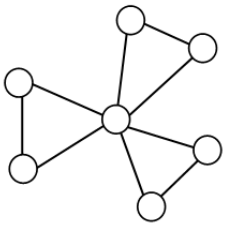
Question set for Chapter 4

1. Suppose a graph $G = (V, E)$, with $|V| \geq 3$, has the following properties:

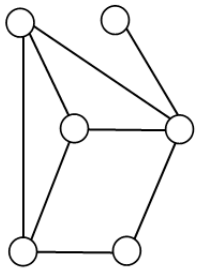
   (a) For any two vertices x and y there is exactly one other vertex z which is adjacent to both x and y;

   (b) There is some vertex v which is adjacent to all others.

   Prove that G must consist of a collection of triangles joined at a common 'hub' vertex, as in the example on the right.
   (The famous **Friendship Theorem** of Paul Erdős, Alfréd Rényi and Vera Sós says that, in fact, property (a) above implies property (b)! See more here: Theorem of the Day)

2. A non-increasing sequence of non-negative integers is a **degree sequence** if some graph G has these integers as its vertex degrees. For example, the sequence $(4, 3, 3, 3, 2, 1)$ is a degree sequence because these are precisely the vertex degrees of the graph shown on the right.

   A 1960 theorem of Paul Erdős and Tibor Gallai says that a non-increasing sequence of non-negative integers $(d_1, d_2, \ldots, d_n)$ is a degree sequence if and only if $\sum d_i$ is even and, for all k, $1 \leq k \leq n$,

   $$\sum_{i=1}^{k} d_i - \sum_{i=k+1}^{n} \min(d_i, k) \leq k(k-1). \qquad (1)$$

   (a) Why do we require that $\sum d_i$ is even?

   (b) Confirm that the degree sequence $(4, 3, 3, 3, 2, 1)$ satisfies inequality (1).

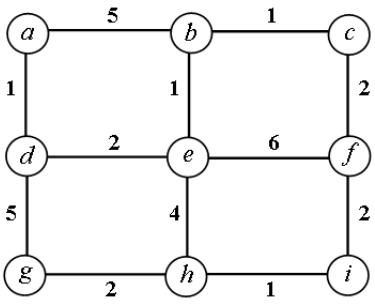
   (c) What is the complexity of testing this inequality for a sequence of length n?

3. Dijkstra's algorithm fails to work properly if negative edge-weights are allowed. Consider the following strategy which attempts to overcome this problem:

   > Let ω be the negative weight of greatest absolute value. Add $|\omega| + 1$ to every weight to create a graph with only positive edge weights. Run Dijkstra's algorithm on the result. Now subtract $|\omega| + 1$ from every edge of the resulting tree.

   Does this strategy work?

4. (a) Run Dijkstra's algorithm on the following graph G using the root vertex $v = a$. Show, diagrammatically or by means of a table, how the algorithm operates at each iteration. Give the final tree and table of distances which are the outputs of the algorithm.

   (b) Draw a minimum-weight spanning tree for the graph G in part (a). You need not show how you obtained this tree.

   (c) Is your tree in part (a) a minimum-weight spanning tree for G? Does your tree in part (b) give minimum-weight paths from a to all other vertices? Give counterexamples where appropriate.