-length encoding.

2. How many characters did you choose to go with 32, 64, or 128?

Ans: I chose to go with 128 characters.