CE 252A **Solutions for Homework 2**

1. *In class, we saw that "pipelined" protocols, also known as "sliding window" protocols, are a type of ARQ (Automatic Repeat Request) protocols that use a "window" to control the amount of data they inject into the network.*

   (a) *What is the window size in stop-and-wait ARQ? How many unique sequence numbers does Stop-and-Wait need? How many bits are needed to represent Stop-and-Wait's unique sequence numbers? Explain.*

   The "window size" is the number of packets that a station will send without receiving an acknowledgement. In stop-and-wait ARQ, the window size is 1; so the transmitter will wait until it sees an ACK before transmitting the next packet.

   Looking at the scenario where an ACK is lost shows us the number of unique sequence values needed. Since a dropped ACK will cause a repeated transmission, two sequence values are needed for the receiver to differentiate between a repeated packet (caused by a dropped ACK) and a new packet. Two sequence numbers can be represented with a single-bit binary number.

   (b) *What is the main advantage of requiring a smaller number of bits to represent the range of unique sequence numbers employed by a protocol?*

   Fewer sequence-number bits simply means less overhead.

   (c) *For a 100Mbits/sec channel with 100ms propagation delay, what is the channel utilization when sending 2KByte frames if Stop-and-Wait is used?*

   Correction: The original solution said to compute $\frac{T_{\mathbf{with}}}{T_{\mathbf{without}}}$. That should have been $\frac{T_{\mathbf{without}}}{T_{\mathbf{with}}}$. The corrected version is below.

   To compute channel utilization we compare the time that is needed to transmit (and acknowledge) a single packet using the protocol and the time that is needed to transmit a single packet without using the protocol: $\frac{T_{\text{without}}}{T_{\text{with}}}$.

   Without the protocol, at 100 Mbits/sec, the time in seconds to transmit a 16,000 bit packet is $T_{\text{without}} = \frac{\text{16,000 bits}}{\text{100,000,000}\frac{\text{bits}}{\text{sec}}} = 0.00016$ sec. We can ignore the propagation time because there is no requirement to wait before transmitting the next packet.

   With the protocol, it takes 0.00016 sec to transmit the packet, 100 msec for the packet to travel to the receiver, and another 100 msec for the ACK to return to the transmitter: $T_{\text{with}} = 0.00016 \text{ sec} + 0.1 \text{ sec} + 0.1 \text{ sec} = 0.20016 \text{ sec}$.

   Then the utilization of the channel is $\frac{T_{\text{without}}}{T_{\text{with}}} = \frac{0.00016 \text{ sec}}{0.20016 \text{ sec}} \approx 0.0008$.

(d) *How can you increase channel utilization 10-fold?*

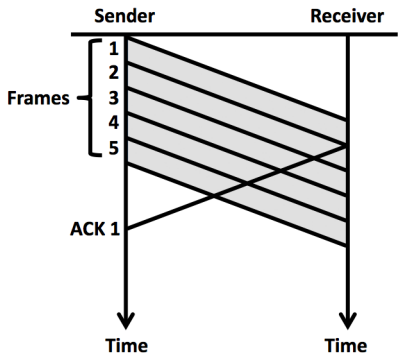
Since the packet time is dominated by the propagation delay, we can increase the utilization by increasing the window size to 10, essentially sending 10 packets during the time that it takes for the first ACK to arrive.

(e) *Describe the additional complexity of your solution compared to Stop-and-Wait.*

A larger sequence number will be needed. More memory will be needed for the larger windows.

2. *Suppose that a sender and a receiver are using ARQ to perform reliable data delivery.*

(a) *In a Go-Back-N ARQ protocol, the window size is 6. Frames with sequence numbers 1, 2, 3, 4 and 5 have been sent. The sender just received an ACK for frame 1. Frames 6, 7, 8, 9 and 10 are waiting to be sent. Draw the time diagram showing this scenario.*



(b) *Which frame(s) can the sender send before it must wait for the next ACK from the receiver? Explain.*

The sender can have six frames unacknowledged after the ACK 1: that is, frames 2, 3, 4, 5, 6, and 7. So after seeing an ACK for frame 1, the sender can send frames 6 and 7.

(c) *Some time later, the sender transmitted frames 20, 21, 22, 23, 24, and 26; however, frame 22 got lost. If Go-Back-N is used, what frame(s) would the sender have to retransmit? Explain.*

With Go-Back-N, the receiver has only one buffer (that is, it has a window size of 1). So the receiver will ACK the last in-sequence packet that it received (21), and the sender must retransmit all the frames that follow it.

(d) *Suppose the same situation as above but sender and receiver use Selective-Repeat ARQ. What frame(s) would the sender need to retransmit? Explain.*

With Selective-Repeat ARQ, the receiver has several buffers and can receive frames out of sequence. The receiver would acknowledge frames 20, 21, 23, 24, and 26 (but not 22 because it was lost). So the sender would need to resend 22.

(e) *Can Selective-Repeat ARQ use cumulative ACKs? Explain.*

Yes. A cumulative ACK indicates that "all frames up to n have been received." And so if a receiver has received a contiguous block of frames ending with frame n, it can ACK them cumulatively.

(f) *What are the trade-offs between Go-Back-N ARQ and Selective-Repeat ARQ?*

Go-Back-N ARQ has a receive window of 1, and so it requires less memory for buffers in the receiver. However since the receiver will not store frames that follow a lost frame, using Go-Back-N ARQ may cause retransmissions of otherwise properly received frames, wasting bandwidth. Selective-Repeat ARQ requires more buffer space in the receiver, and since it can store frames after a lost frame, it will not request unnecessary retransmissions and will not waste bandwidth.

3. *Link-state routing requires routing updates to be flooded to all participating routers. Besides the actual routing update, a link-state packet carries the node id, a sequence number, and a time-to-live. Why do you think this extra information (overhead) is necessary?*

The node id helps prevent forwarding of duplicate updates, the sequence number helps identify old data that should be ignored, and the time-to-live field limits the extent of flooding.

4. *Why are datagrams said to be "self-contained"?*

Datagrams include all of the information needed for them to be routed to their desired destinations. They do not rely on information from prior packets as packets of virtual circuits do.

(The definition of "self-contained" is "Constituting a complete and independent unit in and of itself. Not dependent on others."

See `http://www.thefreedictionary.com/self-contained`)

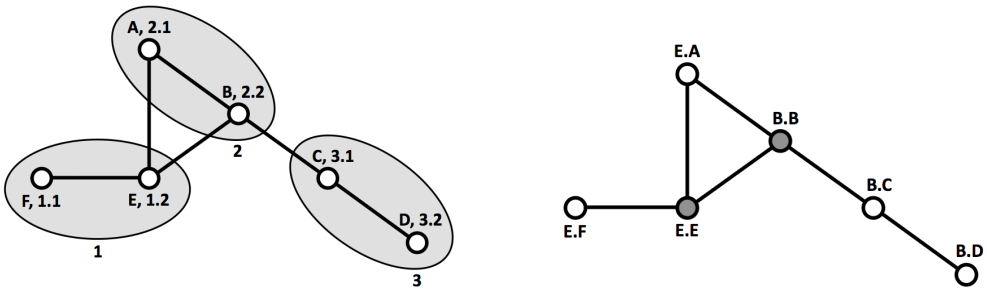


Figure 1: Area Hierarchy, and Landmark Hierarchy with $r_0 = 1$, $r_1 = 2$.

5. *Given the Figure 1 hierarchies on the same topology, compare (quantitatively):*

   (a) *Path lengths.*

   Normally, full routing tables can specify optimal routes for each destination, but with **Area Routing**, network details of other areas are invisible, and so optimality may be sacrificed. For instance, the routing table for router 1.2 might specify that all traffic that is intended for area 2 should go to router 2.2, which is ideal for destinations in area 3, but then traffic that is intended for router 2.1 will take an extra hop.

   With **Landmark Hierarchy Routing**, each router's address is a sequence of landmarks of decreasing radii. Distant sources will be aware of just the largest-radii landmarks and initially will send packets on the shortest path to those landmarks, even if that path is not the shortest path overall. (In the assigned problem, all paths that use Landmark Hierarchy routing happen to be ideal.)

   (b) *Routing table sizes.*

   Area Routing reduces routing-table sizes by representing all routers of an area with a single table entry. In the six-router example, normally each routing table would have five entries. For instance, router 2.1 would have entries for routers 1.1, 1.2, 2.2, 3.1, and 3.2. Using Area Routing, routers 1.1 and 1.2 are represented by an entry for area 1, and routers 3.1 and 3.2 are represented by an entry for area 3. Consequently, each table has only three entries.

   With **Landmark Hierarchy Routing**, each routing table contains entries for nearby landmarks, where "nearby" is determined by the r_i of each router. Routing tables of the example will have $r_0 = 1$ neighbors (that is, adjacent routers) and $r_1 = 2$ neighbors. Each routing table will have two or three entries, which is fewer than the normal five entries:

   Table of A: **B**, **E**
   Table of B: A, C, **E**

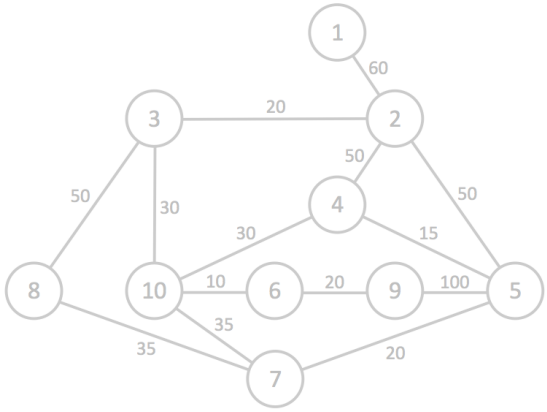
Table of C: **B**, D, **E**
Table of D: **B**, C
Table of E: A, B, F
Table of F: **B**, **E**

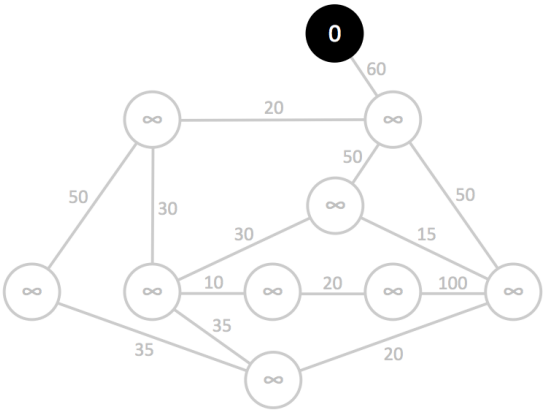
6. *For the adjacency matrix below, use link-state routing to find the shortest-path between nodes 1 and 9. Show each step of the algorithm using the topology graph. Assume nodes already have the latest topology snapshot which is given by the adjacency matrix below.*

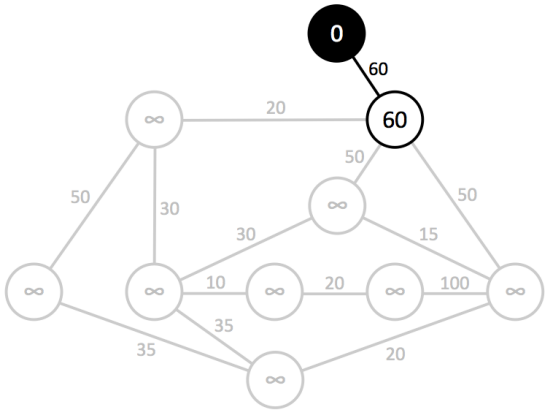
Node	**Node**	**Distance**
1	2	60
2	3	20
2	4	50
2	5	50
3	8	50
3	10	30
4	5	15
4	10	30
5	7	20
5	9	100
6	9	20
6	10	10
7	8	35
7	10	35

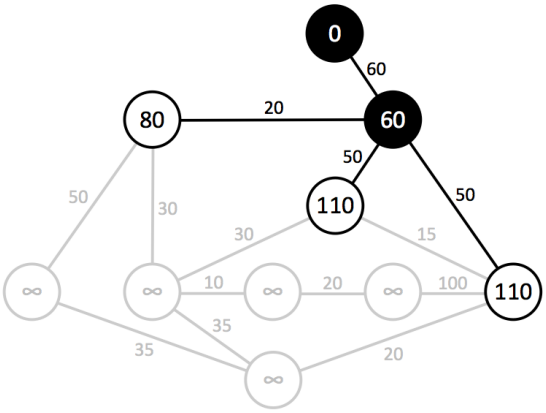
The figures on the next pages show the steps of the algorithm. Nodes are marked with their current shortest distance to node 1. At each step, dark nodes are known to have the overall shortest distance to Node 1. The shortest path is indicated by the dark edges.

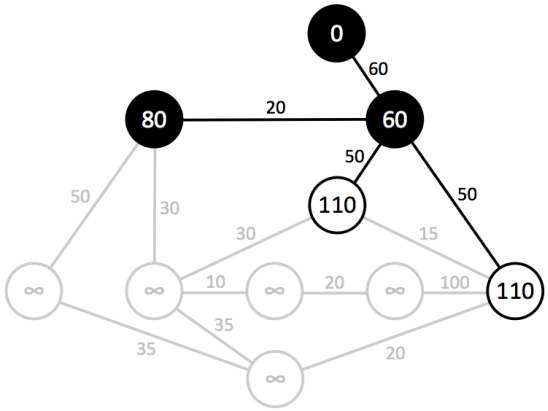
When the algorithm terminates, the shortest path from Node 1 to Node 9 is 1—2—3—10—6—9.

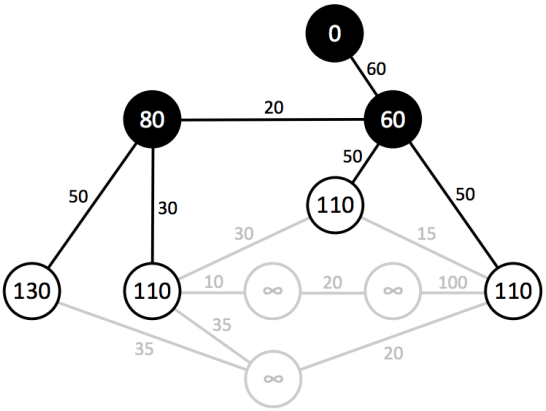

1
60
3
20
2
50
50
30
4
50
30
15
8
10
10
6
20
9
100
5
35
35
7
20

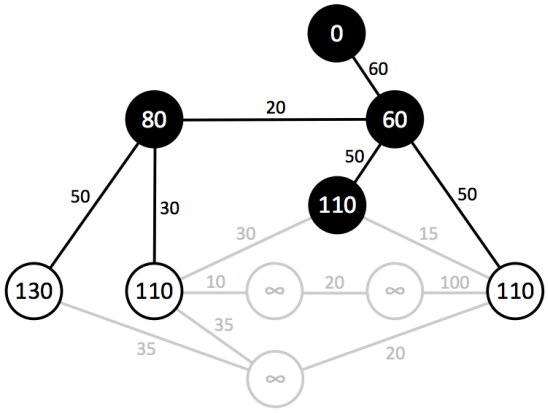

0
60
∞
20
∞
50
50
30
∞
50
30
15
∞
∞
10
∞
20
∞
100
∞
35
35
∞
20

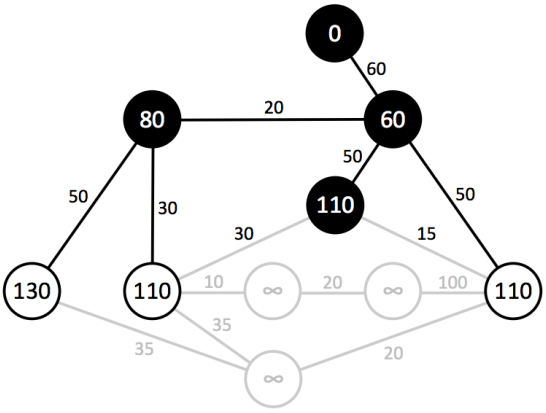

0
60
∞
20
60
50
50
30
∞
50
30
15
∞
∞
10
∞
20
∞
100
∞
35
35
∞
20

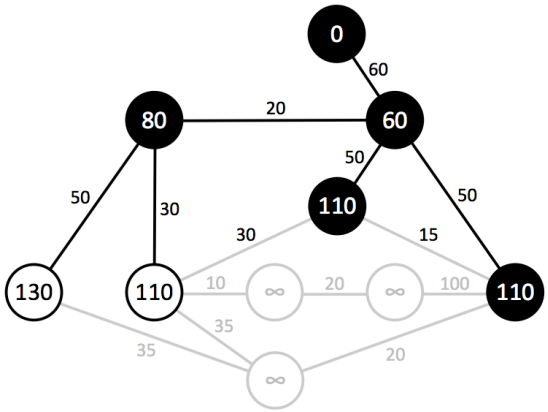

0
60
80
20
60
50
50
30
110
50
30
15
∞
∞
10
∞
20
∞
100
110
35
35
∞
20

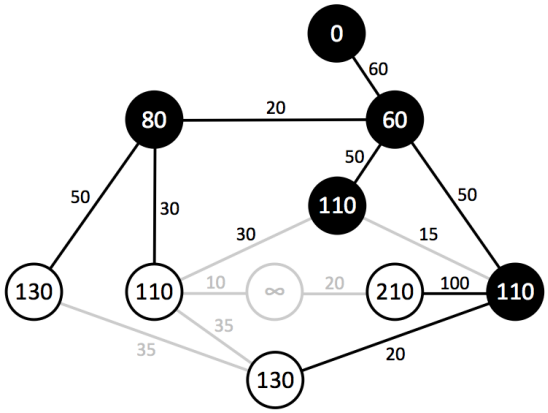

0
60
80
20
60
50
50
30
110
50
30
15
∞
∞
10
∞
20
∞
100
110
35
35
∞
20

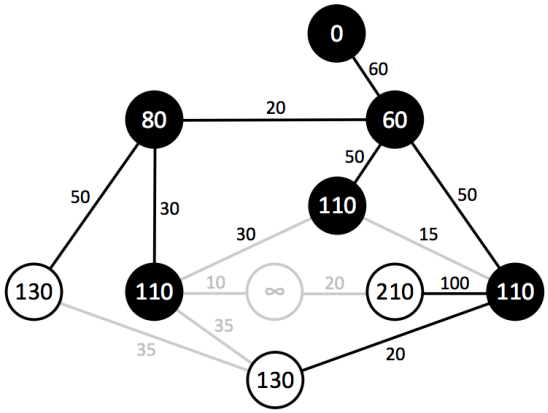

0
60
80
20
60
50
50
30
110
50
30
15
130
110
10
∞
20
∞
100
110
35
35
∞
20

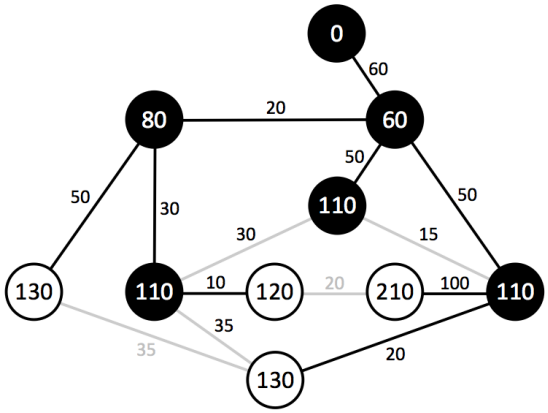

0
60
80
20
60
50
50
30
110
50
30
15
130
110
10
∞
20
∞
100
110
35
35
∞
20

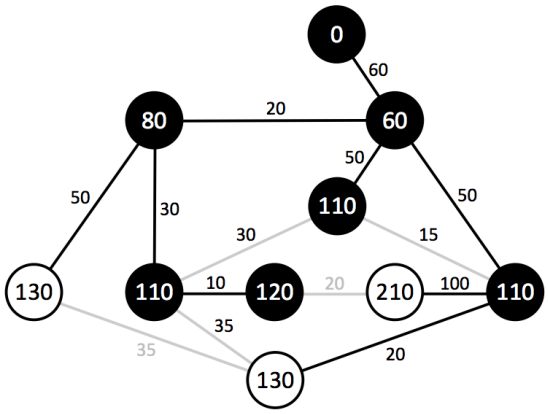

0
60
80
20
60
50
50
30
110
50
30
15
130
110
10
∞
20
∞
100
110
35
35
∞
20

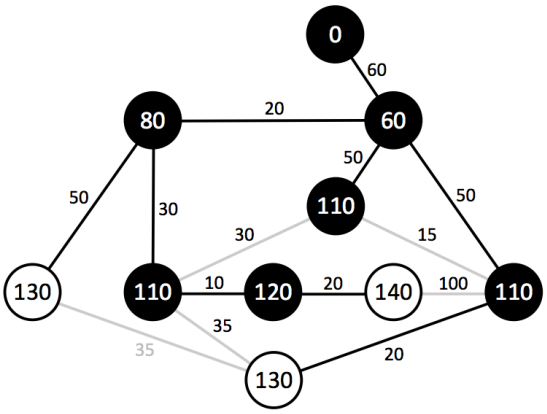

0
60
80
20
60
50
110
50
50
30
30
15
130
110
10
∞
20
∞
100
110
35
35
20
∞

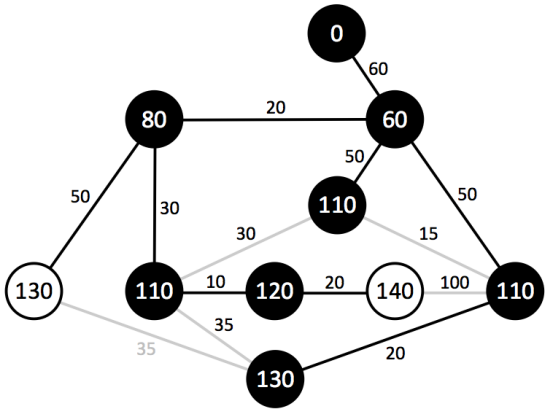

0
60
80
20
60
50
110
50
50
30
30
15
130
110
10
∞
20
210
100
110
35
35
20
130

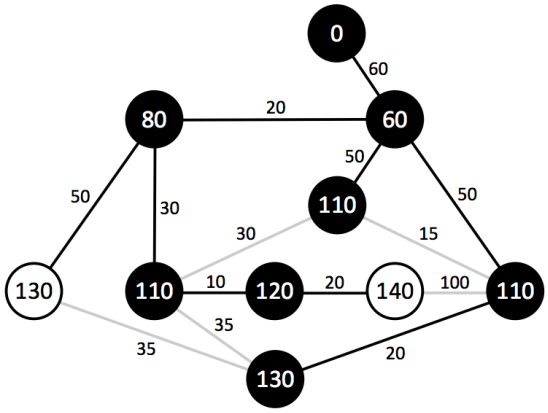

0
60
80
20
60
50
110
50
50
30
30
15
130
110
10
∞
20
210
100
110
35
35
20
130

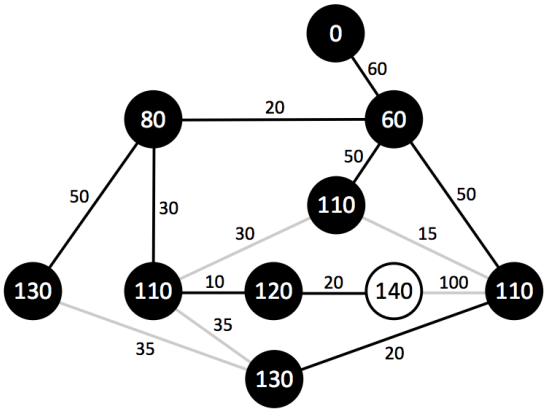

0
60
80
20
60
50
110
50
50
30
30
15
130
110
10
120
20
210
100
110
35
35
20
130

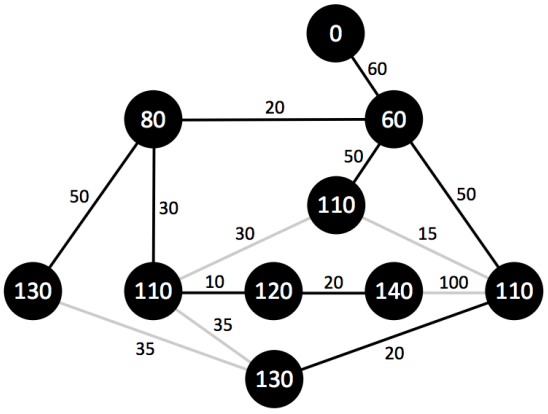

0
60
80
20
60
50
110
50
50
30
30
15
130
110
10
120
20
210
100
110
35
35
20
130

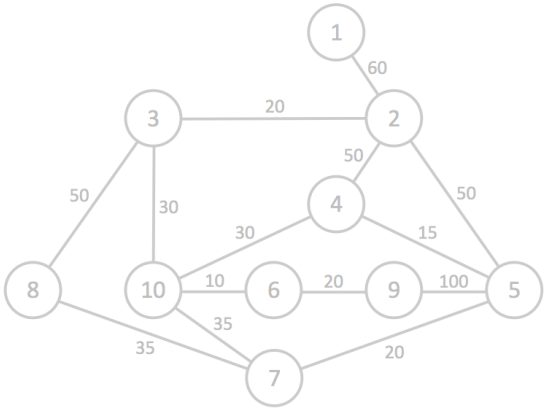

0
60
80
20
60
50
110
50
50
30
30
15
130
110
10
120
20
140
100
110
35
35
20
130


0
60
80
20
60
50
110
50
50
30
30
15
130
110
10
120
20
140
100
110
35
35
20
130


0
60
80
20
60
50
110
50
50
30
30
15
130
110
10
120
20
140
100
110
35
35
20
130


0
60
80
20
60
50
50
110
50
30
30
15
130
110
10
120
20
140
100
110
35
35
130
20


0
60
80
20
60
50
50
110
50
30
30
15
130
110
10
120
20
140
100
110
35
35
130
20


1
60
3
20
2
50
50
4
50
30
30
15
8
10
10
6
20
9
100
5
35
35
7
20