# Optimization Theory (DS2) HW#5
# Shortest Path from Dijkstra to OSPF

June 7, 2017

**Abstract**

Is it true that OSPF and Dijkstra give the same result?

## 1 Introduction

In class, I talked about two forms of Dijkstra's algorithm:

- The source-destination form, which starts from a source and searches the graph until it finds the named destination, then quits.
- The all-destinations form, which builds a spanning tree that can be used to identify the shortest path to all possible destinations.

In this homework, we will look at the latter form and examine a specific set of paths. We'll use the same graph we used during class, in Fig. 1.

The Internet's Open Shortest Path First protocol (see RFCs 2328 and 5340), which I briefly described, uses Dijkstra's algorithm. Each node collects information about the set of nodes and links, and the link costs, then *independently* calculates its own shortest path spanning tree. The Internet only works if the set of spanning trees calculated by the nodes are consistent enough to guarantee that packets don't get stuck in a loop somewhere. In this homework, we are examining that proposition.

## 2 Problems

1. First, for single paths:
   (a) Identify the shortest path from A to E on the graph. Although the edges in the original graph are undirected, this path will be directed.
   *(The answer will be a list of edges.)*
   (b) For each node in that path other than A and E, find the shortest (directed) path to E. (e.g., if the A-E path is four hops, you will have three other paths to E, one starting at each of the other nodes.) Is the set of edges in all of

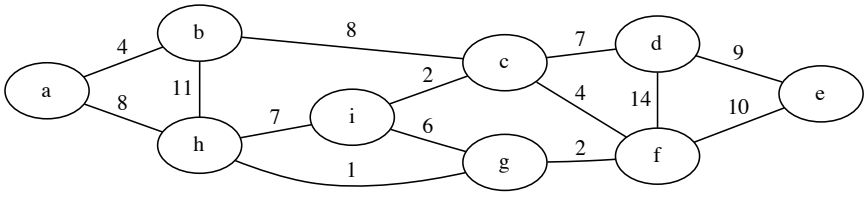


Figure 1: Our nine-vertex, fourteen-edge figure from Intro to Algorithms.

those paths a subset of the edges of the original A-E path? Are all of the edges oriented in the same direction?
*(The answer will be several lists of edges, plus a qualitative answer to each question.)*

2. Next, for trees:

   (a) Find the shortest path spanning tree (*not* the minimum cost spanning tree! Remember, they are (or can be) different.) for node A, including paths to all of the other nodes. Again, the edges should be directed.
   *(The answer will be a set of edges that make a spanning tree.)*

   (b) Select a second node. Find the shortest path spanning tree for it.
   *(The answer will be a set of edges that make a spanning tree, rooted at a different node.)*

   (c) Compare the two trees: Which portions of the trees are identical, which different?
   *(The answer will be a discussion of the results.)*

   (d) Construct a "Next Hop" table for each of the two nodes. Describe any similarities or differences you see.
   *(The answer will be a set of Next Hop tables; see the additional notes at the end of this document.)*

3. If the shortest path from A to some other node Z passes through nodes X and then Y, can it ever be the case that the shortest path from X to Z does *not* pass through Y? Why or why not?
   *(The answer will be a qualitative argument.)*

# 3 Additional Notes

Last year, there was some confusion over some important aspects of the homework, so here are some additional notes on what is expected.

Directed v. Unidirected and Next Hop Table:

There has been some confusion over the next hop table and HW5. First, directed v. undirected graph: the basic network in all of our examples and homeworks begins as an undirected graph -- you can move in either direction across any link. However, a path from, say, A to F is inherently directed. It might go A->B->C->D->E->F, for example. After you run Dijkstra's algorithm, usually people don't bother to mention the direction you would cross a link, since it's obvious from the tree structure.

Understanding the homework, though, depends somewhat on that directionality. If you build a shortest-path tree from A to F, in this case, we expect that it will eventually pass through D. How does it get from D to F? There are two basic ways: (1) Follow a path calculated entirely by the source (A), with that information communicated to the other nodes in some fashion. (2) Allow each node to *independently* decide how to get the packet from itself to the specified destination. In order for this to work, the individual nodes must make *consistent* decisions.

The packet will pass through D. At D, it will use the shortest-path tree that D calculated in order to figure out to get from D to F. So, how do we guarantee that D's decision is consistent with A's? How do we guarantee that the packet won't end up back at C instead of going on to E, like it should?

We can check that by checking that the shortest-path spanning tree calculated at D is consistent with that calculated at A, B and C for some set of destinations. You will want to check that it's consistent for direction of the link, as well as the next hop.

Okay, speaking of the Next Hop, that's the table that each node uses to decide where to go next. Completely making up an example, it might look like:

Node A's Next Hop Table:

| destination | next hop |
|---|---|
| B | B |
| C | C |
| D | B |
| E | B |
| F | C |

That is, A is directly connected to B and C, so it's only one hop to them, but in order to get to D, it has to first send the packet to B.

So the homework is to calculate that next hop table for each of the nodes in our network, and compare them to figure out if they are consistent, will correctly route packets between all pairs of sources and destinations.

For homework 1a, the answer will be a list of edges from the original graph. For 1b, it will be several lists of edges, plus a qualitative answer to the questions.

For 2a, it will be a set of edges that make a spanning tree. For 2b, it will be a set of edges making a spanning tree rooted at a different node. 2c is discussion.

2d will be a Next Hop table like the one above, for *each* node.

3, the answer is qualitative.