

ITEC200 – WEEK 12 CONCEPTUAL SOLUTIONS

This worksheet is comprised of some questions that students do as preliminary work and then some group-work activities that are completed in online classes. The first part of each online class is focused upon discussing the preliminary solutions that students have submitted. (Note that students must submit all preliminary work by 12pm the day before the online class by using the “Submit Preliminary Activities” link from the unit homepage.)

PRELIMINARY QUESTIONS

QUESTION 1

What are two common methods of representing graphs? Can you think of other methods? Perhaps your new representation is just a modification of one of the common methods.

The two methods of representing graphs are as adjacency matrix or as adjacency list.

QUESTION 2

Under what circumstances would it be best to use one type of representation (from question 1) over another?

If you have very few edges, adjacency list is better. If you have lots of edges, adjacency matrix is better.

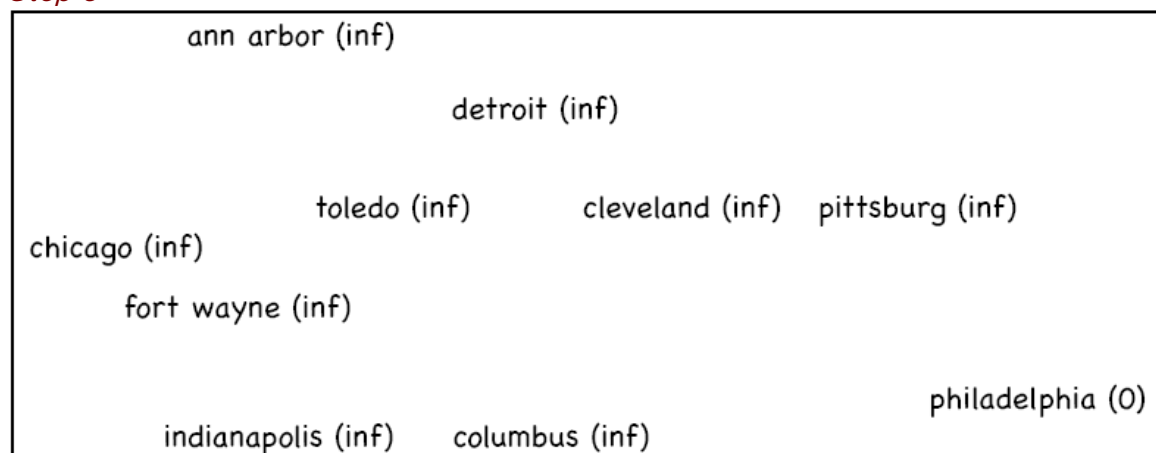
QUESTION 3

Trace the execution of Dijkstra’s algorithm to find the shortest path from Philadelphia to the other cities shown in the following graph (check *Koffman Self-Check 12.6.1*). This means to must find the shortest distance form Philadelphia to each of the other cities.

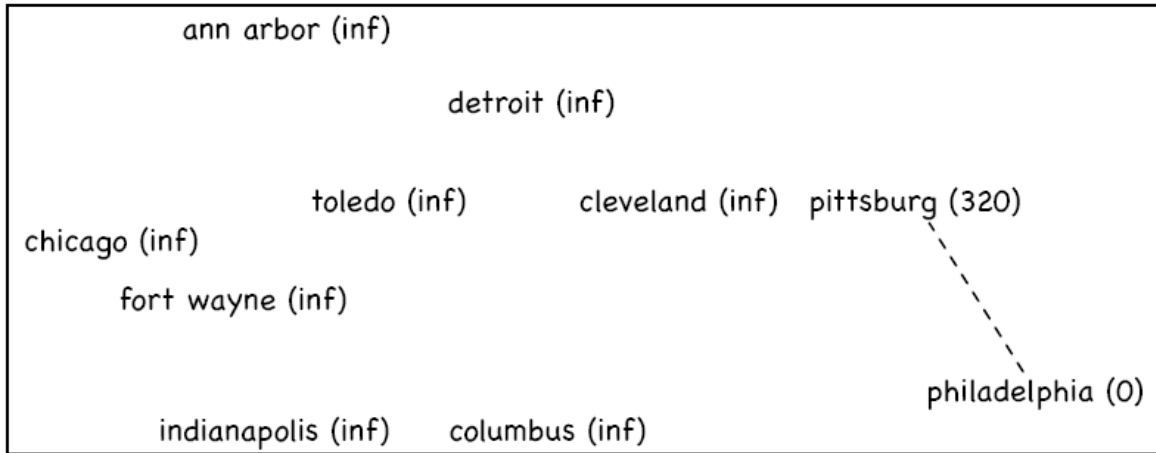
(Koffman Self-Check 12.6.1)

See the table at the end of the trace.

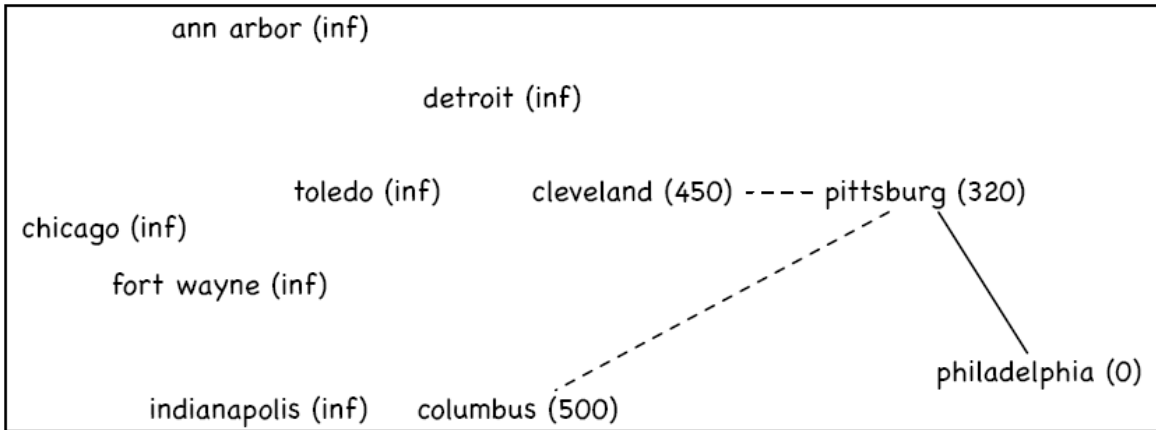
Step 0



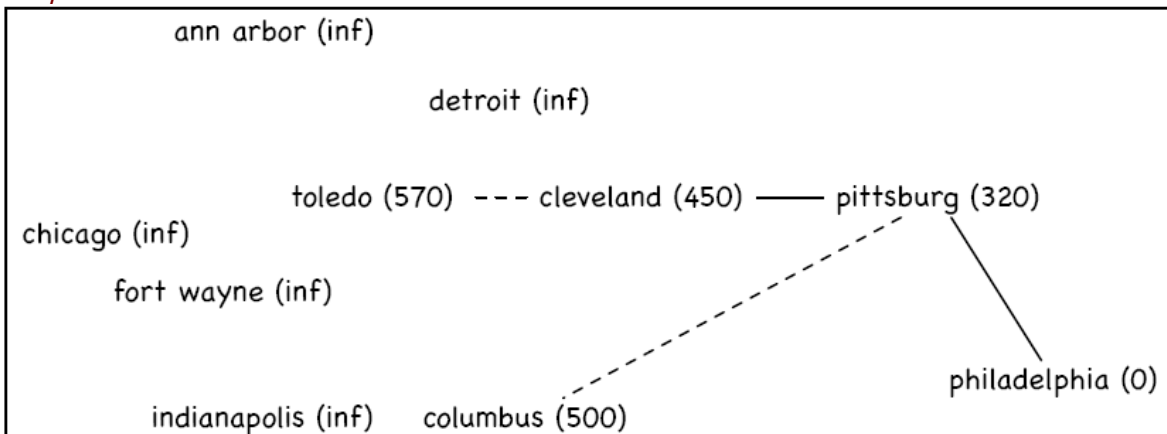
Step 1



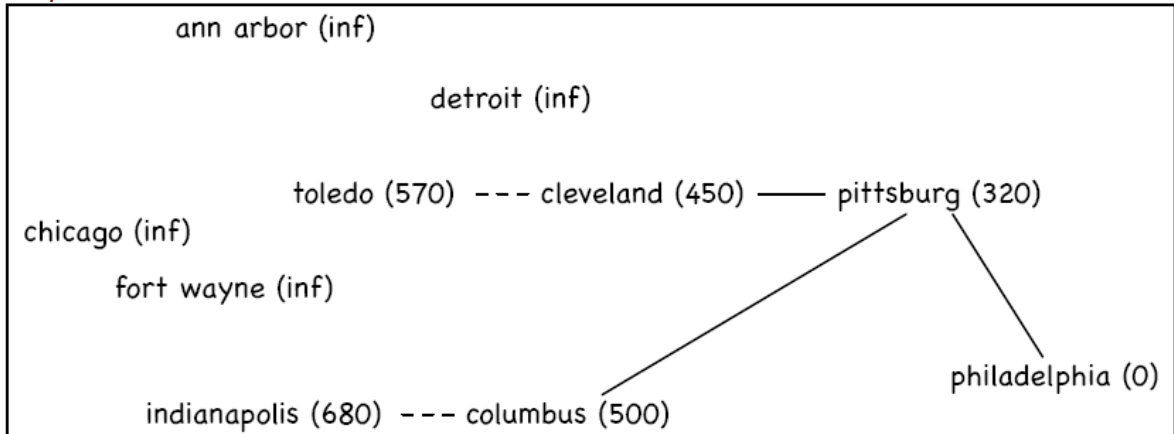
Step 2



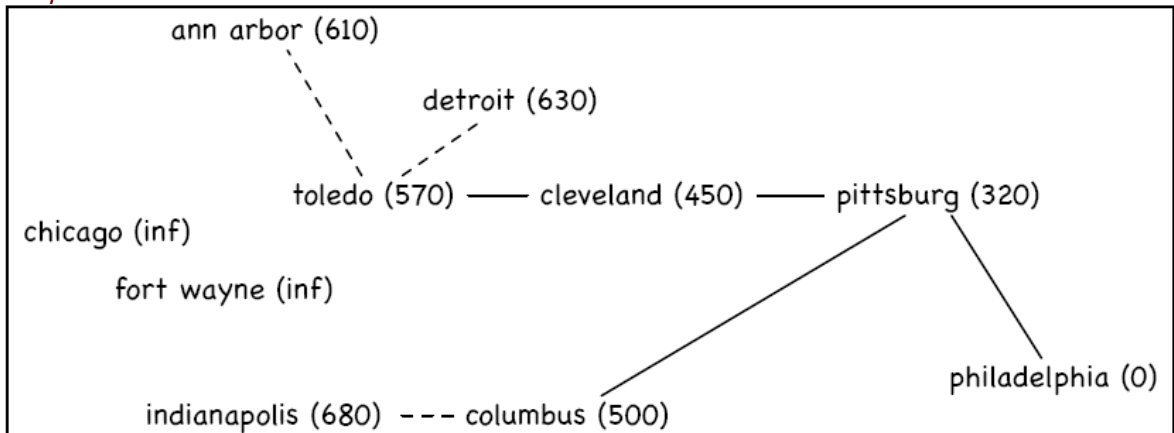
Step 3



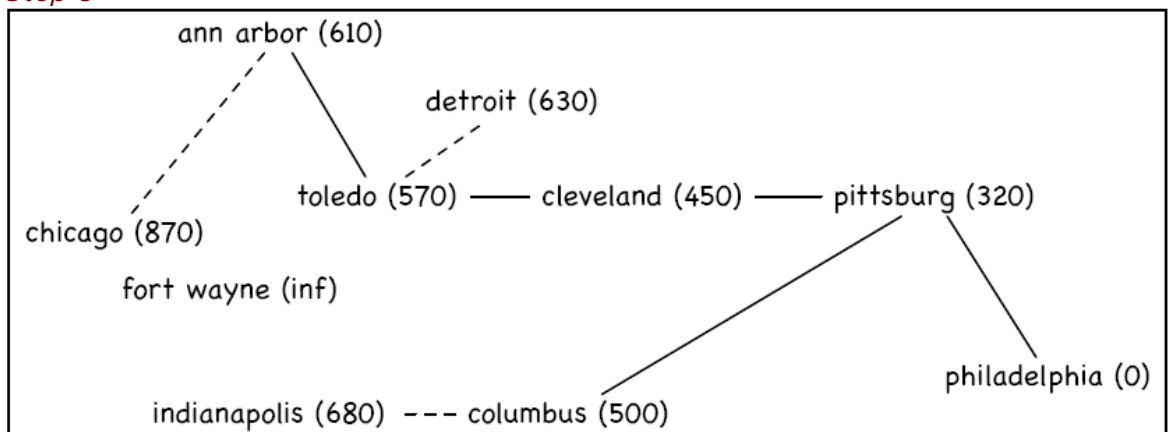
Step 4



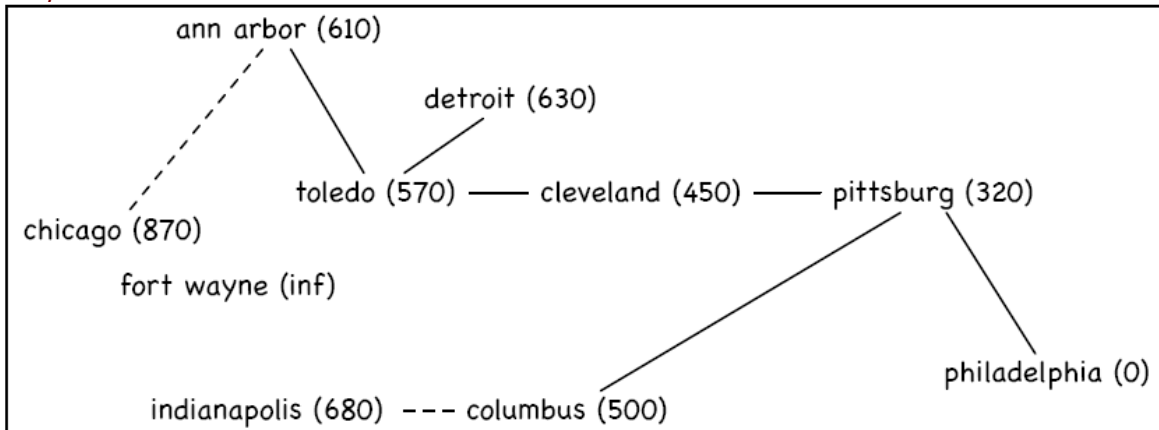
Step 5



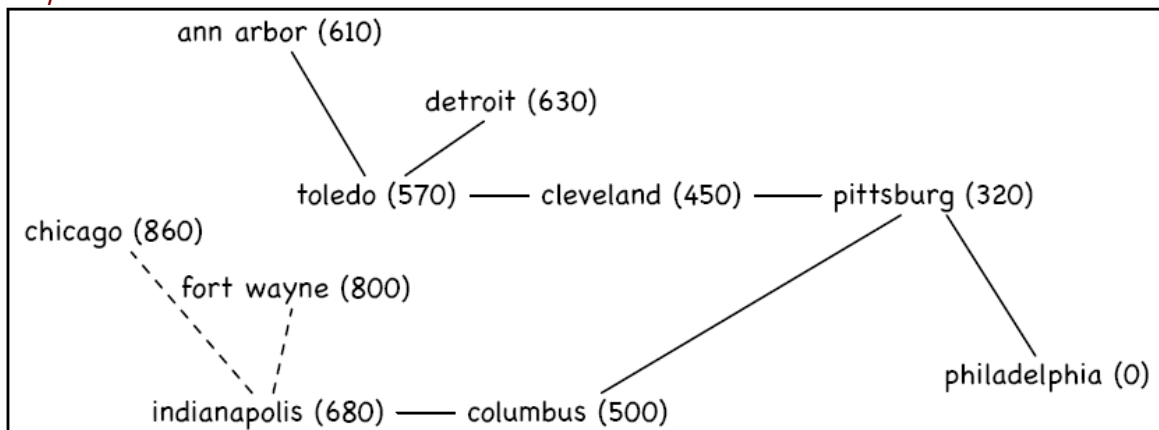
Step 6



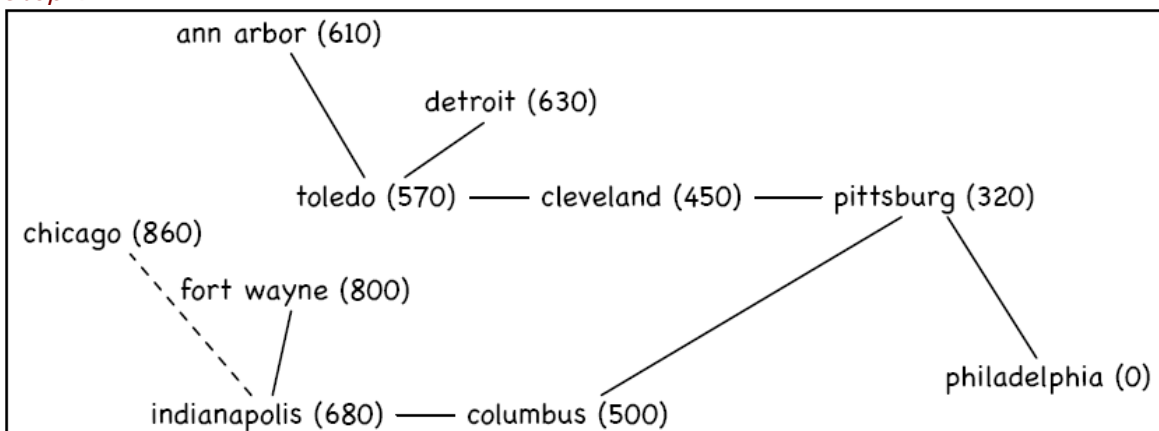
Step 7



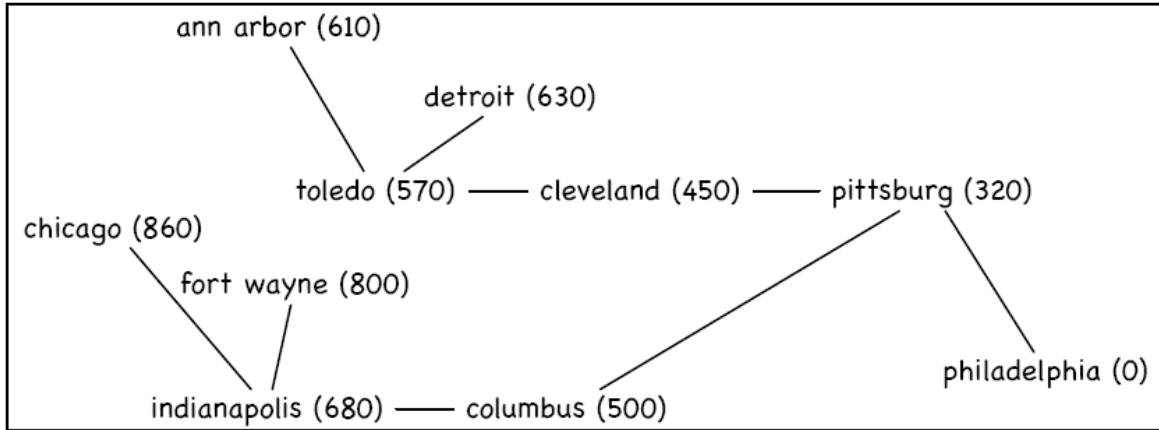
Step 8



Step 9



Step 10

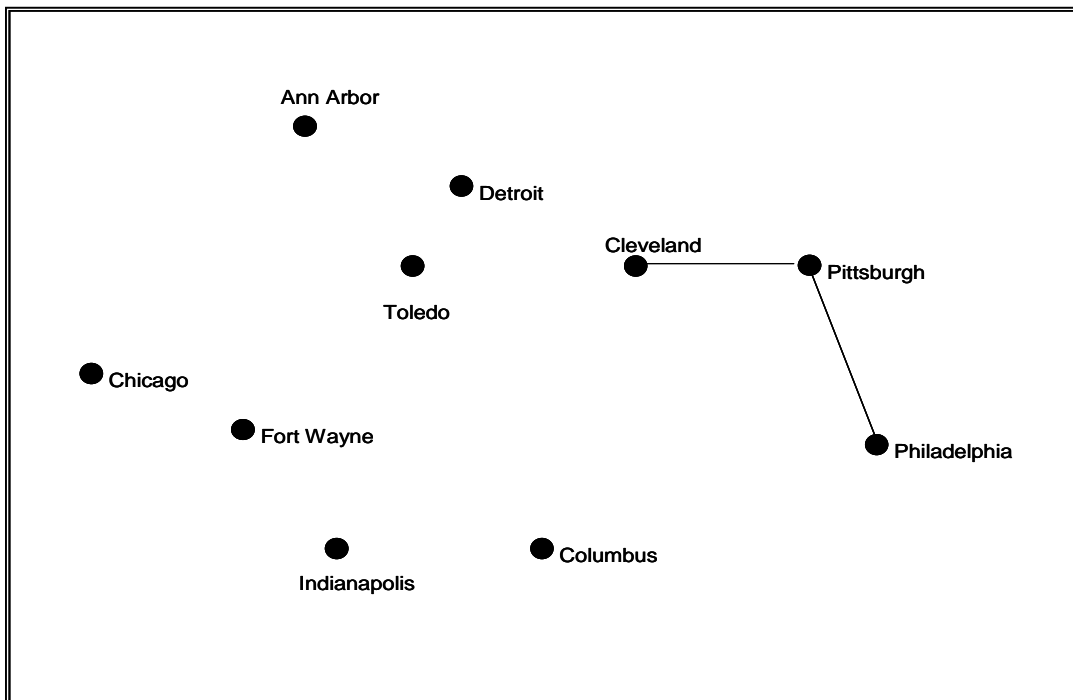
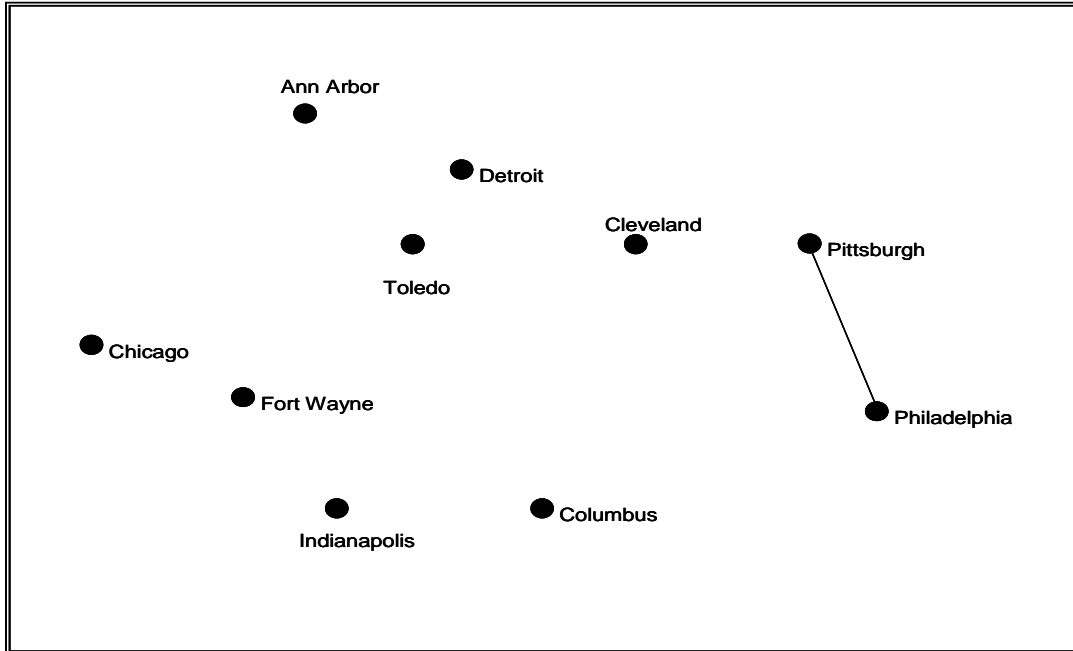


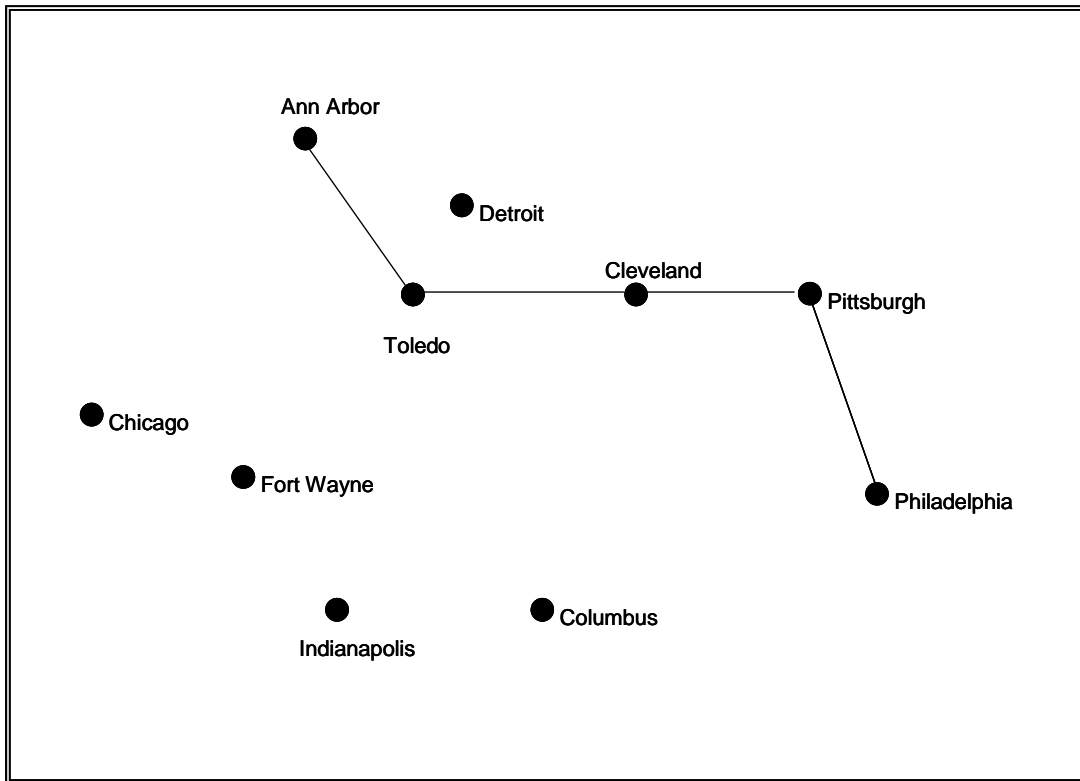
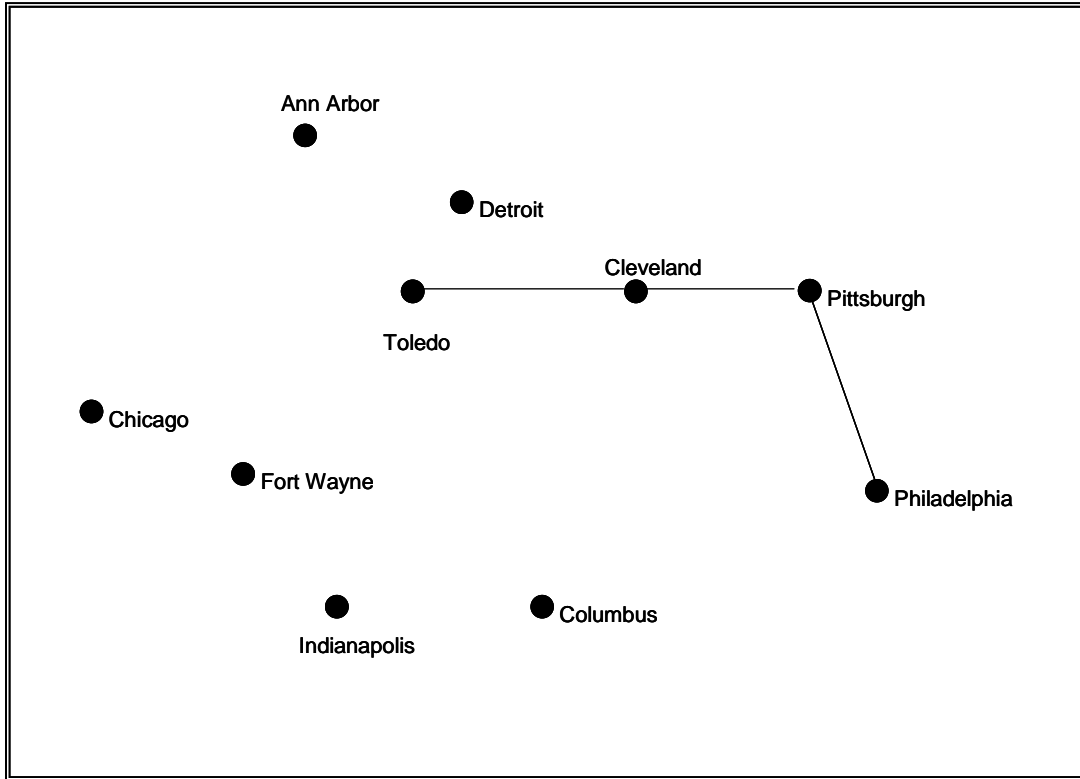
<i>v</i>	<i>u=ph</i> <i>d[v] p[v]</i>	<i>u=pt</i> <i>d[v] p[v]</i>	<i>u=cl</i> <i>d[v] p[v]</i>	<i>u=col</i> <i>d[v] p[v]</i>	<i>u=tol</i> <i>d[v] p[v]</i>	<i>u=ah</i> <i>d[v] p[v]</i>	<i>u=dt</i> <i>d[v] p[v]</i>
<i>Pitt</i>	320 <i>ph</i>	320 <i>ph</i>	320 <i>ph</i>	320 <i>ph</i>	320 <i>ph</i>	320 <i>ph</i>	320 <i>ph</i>
<i>Clev</i>		450 <i>pt</i>	450 <i>pt</i>	450 <i>pt</i>	450 <i>pt</i>	450 <i>pt</i>	450 <i>pt</i>
<i>Col</i>		500 <i>pt</i>	500 <i>pt</i>	500 <i>pt</i>	500 <i>pt</i>	500 <i>pt</i>	500 <i>pt</i>
<i>Tol</i>			570 <i>cl</i>	570 <i>cl</i>	570 <i>cl</i>	570 <i>cl</i>	570 <i>cl</i>
<i>Det</i>					630 <i>tol</i>	630 <i>tol</i>	630 <i>tol</i>
<i>AH</i>					610 <i>tol</i>	610 <i>tol</i>	610 <i>tol</i>
<i>Chi</i>						870 <i>ah</i>	870 <i>ah</i>
<i>FW</i>							
<i>Ind</i>				680 <i>col</i>	680 <i>col</i>	680 <i>col</i>	680 <i>col</i>

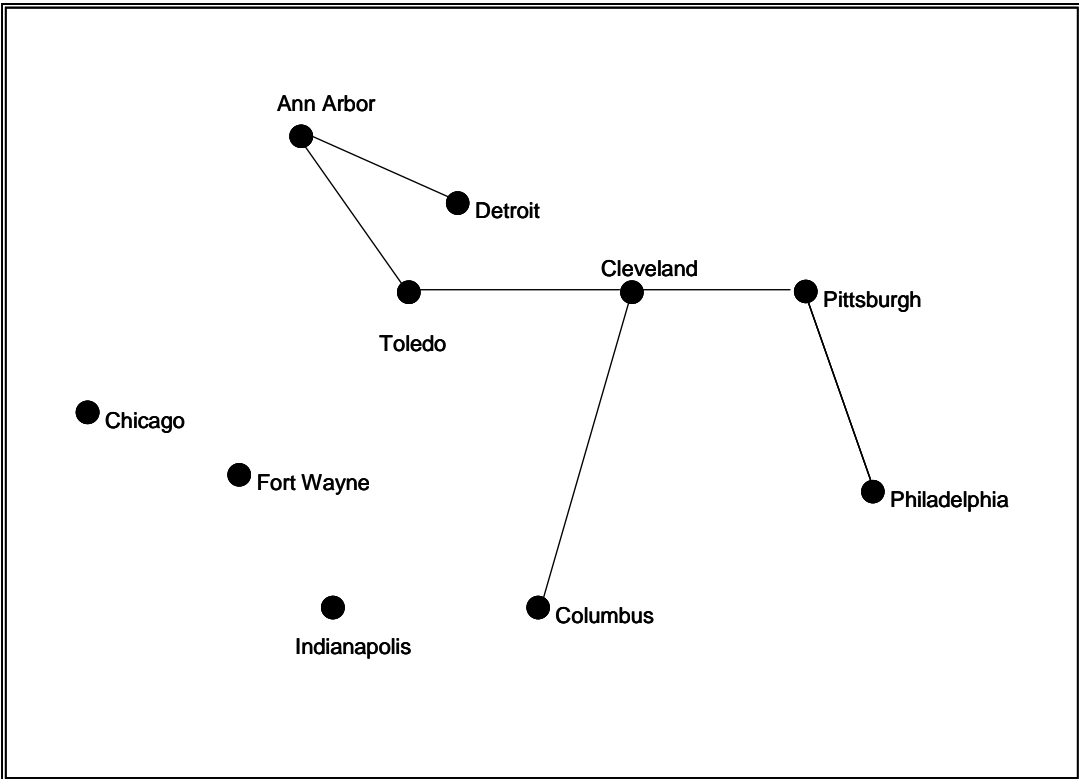
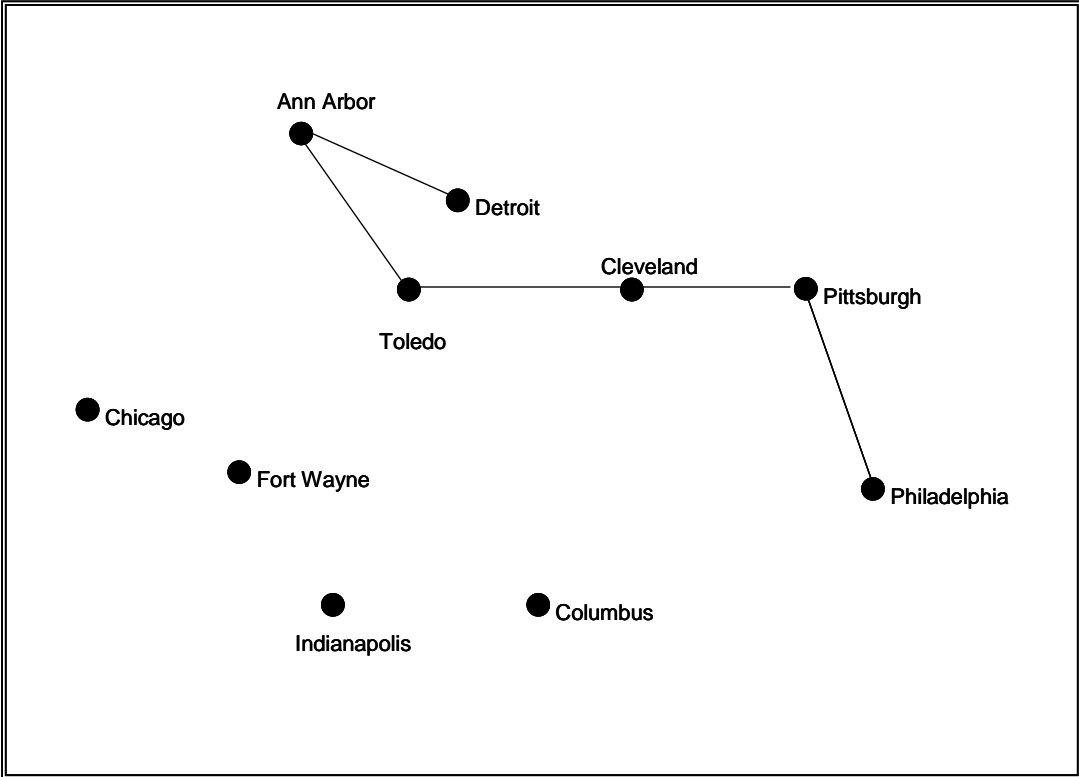
<i>v</i>	<i>u=ind</i> <i>d[v] p[v]</i>	<i>u=fw</i> <i>d[v] p[v]</i>	<i>u=ch</i> <i>d[v] p[v]</i>
<i>Pitt</i>	320 <i>ph</i>	320 <i>ph</i>	320 <i>ph</i>
<i>Clev</i>	450 <i>pt</i>	450 <i>pt</i>	450 <i>pt</i>
<i>Col</i>	500 <i>pt</i>	500 <i>pt</i>	500 <i>pt</i>
<i>Tol</i>	570 <i>cl</i>	570 <i>cl</i>	570 <i>cl</i>
<i>Det</i>	630 <i>tol</i>	630 <i>tol</i>	630 <i>tol</i>
<i>AH</i>	610 <i>tol</i>	610 <i>tol</i>	610 <i>tol</i>
<i>Chi</i>	860 <i>ind</i>	860 <i>ind</i>	860 <i>ind</i>
<i>FW</i>	800 <i>ind</i>	800 <i>ind</i>	800 <i>ind</i>
<i>Ind</i>	680 <i>col</i>	680 <i>col</i>	680 <i>col</i>

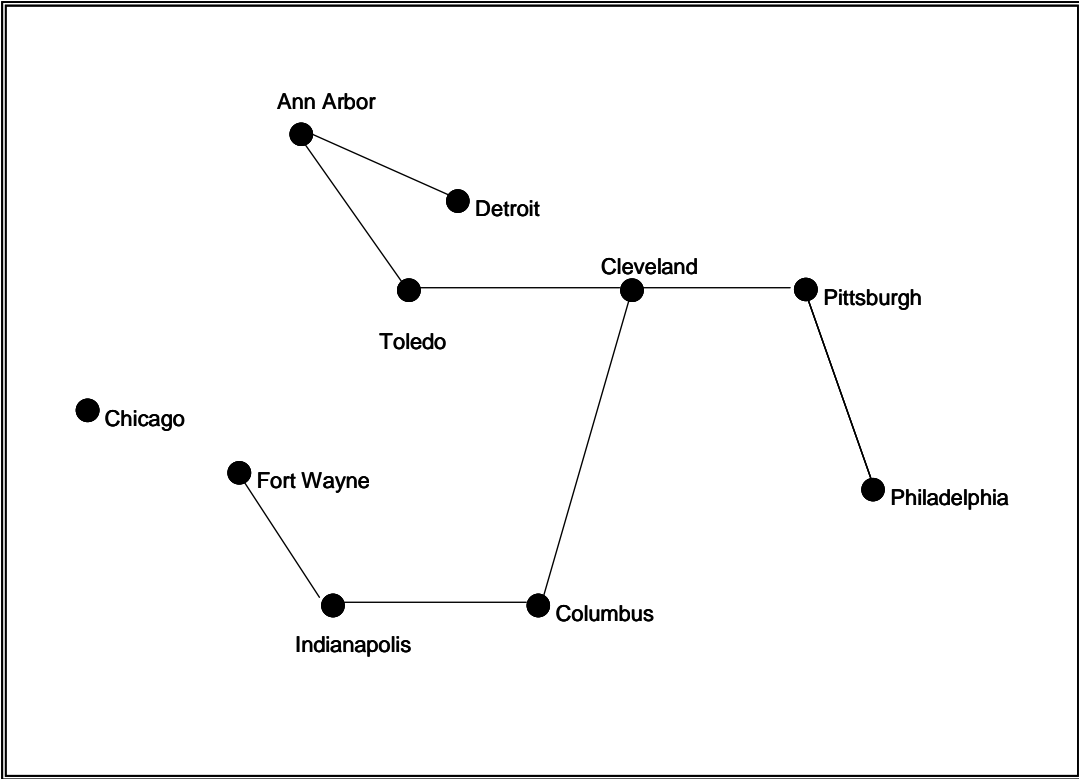
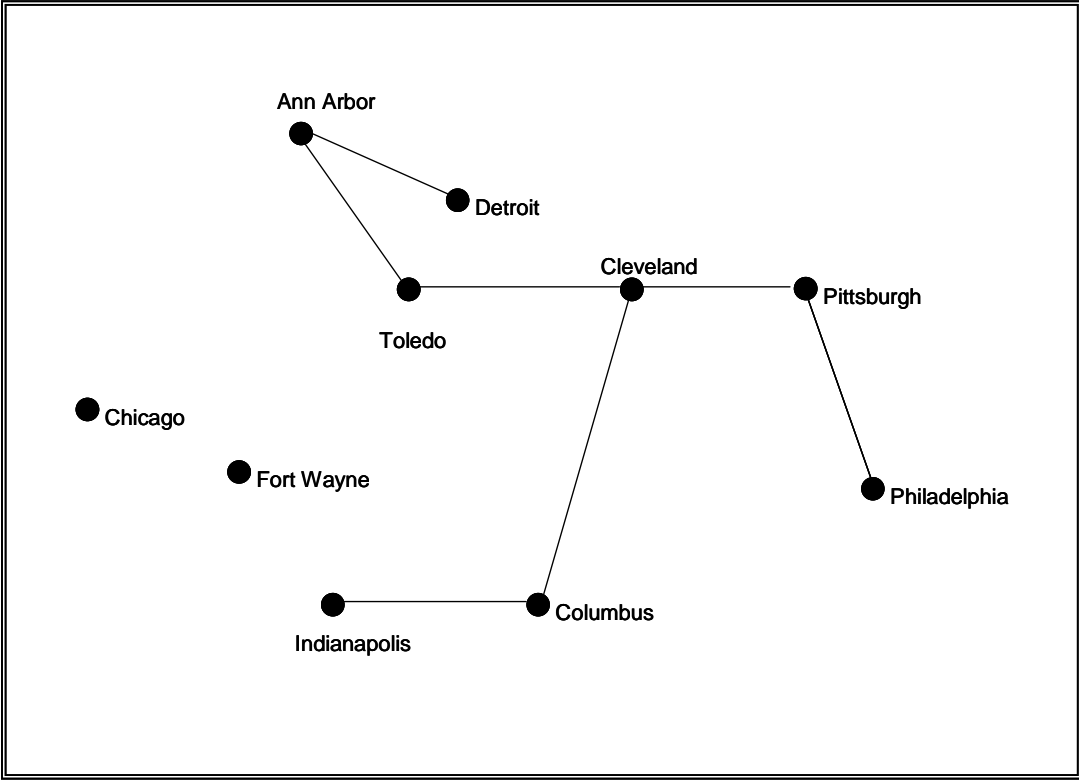
QUESTION 4

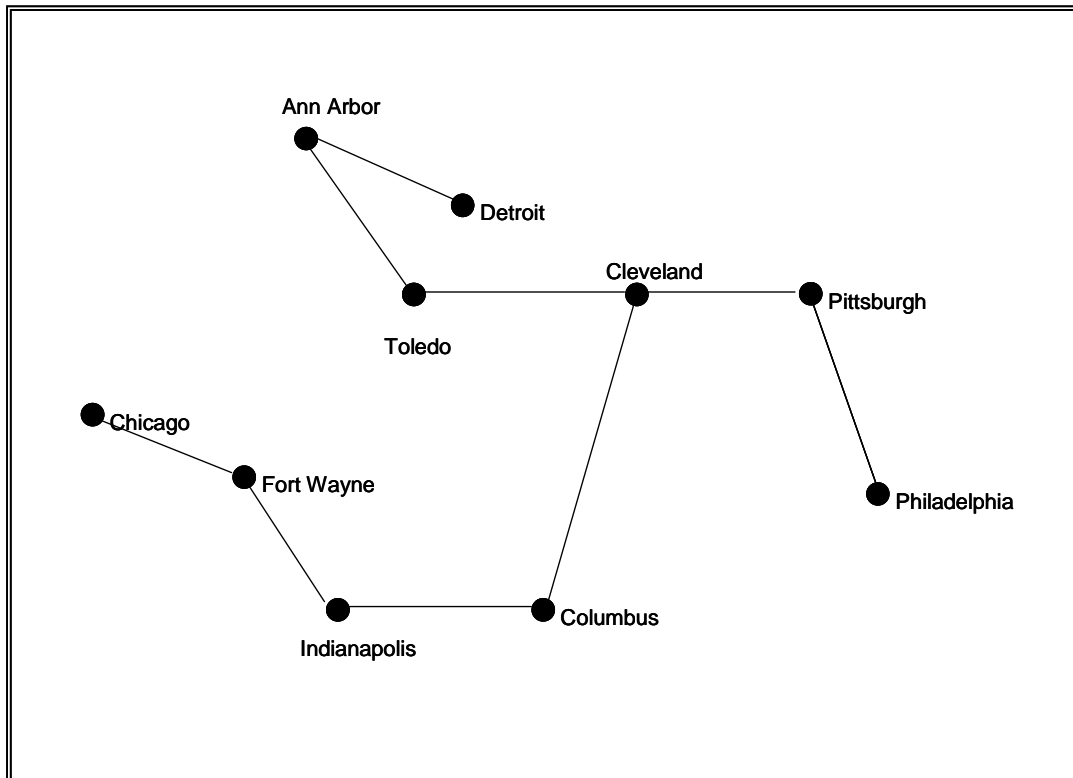
Trace the execution of Prim's algorithm to find the minimum spanning tree from the graph shown in Question 3 (graph in *Koffman Self-Check 12.6.1*).
(*Koffman Self-Check 12.6.4*)











CLASS ACTIVITIES

Complete the following activities in groups, asking questions of each other. This will form the basis for class discussion.

ACTIVITY 1 – CROSS CHECK (10 MINUTES)

Review each other's solutions to the preliminary tasks, helping each other where necessary. Check that you all agree on your solutions. Ask questions, making sure that you totally understand the material.

ACTIVITY 2 – DIJKSTRA'S FRONTIER (20 MINUTES)

Dijkstra's algorithm is often described in terms of a frontier spreading across the graph. Initially, only the start node is in the frontier. At each step, the node with the smallest path that is in the frontier is chosen to be the current node. All nodes that can be reached from the current node are put in the frontier. If they are already in the frontier, they have their distance, (potentially updated) any node that can be reached from this one but has been through the frontier is not added to the frontier again. Once all the nodes have been in and out of the frontier, the algorithm is completed.

This is the exactly the same algorithm as normal Dijkstra's, just described in a different way. Reconcile this description of Dijkstra's with the algorithm presented in the text.

See the explanation in the book and at the online class.