1. (a) Using Huffman encoding scheme on a set $S$ of $n$ symbols with frequencies $f_1, f_2, \ldots, f_n$, what is the longest a codeword could possibly be? Give an example set of frequencies that would produce this case.

(b) Prove that if some character occurs with a frequency more than $2/5$, then there is guaranteed to be a codeword of length 1.

(c) Prove that if all characters occur with frequency less than $1/3$, then there is guaranteed to be no codeword of length 1.

2. Consider the following problem: there are $n$ cities and there are direct roads connecting some pairs of cities. All roads are one-way, but a pair of cities, say $A$ and $B$ may have two one-way roads between them, one going from $A$ to $B$ and the other going from $B$ to $A$. Associated with each road is a positive number that indicates the time it takes to go from one city to another (if there are two roads connecting a pair of cities, the time associated with each road may be different). This network of one-way roads is such that every city is reachable from every other city. There is a special city $S$. Give an efficient algorithm to find the routes that take the smallest amount of time to reach from every city to every other city with the constraint that all routes must pass through $S$.

3. Consider a graph $G = (V, E)$ that is connected and has at most $n + 8$ edges, where $n = |V|$. Give an algorithm with running time $O(n)$ that takes a near-tree $G$ with costs on its edges, and returns a minimum spanning tree of $G$. You may assume that all the edge costs are distinct.

4. Chapter 6, Exercise 3 (page 314).

5. Chapter 6, Exercise 5 (page 316) from the text.

6. Chapter 6, Exercise 9 (page 319) from the text.
7. Consider the following problem that arises in scheduling wireless broadcasts. There is an item of interest, say $I$. There are $n$ requests for $I$ arriving at different times (non-negative integers). Request $i$ has weight $w_i$ and arrives at time $t_i$. A server broadcasts $I$ to the clients at different times. When $I$ is broadcast all unsatisfied requests that arrive before the broadcast get satisfied. For any request $i$ if $t > t_i$ is the first time after $t_i$ at which the $I$ is broadcast, then the cost of satisfying request $i$ is $w_i(t - t_i)$. Due to costs involved in broadcasting, the server can broadcast item $I$ at most $k$ times ($k$ is part of the input). Give an efficient algorithm that determines the times at which the $k$ broadcasts must be scheduled so that the total cost of satisfying all client requests is minimized.

<table>
<thead>
<tr>
<th>I</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$w_0$=2</td>
<td>$w_1$=4</td>
</tr>
</tbody>
</table>

An example scenario is shown in the above figure. There are 4 requests: $w_1 = 2$, $t_1 = 0$, $w_2 = 4$, $t_2 = 1$, $w_3 = 10$, $t_3 = 2$, and $w_4 = 6$, $t_4 = 4$. The server broadcasts item $I$ twice, once at time 2 and once at time 5. The total cost of satisfying all requests is $4 + 4 + 30 + 6 = 44$.

8. A library has $n$ books that must be shelved in alphabetical order on adjustable height book-shelves. Book $i$ is characterized by an ordered pair $(h_i, t_i)$, where $h_i$ and $t_i$ are the height and the thickness of book $i$ respectively. The width of each shelf is $W$ and the sum of thickness of books on each shelf must be at most $W$. The height difference between two consecutive shelves is at least the maximum height of any book on the lower of the two shelves. Give an algorithm that minimizes the total height of shelves needed to store all the books. You may assume that the books are available in alphabetical order.