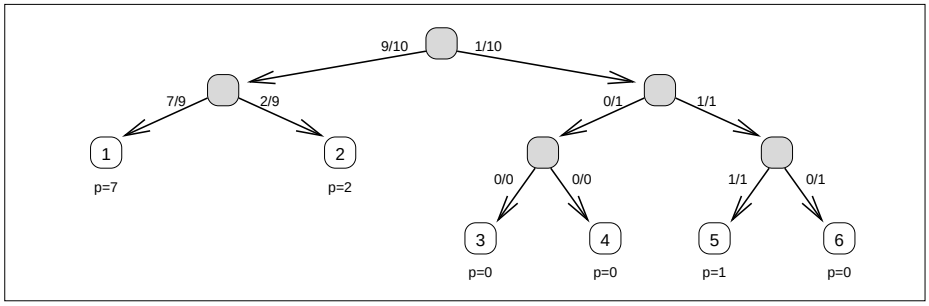

Figure 5.4: Example of binary arithmetic coding used to deal with a multi-symbol alphabet. In this example the source alphabet is $S = [1 \ldots 6]$, with symbol frequencies $P = [7, 2, 0, 0, 1, 0]$, and the tree is based upon the structure of a minimal binary code.

number s, again with each bit biased by exactly the right amount. This tree has the advantage of requiring almost the same number of binary arithmetic coding steps to transmit each symbol, and minimizes the worst case number of steps needed to code one symbol.

Figure 5.4 shows the tree that results if the alphabet $S = [1, 2, 3, 4, 5, 6]$ with frequencies $P = [7, 2, 0, 0, 1, 0]$ is handled via a minimal binary tree. To code symbol $s = 2$, for example, the left branch out of the root node is taken, and a code of $-\log_2(9/10)$ bits generated, then the right branch is taken to the leaf node 2, and a code of $-\log_2(2/9)$ bits generated, for a total codelength (assuming no compression loss) of $-\log_2(2/10)$, as required. Note that probabilities of $0/1$ and even $0/0$ are generated but are not problematic, as they correspond to symbols that do not appear in this particular message. Probabilities of $1/1$ correspond to the emission of no bits.

The second possible measure of efficiency is to minimize the average number of calls to function *binary_arithmetic_encode*$()$. It should come as no surprise to the reader (hopefully!) that the correct tree structure is a Huffman tree, as this minimizes the weighted path length over all binary trees for the given set of probabilities. The natural consequence of this is that, as far as is possible, the conditional binary probabilities used at each step will be approximately 0.5, as in a Huffman tree each node represents a single bit, and that single bit carries approximately one bit of information.

The third possible measure of efficiency is the hardest to minimize, and that is compression effectiveness. In any practical arithmetic coder each binary coding step introduces some small amount of compression loss, and these must be aggregated to get an overall compression loss for the source symbol. For