1. 2.3.3 There are exactly five cities in a network. The cost of building a road directly between i and j is the entry $a_{i,j}$ in the matrix below. An infinite entry indicates that there is a mountain in the way and the road cannot be built. Determine the least cost of making all the cities reachable from each other.

$$\begin{pmatrix} 0 & 3 & 5 & 11 & 9 \\ 3 & 0 & 3 & 9 & 8 \\ 5 & 3 & 0 & \infty & 10 \\ 11 & 9 & \infty & 0 & 7 \\ 9 & 8 & 10 & 7 & 0 \end{pmatrix}$$

Solution:
First, label the vertices as shown below:

$$\begin{array}{c} \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{pmatrix} 0 & 3 & 5 & 11 & 9 \\ 3 & 0 & 3 & 9 & 8 \\ 5 & 3 & 0 & \infty & 10 \\ 11 & 9 & \infty & 0 & 7 \\ 9 & 8 & 10 & 7 & 0 \end{pmatrix} \end{array}$$

Using Kruskal's algorithm, we produce the following iterations of edges:
1)ab
2)bc
3)ed
4)be.
Total weight: 21. Hence, the least cost of making all the cities reachable from each other is 21. □

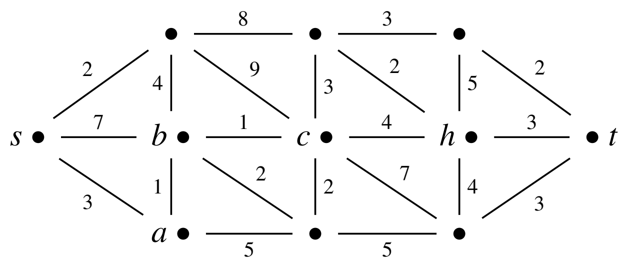
2. 2.3.12 In a weighted complete graph, iteratively select the edge of least weight such that the edges selected so far form a disjoint union of paths. After $n-1$ steps, the result is a spanning path. Prove that this algorithm always gives a minimum-weight spanning path, or give an infinite family of counterexamples where it fails.

Solution:
The algorithm does not always work. Consider a weighted graph constructed from C_n such that the vertices, are labeled in order around the cycle as $v =, 1, 2, ..., n$. Now add an edge between vertex $n-1$ and vertex 2. Now, suppose all the edges are weight 1 except the edge from vertex $n-1$ to vertex 2, call it a, is 2 and the edge from vertex n to vertex 1, call it b, has weight 7. Then the algorithm will output a spanning path using the edge b but it would be cheaper to go through a instead.

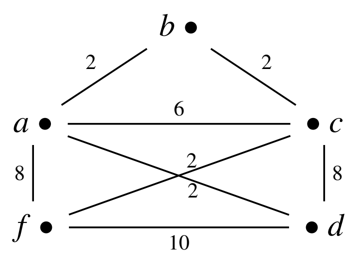
3. A,B,C from worksheet.

Solution, part a:

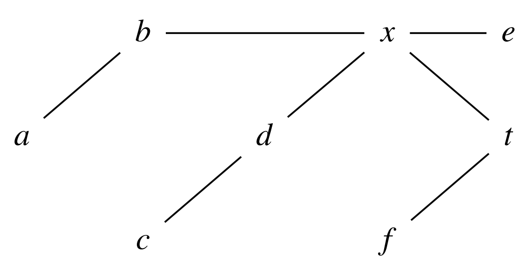
The shortest st-path is $sabcht$. It has length 12.



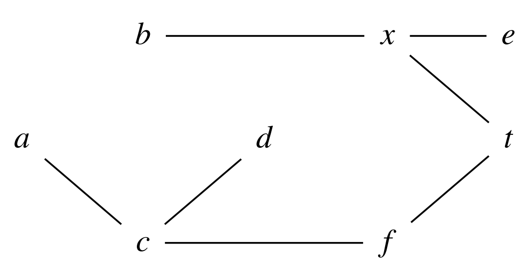
Solution, part b:
Consider the graph below. Since d and f are the only odd vertices, they are the only ones that require repeating an edge. Therefore, solving the Chinese Postman problem requires finding the minimum weight df path. By inspection, this is $dabcf$, which has weight 8. Hence, the walk $dabcfacdfcbad$, of weight 48, solves the Chinese Postman problem on the below graph. This solution is unique because the minimum weight df-path is unique.



Solution, part c:
Breadth-first tree obtained by starting at vertex x shown below.



Breadth-first tree obtained by starting at vertex f shown below.



4. 2.3.14 Let C be a cycle in a connected weighted graph. Let e be an edge of maximum weight on C. Prove that there is a minimum spanning tree not containing e. Use this to prove that iteratively deleting the heaviest non-cut-edge until the remaining graph is acyclic produces a minimum-weight spanning tree.

Solution:
Let G be a connected weighted graph with a cycle C where e is an edge of maximum weight in C. Let T be a minimum weight spanning tree in G. If $e \notin E(T)$, then the result follows. Otherwise, if $e \in E(T)$, there exists some other edge in C that is not in T. This edge, call it f, must have weight less than or equal to the weight of e by assumption. Thus, $w(T - e + f) \leq w(T)$. Hence, $T - e + f$ is a minimum weight spanning tree not using edge e.

Suppose we iteratively delete the maximum weight edges lying on a cycle in G. The algorithm continues until there are no more cycles left in G. Hence, the output is acyclic and connected by definition. By the previous argument, each iteration does not change the weight of a minimum weight spanning tree on G. Hence, the algorithm produces a minimum spanning tree. □

5. 2.3.22.Solve the Chinese Postman Problem in the k-dimensional cube Q_k under the condition that every edge has weight 1.

Solution:
By definition, Q_k is k-regular and connected. If k is even, then Q_k is Eulerian. In that case, the Postman problem is solved by an Euler circuit with a weight of $k * 2^{k-1}$. Otherwise, if k is odd, we Eulerize Q_k by adding a degree to every vertex. We do this by duplicating 2^{k-1} edges in disjoint copies of Q_{k-1} in Q_k. The resulting tour has weight $k2^{k-1} + 2^{k-1} = (k+1)2^{k-1}$

6. 3.1.2. Determine the minimum size of a maximal matching in the cycle C_n.

Solution:
If n is even, then we can always add another edge to our matching up until we have a matching of $n/2$ in C_n. If, however, n is odd, then we can still add an edge up until the floor of $n/2$.

7. 3.1.8. Prove or disprove: every tree has at most one perfect matching.

Solution:
Let M_1 and M_2 be two perfect matchings on a tree T....