Department of Electrical and Computer Engineering
University of Waterloo
Please send any comments or criticisms to dwharder@alumni.uwaterloo.ca
with the subject ECE 250 Questions 2.4.
Assistances and comments will be acknowledged.

5.1*a* What differentiates a binary tree from a general tree where each node has at most two sub-trees?

5.1*b* One benefit of having every member function of a binary tree check if the current node is empty (that is, `this == nullptr`) is that a function can be written as:

```
template <typename Type>
int Binary_node<Type>::height() const {
    return empty() ? -1 :
                        1 + std::max( left()->height(), right()->height() );
}
```

as opposed to explicitly having to check:

```
template <typename Type>
int Binary_node<Type>::height() const {
    return 1 + (( left() == nullptr) ? 0 :  left()->height())
             + ((right() == nullptr) ? 0 : right()->height());
}
```

What are the negative effects of always having each member function check whether it is being called on an empty node or not?

5.1*c* What are the least and greatest number of leaf nodes in a binary tree with *n* nodes?

5.1*d* Is there any restriction as to the number of nodes in a full binary tree (where each node has either zero or two children)?

5.1*e* What is the relationship between the number of nodes in a full binary tree and the number of leaf nodes?

5.1*f* What is the maximum depth of a full binary tree?

5.1*g* Write a member function that returns the number of leaf nodes that are descendant from the node the member function is called on.

```
template <typename Type>
int Binary_node<Type>::leaf_count() const {
```

5.1*h* A Huffman encoding of a document is a means of compression by allocating fewer bits to encode letters that appear often and, thus, requiring more bits for letters that occur only seldom. A Huffman tree is a full binary tree where each internal node is a decision point and each leaf node is a letter. In order to decode a string of bits, begin at the root:

1. If the node is a letter, output that letter, otherwise
2. If the next bit is a 0, move to the left sub-tree and if the bit is a 1, move to the right sub-tree and go back to Step 1.

Please send any comments or criticisms to dwharder@alumni.uwaterloo.ca
with the subject ECE 250 Questions 2.4.
Assistances and comments will be acknowledged.

The following seen in Figure 1 is taken from the chapter *Huffman Coding* in the text *CS 573 Algorithms* by Sariel Har-Peled and is a Huffman tree for the frequency of letters in Charles Dickens book “A Tale of Two Cities”.

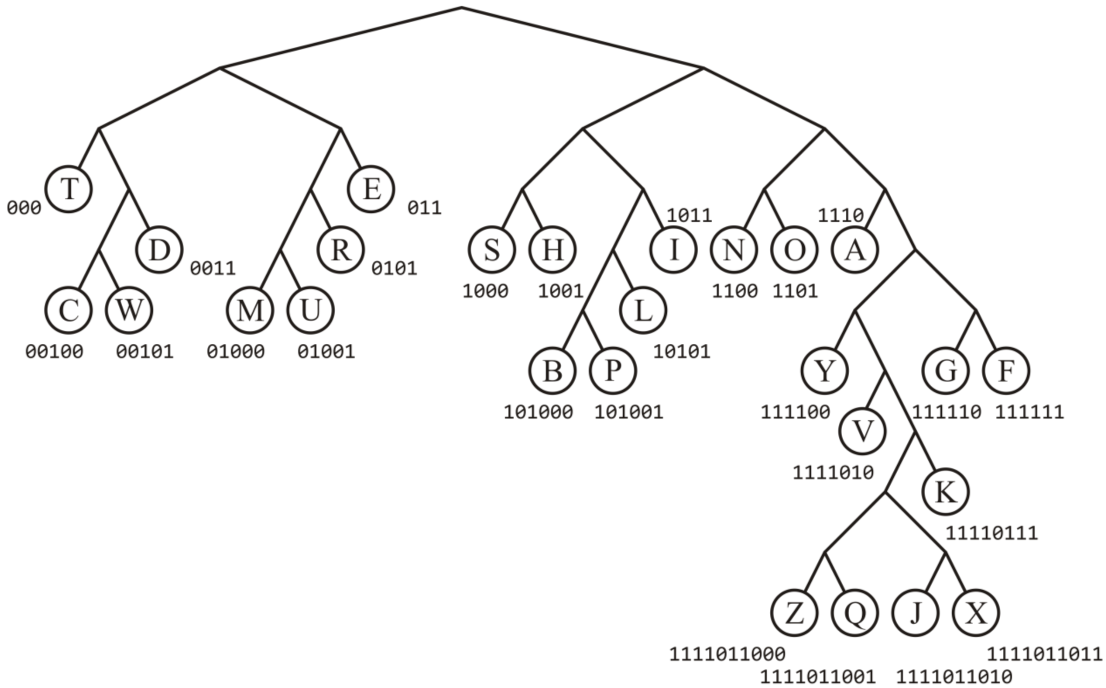

Figure 1. A Huffman tree for the encoding of “A Tale of Two Cities”—letters only.

Thus, “It was the best of times” would be coded as

I	T	W	A	S	T	H	E	B	E	S	T	O	F	T	I	M	E	S
1011	000	00101	1110	1000	000	1001	011	101000	011	1000	000	1101	111111	000	1011	01000	011	1000

These would be strung together as:

1011000**00101**1110**1000**000**1001**011**101000**011**1000**000**1101**111111**000**1011**01000**011**1000**

Note that this uses 75 bits to encode these 19 letters. What is the average number of bits per character, and what is the savings if we were to use 8-bit ASCII encoding for each character?

To decode this, start at the root. The first four bits indicates we should go right-left-right-right, as shown in Figure 2.

Department of Electrical and Computer Engineering
University of Waterloo
Please send any comments or criticisms to dwharder@alumni.uwaterloo.ca
with the subject ECE 250 Questions 2.4.
Assistances and comments will be acknowledged.

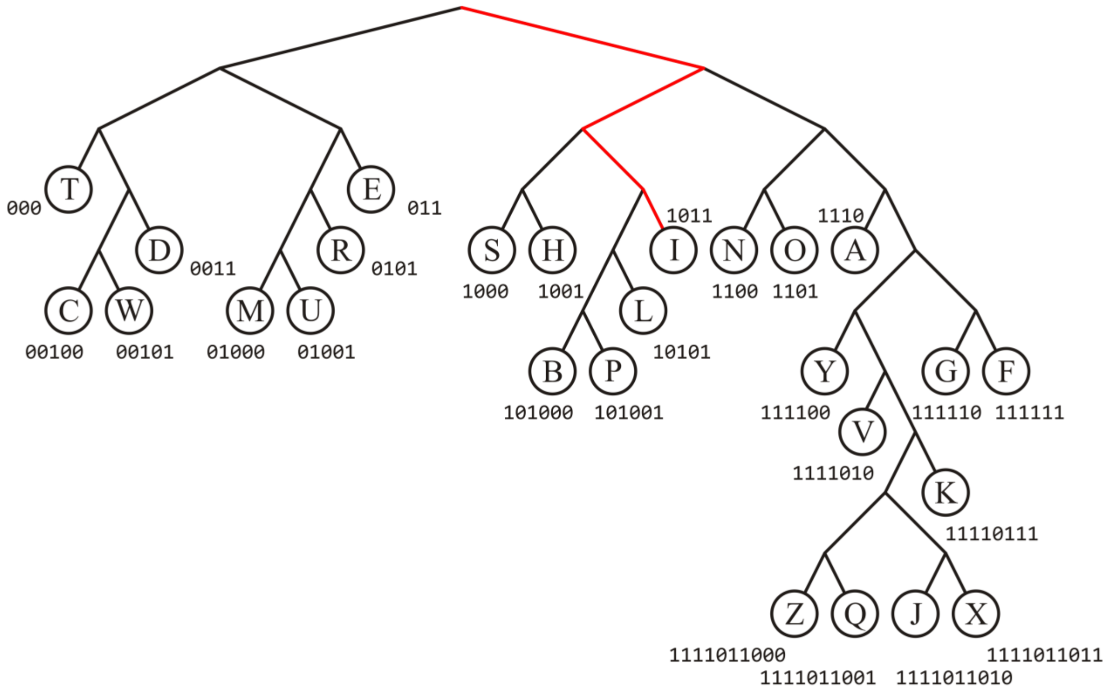

Figure 2. Decoding 1011.

Arriving at "I", we write that letter down, and continue. The next three bits, `000`, take us left-left-left from the root to the letter "T". Use this technique (and perhaps a good guess) to decode the text:

101100010111000111011111111100101111111111001011010000110000000110101000100110111 1001
1111000001001111000010110011110 1

What is the best-case scenario if one bit is changed in a Huffman encoding of a document? What is the worst case?

5.1*i* Without using a calculator, is an approximation of lg(n) for n = 4000, n = 256 000 and n = 8 000 000?