ENG 115 Engineering Applications of OR Fall 1998 Handout 3

The shortest path problem

Consider the following problem. You are given a map of the city in which you live, and you wish to figure out the fastest route to travel from your home to your office. In your city, some of the streets are two-way, and some are one-way. Furthermore, traveling down a street in one direction might not take the same time as in the other direction (e.g, if there is some construction taking place on your side of the street).

First of all, we would like to give a mathematical model of this problem. To do this, it will be useful to introduce the notion of a *directed graph.* A directed graph consists of a set of nodes, and a set of arcs. For example, the picture below shows a graph in which 1, 2, 3, 4, 5, and 6 are the nodes of the graph. That is, in drawing a graph we represent a node by a circle with its name indicated inside. An arc is an ordered pair of nodes, such as $(1, 2)$. The arc $(1, 2)$ is represented below as the arrow that points *from* node 1 *to* node 2. For nodes 2 and 3, there is an arc from 2 to 3 and an arc from 3 to 2. Thus, if we consider the graph below, then the set of nodes is $\{1, 2, 3, 4, 5, 6\}$ and the set of arcs is

$$\{(1, 2), (1, 3), (2, 3), (2, 4), (3, 2), (3, 5), (4, 3), (4, 6), (5, 2), (5, 6)\}.$$

If we let N be the name for the set of nodes, that is, $N = \{1, 2, 3, 4, 5, 6\}$, and if we let A be the name for the set of arcs then

$$A = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 2), (3, 5), (4, 3), (4, 6), (5, 2), (5, 6)\}.$$

When we specify the elements that are contained in a set, then it does not matter in which order we list them. So for example, we could equally well have described N as $\{1, 3, 4, 6, 5, 2\}$; that is the same set. A graph consists of a set of nodes and a set of arcs; hence, if we call the graph G, then we often write that $G = (N, A)$ to mean that N is its set of nodes, and A is its set of arcs.

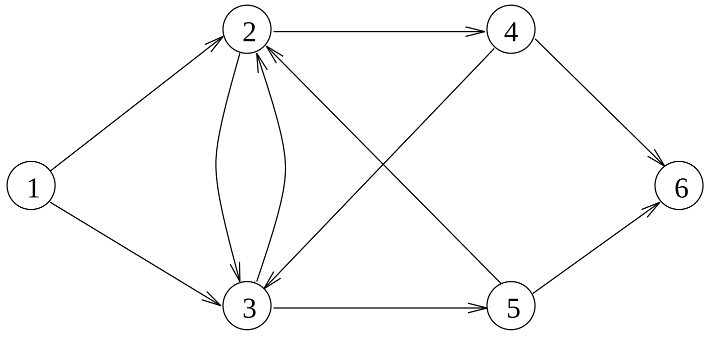

Figure 1: A graph with 6 nodes and 10 arcs

A path in a graph is a sequence of arcs that, from a visual perspective, you could follow with your pencil without lifting the pencil up. For example, $(2, 3), (3, 5), (5, 6)$ is a path from node 2 to node 6 in the graph given in Figure 1. There are two important things to notice. First, a path is a sequence of arcs, not a set of arcs: the order in which we list the arcs *does* matter. Second, we are following each arc in its given direction. For example, $(3, 2), (2, 1)$ is not a path from node 3 to node 1, since there is no arc $(2, 1)$ in the graph in Figure 1; only $(1, 2)$ is an arc in this graph. In general, we can write a path as follows: let i_1, i_2, ..., i_k denote nodes in the graph (not necessarily the nodes $1, 2, \ldots, k$); then

$$(i_1, i_2), (i_2, i_3), (i_3, i_4), \ldots, (i_{k-1}, i_k)$$

is a path in the graph from node i_1 to node i_k provided that each of (i_1, i_2), (i_2, i_3) through (i_{k-1}, i_k) is an arc in the graph. This path has $k-1$ arcs in it.

We will often be interested in directed graphs for which each arc has an associated length. We will denote the length of each arc (i,j) in A by $\ell(i,j)$. In the graph below, we have added lengths by writing each arc's length right next to it. For example, the length of arc $(3,2)$ is 5, or equivalently, $\ell(3,2) = 5$. The length of a path is the sum of the lengths of the arcs in it. For example, the path from node 2 to node 6 given above, that is, $(2,3), (3,5), (5,6)$, has length equal to $3 + 1 + 2 = 6$. In this case, there are two paths from node 2 to node 6 of length 6. Can you find another one? We will be interested in finding the shortest path between a given pair of nodes. This is the next optimization model that we shall consider in this course.

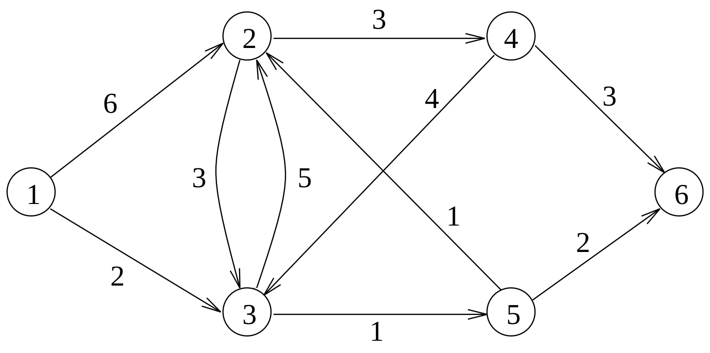

Figure 2: A graph with arc lengths

The *shortest path problem* can be stated as follows: given a directed graph $G = (N, A)$, and a specified *source* node s (which is in N), where each arc (i,j) in A has a specified non-negative length $\ell(i,j)$, for each node i in N find a shortest path from s to i. (Unlike the traveling salesman problem, we do not need to go through all other nodes on the way; we just want *the* shortest path to get there.)

As the next step, we shall explain why this problem can be used to model our problem of finding the quickest way to the office. We can model the map of our city by a graph as follows. Introduce a node for each intersection on the map. For each pair of intersections (say, 7th Ave. & 33rd St. and 7th Ave. & 32nd St.) if there is a street connecting them (going the right way) and this street does not cross any other intersection along the way, we introduce an arc from the node corresponding to the first intersection, to the node corresponding to the second intersection. In our example, 7th Ave. is one-way going downtown, and so there is only an arc from the first to the second of them, and not from the second to the first. The length of an arc is the length of time to drive between the two intersections in that direction. By solving the shortest path problem for the graph derived from our city map, we can compute routes in the city.

Next we need an algorithm to solve this mathematical model. In this case, we will be able to give a very simple algorithm that for any input, finds an optimal solution. First think about finding a node that is closest to the source s. In some trivial sense, s is the closest node to itself, so we will set $Closest(1) = s$. Now we wish to find some other node i for which the shortest path from s to i is as short as possible. We shall call this node $Closest(2)$. Clearly, there might be several nodes that are all the same distance from s, but for at least one of these nodes i there is an arc (s,i), since if you can reach i from s only by passing through some other node along the way, that other node must be at least as close to s as i. (Would this still be true if an arc could have a negative length?) So $Closest(2)$ can be identified by considering all arcs of the form (s,i); let (s,i^*) be the arc leaving s that is shortest. Then $Closest(2) = i^*$, and the shortest path from s to i^* consists of the single arc (s,i^*). Next consider identifying the node that is next closest to s (counting ties, so it might be just as close as $Closest(2)$). The shortest path might be just one arc from s, or else it might first pass through $Closest(2)$ and then continue on to it. By considering all of the paths of this form, we can identify the next closest node, $Closest(3)$. We can continue in this way until we have assigned each node to be $Closest(j)$ for some $j = 1, \ldots, n$. At each stage, we know that the shortest path to $Closest(j)$ must consist of the shortest path to $Closest(i)$, for some $i = 1, \ldots, j-1$, and then one arc from $Closest(i)$ to $Closest(j)$.

We will now describe the algorithm to compute the shortest path from s to each other node in the graph in a more formal way. The fact that the algorithm always finds the correct solution is a direct consequence of the previous discussion. Since this algorithm was first proposed by E. Dijkstra, it is commonly called *Dijkstra's algorithm*.

- {Initialize} Set $Label(s) := 0$, and $Label(i) := +\infty$ for all other nodes i in N. Set $j := 0$.
  Let $Prev(i)$ be undefined for each node i in N; all nodes are unmarked.
- {Main Loop} Until all nodes are marked with a $*$ do the following:
  1. Set $j := j + 1$;
  2. Among all unmarked nodes, select a node i for which the label is minimum;
  3. Mark node i with a $*$; Set $Closest(j) := i$;
  4. For each arc of the form (i, j), or in other words, for each arc leaving node i, compare $Label(j)$ with $Label(i) + \ell(i, j)$; if the latter is smaller, then set $Label(j) := Label(i) + \ell(i, j)$, and set $Prev(j) := i$.
     (Note: in fact, it suffices to consider all arcs leaving i that go to unmarked nodes j.)

It is not clear that, when this algorithm finishes, you have computed any path from s to each other node, let alone a shortest path for each of these nodes. Let us first run Dijkstra's algorithm on the above example, where node 1 is the specified source.
Initialization:

Node	1	2	3	4	5	6
Label	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
Prev	–	–	–	–	–	–

Clearly, node 1 is the one to be marked, that is, $i = 1$. There are two arcs leaving node 1: $(1, 2)$ and $(1, 3)$. Since $Label(2) = +\infty$, $Label(1) = 0$, and $\ell(1, 2) = 6$, it follows that we should update $Label(2) = 6$ and $Prev(2) = 1$. This can be interpreted as follows: we have found a shorter path to node 2; take the shortest path to node 1 (which is no path at all) and then take arc $(1, 2)$. Since the node previous to 2 in this path is node 1, we have set $Prev(2) = 1$. Similarly, we set $Label(3) = 2$, and $Prev(3) = 1$.
After the first iteration of the main loop:

Node	1^*	2	3	4	5	6
Label	0	6	2	$+\infty$	$+\infty$	$+\infty$
Prev	–	1	1	–	–	–

Now node 3 is the next one to be marked. There are arcs leaving 3 to nodes 2 and 5. For the first of these, $Label(2) = 6$ whereas $Label(3) + \ell(3, 2) = 2 + 5 = 7$, and so we leave $Label(2)$ unchanged. We did not find an improved path to node 2. For node 5, we set $Label(5) = 3$ and $Prev(5) = 3$. As above, this means that the best path we have found from 1 to 5 consists of taking the best path that we have found from 1 to 3 (which consists of just the arc $(1, 3)$) and then arc $(3, 5)$.
After the second iteration of the main loop:

Node	1^*	2	3^*	4	5	6
Label	0	6	2	$+\infty$	3	$+\infty$
Prev	–	1	1	–	3	–

Now node 5 is the next node to be marked. There are two arcs leaving node 5, to nodes 2 and 6. In the former case, we discover a path of length 3+1=4 to node 2, and so we set $Label(2) = 4$ and $Prev(2) = 5$. For the latter, we set $Label(6) = 3 + 2 = 5$, and $Prev(6) = 5$.
After the third iteration of the main loop:

Node	1^*	2	3^*	4	5^*	6
Label	0	4	2	$+\infty$	3	5
Prev	–	5	1	–	3	5

Check that you get the results tabulated below for the remainder of the execution of the algorithm on this graph. The only point to mention is that in processing the arcs leaving node 2 (the next marked node) we need only consider the arc $(2,4)$, since the arc $(2,3)$ leads to a node that is already marked.
After the fourth iteration of the main loop:

Node	1^*	2^*	3^*	4	5^*	6
Label	0	4	2	7	3	5
Prev	–	5	1	2	3	5

After the fifth iteration of the main loop:

Node	1^*	2^*	3^*	4	5^*	6^*
Label	0	4	2	7	3	5
Prev	–	5	1	2	3	5

And finally, after the sixth iteration of the main loop:

Node	1^*	2^*	3^*	4^*	5^*	6^*
Label	0	4	2	7	3	5
Prev	–	5	1	2	3	5

By now it should be clear how to deduce the shortest paths from this information. Take node 4, for example. We get there by coming from $Prev(4) = 2$. But how do we get to node 2? From node $Prev(2) = 5$. And we get to node 5 from node $Prev(5) = 3$. And we get to node 3 from node $Prev(3) = 1$, which is the source. (We can detect that we have traced back to the source by the fact that its $Prev(\cdot)$ value is still undefined. So the shortest path from node 1 to node 4 is $(1,3),(3,5),(5,2),(2,4)$.

While we can perform this tracing-back each time we wish to determine a shortest path, we can also give a nice way to concisely describe all of the shortest paths. For each node i that is not the source, highlight the arc from $Prev(i)$ to i. This is done for our example in the figure below. This collection of arcs is called

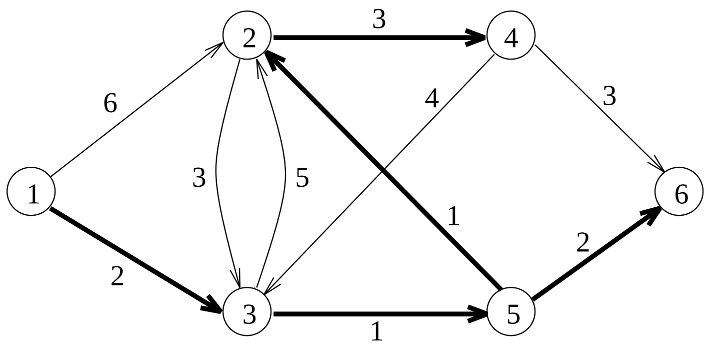

Figure 3: Shortest path tree

the *shortest path tree*. With an active imagination (and holding this piece of paper sideways), you can think of this as a tree growing up from the source node. Its importance should be clear. For each node i we have highlighted the shortest path from 1 to i.

In the remainder of this handout we shall explain a simple way to verify that you have computed correctly the shortest path between two nodes in a given graph, without having to rerun the entire algorithm. Suppose that the input is a directed graph $G = (N, A)$, where each arc $(i,j) \in A$ (the $\in$ symbol means "is an element of") has a given length $\ell(i,j) \geq 0$, and you wish to compute the shortest path from a given node s to each other node in the graph. In fact, you have run Dijkstra's algorithm, and computed that the shortest path from s to one node w consists of the the $k+1$ arcs $(s,i_1),(i_1,i_2),\ldots,(i_{k-1},i_k),(i_k,w)$. You want to know some easy way to verify that you have computed the correct path, other than just running the algorithm again to see that you did each step correctly.

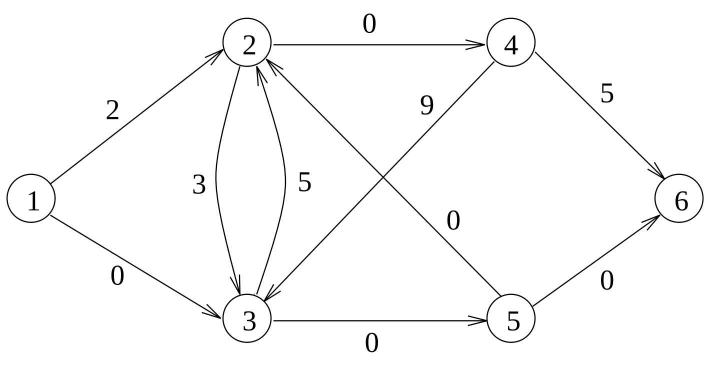

Figure 4: An easy shortest path input

Consider the input in Figure 4. Can you give a convincing argument that you know that shortest path from node 1 to each other node? (Think about this before reading on!!)

Since each arc length is nonnegative, then the length of any path is nonnegative. However, for each node i in Figure 4, it is quite easy to identify a a path of total length 0 from the source to node i. Since all path lengths are non-negative, then certainly a path of length 0 is a shortest path (because no path of negative length exists!) Of course, as in the example above, if the length of the whole path is 0, then the length of each arc in it must also be 0. (Because, once again, there are no negative length arcs.) This seems like a very special case, but the end conclusion will be that it is still a very useful and powerful idea.

The next step will be to consider a rather peculiar variant of the shortest path problem. In this new problem, we are given a graph as in the usual shortest path problem, plus each node $i \in N$ has a special price $p(i)$. In this new problem, we view the nodes as representing cities and the arcs as roads connecting them. When we travel along an arc $(i,j) \in A$ we incur a cost equal to its length $\ell(i,j)$; in addition, whenever we enter a city we are given a present meant to entice us to stay in that city of value $p(i)$. If we leave that city, we must pay that amount back. Once again, we return to our original input, as given in Figure 2, and add such p values, where each node's value is specified in a box right next to it.

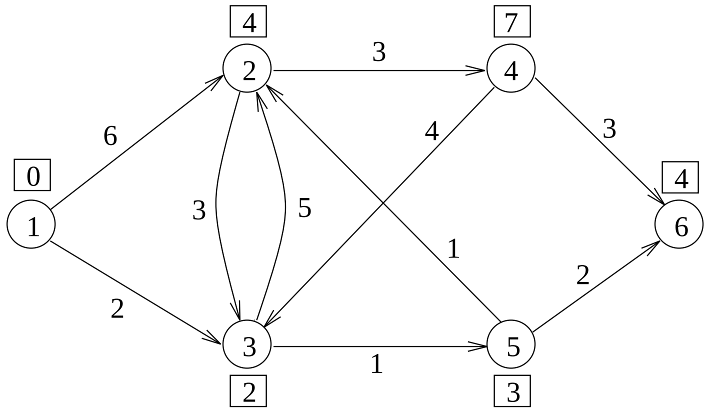

Figure 5: A graph with node enticements

In general, what is the cost of our path from the source node s to node w (for our new problem)? Well, we must pay $p(s)$ to leave the source, and we will get $p(w)$ when we finally enter node w. All of the other presents acquired en route must be paid back, so that the total cost is

$$\ell(s,i_1) + \ell(i_1,i_2) + \cdots + \ell(i_{k-1},i_k) + \ell(i_k,w) + p(s) - p(w).$$

A shorthand notation for this is to write it as

$$\ell(s,i_1) + \sum_{j=2}^{k} \ell(i_{j-1},i_j) \quad + \ell(i_k,w) + p(s) - p(w).$$

So we can think of the total cost of this path as its total length with respect to the original length function ℓ plus $(p(s) - p(w))$. But this is true no matter which path from s to w we consider! Therefore, the cheapest path in this new setting is exactly the same as the shortest path for the original lengths. (Make sure you understand exactly why this is!) We have obtained an equivalent problem to solve; a path that is shortest for this new variant must be a shortest path in original sense, and vice versa. (Make sure you understand this; thinking about the specific example given in Figure 5 is probably helpful.)

Here is another view of the new problem, however. We would like to get rid of the fact that there are these two types of costs, arc lengths and "node enticements", but still leave the problem unchanged, even in computing the cost of any path correctly. Here is a simple idea that might be seem a bit odd at first. Think about using an arc (i,j). To use it, one must first leave node i, then traverse arc (i,j), and then enter node j. All are required if we are to use arc (i,j) at all. So the effective cost of traversing this arc is $p(i) + \ell(i,j) - p(j)$. We define the *adjusted* length of an arc (i,j) of the graph to be

$$\bar{\ell}(i,j) = \ell(i,j) + p(i) - p(j).$$

The total adjusted length of a path is the sum of the adjusted lengths of arcs in that path. It should be clear that the adjusted length of any path is exactly the quantity that we wanted to minimize in the node enticement version of the shortest path problem. Just to double check, let's compute the total adjusted length of our given path from s to w.

$$\begin{aligned}
\text{Total adjusted length} &= \bar{\ell}(s,i_1) + \bar{\ell}(i_1,i_2) + \cdots + \bar{\ell}(i_{k-1},i_k) + \bar{\ell}(i_k,w) \\
&= [p(s) + \ell(s,i_1) - p(i_1)] + [p(i_1) + \ell(i_1,i_2) - p(i_2)] + \cdots \\
&\quad + [p(i_{k-1}) + \ell(i_{k-1},i_k) - p(i_k)] + [p(i_k) + \ell(i_k,w) - p(w)] \\
&= \ell(s,i_1) + \ell(i_1,i_2) + \cdots + \ell(i_{k-1},i_k) + \ell(i_k,w) + p(s) - p(w)
\end{aligned}$$

which is exactly what we wanted the total adjusted length to be. The adjusted lengths for the example given in Figure 5 are given in the figure below.

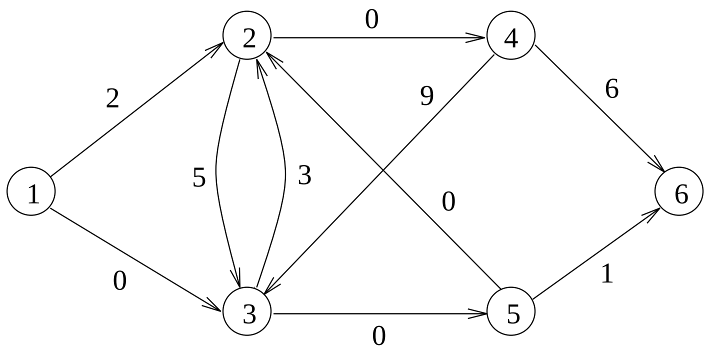

Figure 6: Adjusted arc lengths

But what does this have to do with verifying that we got the correct answer for the shortest path from s to w in our original problem? First, let's summarize what we just figured out. We give each node i a value $p(i)$ (any value is possible). If we consider the problem where we try to find a shortest path from s to w with respect to the adjusted arc lengths $\bar{\ell}(i,j) = \ell(i,j) + p(i) - p(j)$ (for each $(i,j) \in A$) instead of the original ones $\ell(i,j)$, then the shortest path found is also a shortest path for the original lengths. So we could solve the adjusted problem instead of the original one, if that turns out to be easier.

But how do we set the values $p(i)$ for each node $i \in N$? Suppose we let $p(i)$ = length of the shortest path from s to i. (If we have run Dijkstra's algorithm correctly, we presumably know these.) Do this for the graph in Figure 2; after all, we have already run Dijkstra's algorithm for this graph, and the output from the algorithm gives us the proposed p value for each node. I claim that in computing the adjusted costs with these p values, you will rederive one of the figures given above. Which one is it? Do this exercise before continuing to read.

Next we will show that some of the properties of the adjusted lengths that you have just computed are not at all coincidental, and hold when you perform this procedure for any graph whatsoever.
Claim 1. If, for each $i \in N$, $p(i)$ is set to the length of the shortest path from s to i (with respect to the original length function ℓ), then, for each arc $(i,j) \in A$, $\bar{\ell}(i,j) \geq 0$.

Proof. First observe that for each arc $(i,j) \in A$, the length of the shortest path from s to j is at most the length of the shortest path from s to i plus $\ell(i,j)$ (since we can build a path from s to j by first going to i and then taking arc (i,j). By the way that we set the p values, this means that $p(j) \leq p(i) + \ell(i,j)$, and hence $p(i) + \ell(i,j) - p(j) \geq 0$. But then, $\bar{\ell}(i,j) = p(i) + \ell(i,j) - p(j) \geq 0$. □

Claim 2. If, for each $i \in N$, $p(i)$ is set to the length of the shortest path from s to i (with respect to the original length function ℓ), then, for each node $v \in N$, the total adjusted length of the shortest path from s to v is 0.

Proof. Recall that this shortest path is shortest with respect to both ℓ and $\bar{\ell}$. We know that the total adjusted length of *any* path from s to v is its total length with respect to the original lengths ℓ plus $(p(s) - p(v))$. But $p(s) = 0$ and $p(v)$ is the length of the shortest path from s to v (with respect to the original lengths ℓ). So the total adjusted length of any shortest path is 0. □

Claim 3. If, for each $i \in N$, $p(i)$ is set to the length of the shortest path from s to i (with respect to the original length function ℓ), then, for each arc (i,j) in a shortest path from s to v, $\bar{\ell}(i,j) = 0$.

Proof. Claim 1 showed that each adjusted length is non-negative. Claim 2 showed that the total adjusted length of any shortest path is equal to 0. But the only way these two can both happen is that that *every* arc in a shortest path must have adjusted length equal to 0. □

These claims have the following nice consequences. Suppose that you run Dijkstra's algorithm. Now, if you compute $\bar{\ell}$ where each $p(i)$ is the shortest path length from s to i, we get an equivalent input in which each arc has adjusted length that is non-negative, and each arc in a shortest path has adjusted length 0. But then by the original simple case that we discussed at the start (as in Figure 4) we know that we have the shortest path with respect to the adjusted lengths, and thus have the shortest path with respect to the original ones.

To summarize: we can check if our path from s to w is indeed shortest by (1) computing $\bar{\ell}(i,j)$ for each arc in the graph for p values set by the shortest path lengths just found by Dijkstra's algorithm; (2) for each $(i,j) \in A$ check that $\bar{\ell}(i,j) \geq 0$; (3) for each arc (i,j) in the path, check that $\bar{\ell}(i,j) = 0$. If this holds, then you have computed a correct shortest path.

Now return to Figure 5. This figure indicates another setting for the p values. How does this procedure prove that these values are *not* the shortest path values? If we look at Figure 6, we see that there is no path from node 1 to node 6 of total adjusted length equal to 0. If the p values did indicate the shortest path lengths, then there must be such a path. Hence, these are not the correct values.