1. Huffman encoding

We consider a text of length \( m \) on an alphabet \( C \), with probabilities for \( p(c_i) \) for each character \( c_i \in C \), so that the number of times \( c_i \) appears in the text is \( p(c_i)m \). Let’s assume the easy special case in which the probabilities are all powers of two, for instance \( p(c_i) = 2^{-2} = 1/4 \), or \( p(c_i) = 2^{-3} = 1/8 \), and so on. Prove that in this case the average number of bits per character in the Huffman encoding is \(- \sum_{c_i} p(c_i) \log p(c_i)\), achieving the Shannon entropy lower bound.

Hint: Recall that at each iteration the Huffman encoding algorithm joins the two items of minimum frequency to form a new pseudo-character. I would begin by proving that the the two items each have the same frequency \( 2^j \). Then prove that the number of pseudo-characters on the path in the final tree to a character with frequency \( 2^j \) is \( j \).

2. Interrelatedness of NP-complete problems

Let’s assume for the purpose of “almost-contradiction” that you discover an polynomial-time algorithm for Vertex Cover. Looking at Figure 34.13, we see that Clique is reducible to Vertex Cover; so your algorithm gives us a polynomial-time algorithm for Clique as well. We know, in fact, that finding a polynomial-time algorithm for one NP-complete problem gives a polynomial-time algorithm for all NP-complete problems. Describe, using the existence of the reductions pictured in Figure 34.13 and whatever other relevant information you need about NP-completeness, how you algorithm would give a polynomial-time algorithm for Subset Sum. You do not have to describe any of the reductions, just use the fact that they exist.

3. NP-completeness reduction

Do problem 34.5-2, on the 0-1 integer programming problem.