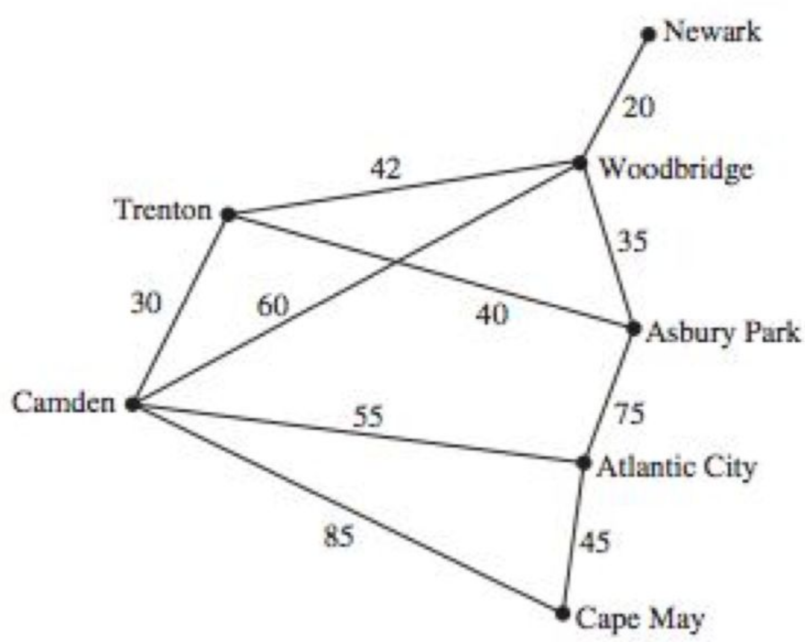
#1
Part a asks us to find the shortest distance between Newark and Camden and between Newark and Cape May, using the below roads.



We can use Dijkstra's Algorithm to do this.

Dijkstra's Algorithm is as follows

1. Mark the ending vertex with a distance of zero.. Designate this vertex as current..

2. Find all vertices leading to the current vertex.. Calculate their distances to the end. Since we already know the distance the current vertex is from the end, this will just require adding the most recent edge. Don't record this distance if it is longer than a previously recorded distance.

3. Mark the current vertex as visited. We will never look at this vertex again.

4. Mark the vertex with the smallest distance as current, and repeat from step 2.

Now, we will use it solve our problems.
The **bold** city will indicate being set at current. An <u>underlined</u> city will indicate it is visited.

STEP 1:

Camden	0
Trenton	30
Woodbridge	60
Atlantic City	55
Cape May	85
Asbury Park	
Newark	

STEP 2:

<u>Camden</u>	0
Trenton	30
Woodbridge	60
Atlantic City	55
Cape May	85
Asbury Park	70
Newark	

STEP 3:

Camden	0
Trenton	30
Woodbridge	60
Atlantic City	55
Cape May	85
Asbury Park	70
Newark	

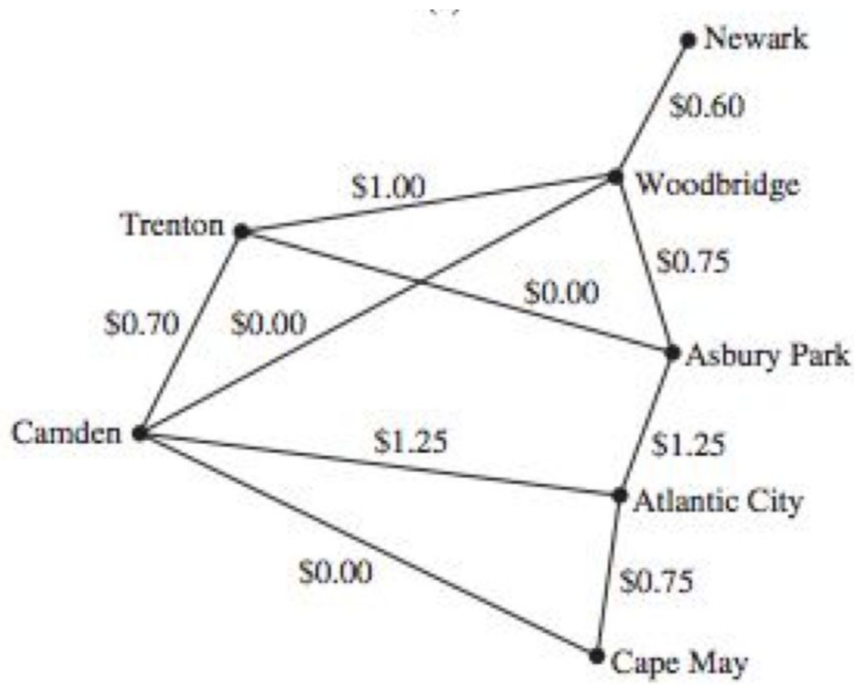
STEP 4:

Camden	0
Trenton	30
Woodbridge	60
Atlantic City	55
Cape May	85
Asbury Park	70
Newark	80

Success! We can stop our algorithm. We have marked Woodbridge as visited, and it is the only way to get to Newark, so we have found our shortest path. Thus, we follow the path from Newark to Woodbridge to Camden to get a path of length 80.

In the second section of part a, we change the endpoint to Cape May, apply the algorithm, and get the path to be Newark to Woodbridge to Camden to Cape May, with length 165.

In part b of the problem, we replace the lengths from part a with the toll costs that we are given below.



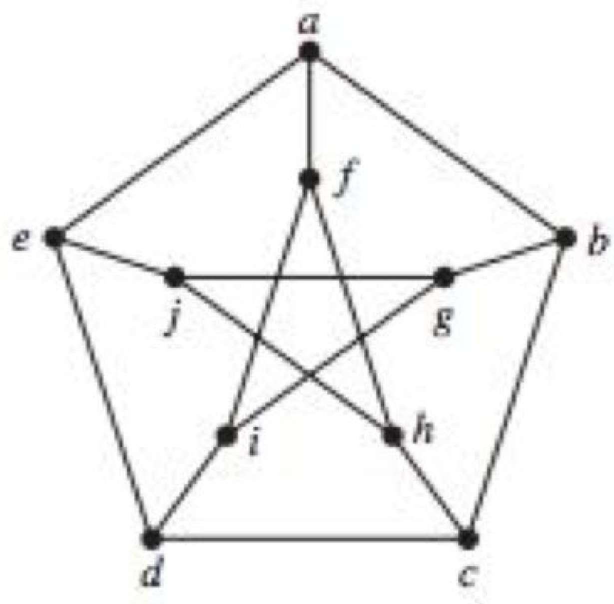
We repeat the algorithm for Newark to Camden and get the route to be Newark to Woodbridge to Camden with a toll of $0.60.

We repeat the algorithm for Newark to Cape May and get the route to be Newark to Woodbridge to Camden to Cape May with a toll of $0.60

2

A Hamiltonian circuit is a circuit that visits every node once with no repeats.. As it is a circuit, it must start and end at the same node. A classic example is a package delivery driver, who must stop at every house (node) and return back to his start. Thus, he must find a circuit that visits every node once with no repeats and returns back to the starting node.

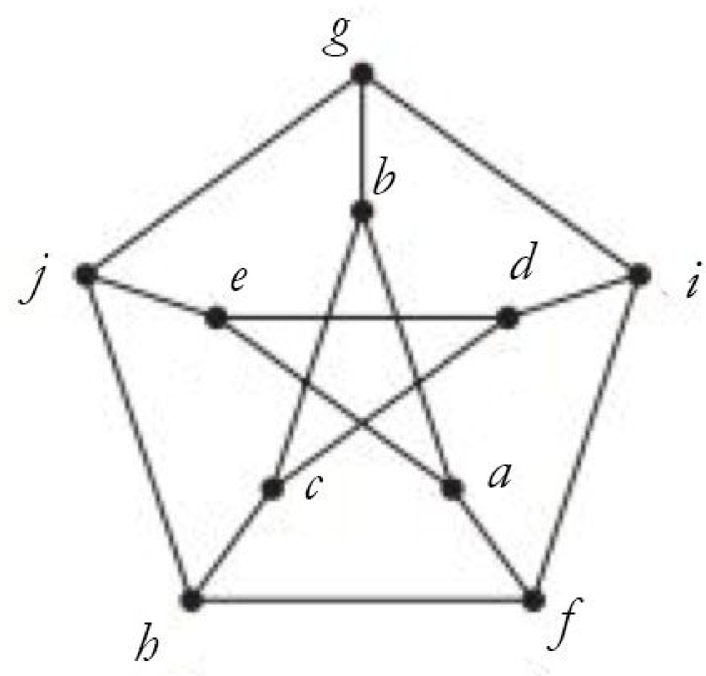
We are given the following graph, known as the Petersen Graph



And asked to explain why it does not have a Hamiltonian circuit.

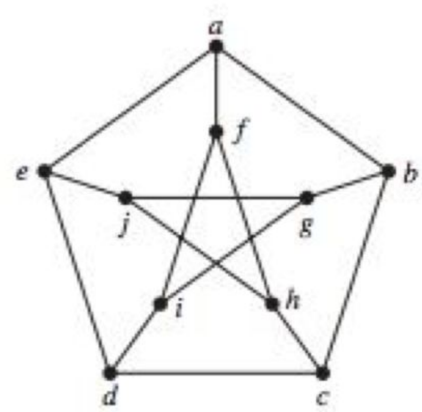
The graph is made up of two groups, the outer edge, *a-e* and the inner edge, *f-j*. While it may seem that the two groups are different, they are in fact the same. Each node in the *a-e* is connected to two other nodes in *a-e* such that there is a cycle between the 5 nodes and each node is connected to one unique node in *f-j*. Similarly, Each node in the *f-j* is connected to two other nodes in *f-j* such that there is a cycle between the 5 nodes and each node is connected to one unique node in *a-e*.

This can be shown if we redraw the graph like this.



It is the exact same graph! Node *a* is connected to *b,e,* and *f*, just as it was before. All of the other nodes have the same connections as before. This helps us to establish symmetry, but as of now doesn't help us in our proof that there can not be a Hamiltonian circuit. We must try a different approach.

I will use the original graph to explain my argument, and include it here for reference.



We will prove this using the pigeonhole principle. The pigeonhole principle states that if there are more pigeons than holes, at least one hole must have at least 2 pigeons.

We can divide the edges into 3 groups, external (*a-b, b-c, c-d, d-e, e-a*), internal (*f-h, h-j, j-g, g-i, i-f*), and middle (*a-f, b-g, c-h, d-i, e-j).* We now have 3 groups of 5 edges each.

A Hamiltonian circuit will, be definition, include 10 nodes and 10 edges, and each node will have 2 edges connected to it. Thus we know that we will have 10 edges. By applying the pigeonhole principle to our 3 groups above, it follows that at least one group of edges will have 4 or more edges included in our circuit.

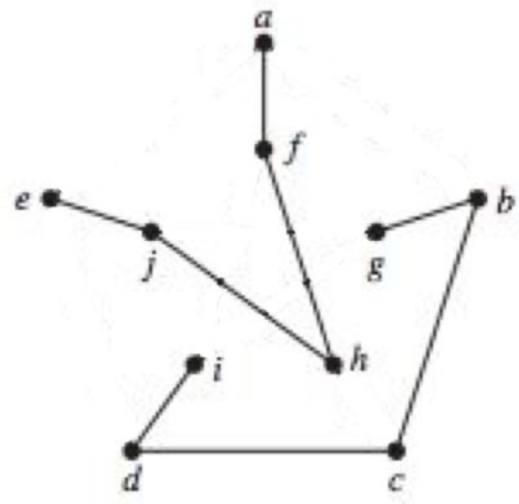
We will first examine the case of 4 edges in the external edges. It obviously can't contain all 5 edges, so without loss of generality, let's assume we use edges *a-b, b-c, d-e* and *e-a*, excluding *c-d*. It follows that edges *a-f, g-b,* and *e-j*, must not be in the circuit, as there would be 3 edges from one node. It also follows that edges *d-i* and *c-h* must be included. This leaves us with 4 edges, and they must be internal edges. Thus, we must connect *i* to *h* with exactly 4 edges. However, there is no such connection. Thus, it is impossible for their to be a Hamiltonian Circuit with 4 external edges.

We have previously proved that the internal edges and external edges are interchangeable. Thus, it is also impossible for their to be a Hamiltonian circuit with 4 internal edges.

This leaves us with 4 middle edges. Lets assume, without loss of generality, that we include edges *a-f, b-g, d-i, e-j.* There are now two separate cases. The first is that we include edge *c-h*, and the second case is that we don't.

We will start with the first case, that we include it. In this scenario, if we have 5 groups of nodes and must connect all 5 of them in a circuit. This requires 5 additional edges. We have 5 edges left, so this is promising. However, the maximum number of external edges we can have in this situation, without creating a node with 3 edges, is 2. This is because each node already has one 1 edge, and when we add 2 edges, we have 4 nodes with 2 edges and 1 node with 1 edge. This node with 1 edge must connect to another, but all others already have 2 edges. The same property is true for the internal edges. If we have 2 internal and 2 external, we only have 4 edges, which can't connect all nodes and isn't enough for a circuit. Thus, we can't include all 5 edges required for including edge *c-h* without creating a node with 3 edges, and thus it is impossible to create a Hamiltonian circuit in this scenario.

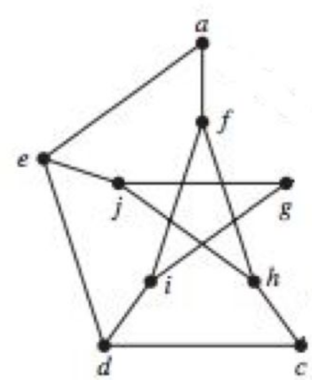
We are left with our last scenario, where we don't include *c-h*. We must then include edges *c-d* and *b-c* in order to include *c*. We must also include edges *h-j* and *f-h* in order to include *h.* We know we can't include edges *a-b* and *d-e*, as they would make nodes *b* and *d* degree 3. We are now left with the below shape.



The only way to connect this graph would be to connect *a* to *g* and *e* to *i*. However, there are no edges there. Thus, it is impossible to create a Hamiltonian circuit in this scenario.

Thus, we have proven that it is impossible to create a Hamiltonian circuit with the Petersen graph.

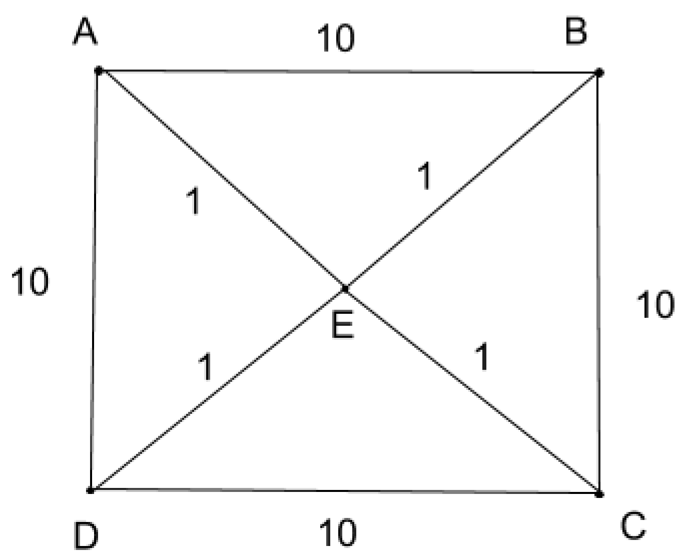
The next step of the problem is to show that if we delete any node, we can create a Hamiltonian circuit. As was shown above, the graph is completely symmetrical, so showing an example of a Hamiltonian circuit by deleting node *b* will show its true for any node. This is simple. We delete node *b* (and its incident edges) and get the below graph.



We can create a circuit *a-e-d-c-h-j-g-i-f-a.* We have gotten rid of one node, which was causing us the problem in creating a circuit, and the graph now has the property of including Hamiltonian circuits.

We now understand this in the context of a social network. We can take the example of Facebook, where a node is a person and an edge is a "friendship" between the two, which allows the two to send messages to each other. Lets now say Person A wants to send a party invitation to everyone in the graph. She wants to pass the message along to everyone in the graph, but she wants everyone to send and receive the message just once. Thus, each Person would have two edges connected to them, and as you move along the circuit, a node coming in would indicate receiving the message and a node coming out would indicate sending the message along. Person A wants it sent back to herself to ensure everyone has copied the details correctly, and so she requires a Hamiltonian Circuit, in which each node is included once and the message returns to the starting node. This would be impossible with our original graph. At least one person would have to either send or receive the twice, as it is impossible to connect all the nodes without repeating and return to the original node. However, if Person A decides that she wants to exclude one person from the graph, she would be able to send a message that travels from her to each person in the graph and back to her, accomplishing her goal.

#3



Here we see a graph where the optimal circuit that visits every vertex at least once is optimized by visiting a vertex more than once. The minimized circuit, starting at A, would be

A-E-B-E-C-E-D-E-A , with a total weight of 8. Even going down one of the 10 weighted edges once would be more than this optimized circuit.