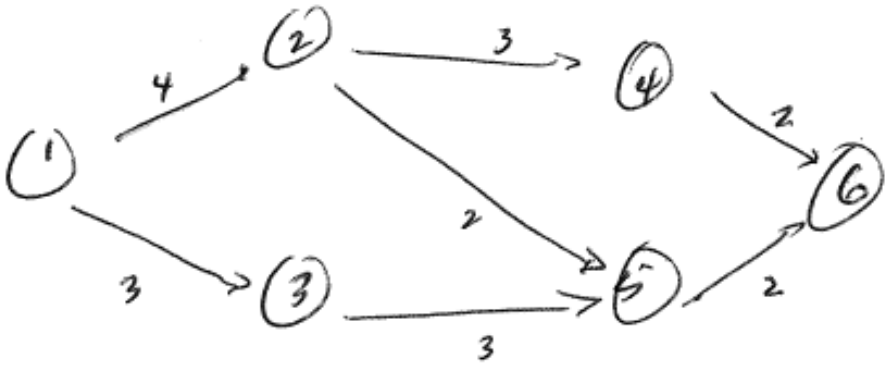
The Shortest Path (Dijkstra's Algorithm)

Last time we defined the PowerCo problem, and today we'll solve it using an algorithm developed by Edgar Dijkstra in 1959.

The nice thing about this algorithm is that not only do we find the shortest path from our initial node (source) to our terminal node (sink) in our graph, but we actually find the shortest path between the source and every other node in between.

Here's the graph, and below that we'll begin by listing our nodes and the source.

Side Remark: Normally I think our author does a fairly good job explaining algorithms, but I think he makes it overly complicated here. You can use the book's notation if you like, but I think our notation here will be a lot more straightforward.



The algorithm starts by listing the nodes:

	1	2	3	4	5	6
1	$\boxed{0}$	4_1	3_1	∞	∞	∞

A couple of notes here:

- The subscript is being used to identify which node we started with.
- ∞ is used to denote that the nodes are not connected.
- Once a boxed number appears in a column, we no longer put anything in that column.

Now, we see that 3 is the smallest unboxed distance, so it comes down and is boxed. Because we will now be coming from node 3, the "3" is placed at the front.

	1	2	3	4	5	6
1	$\boxed{0}$	4_1	3_1	∞	∞	∞
3			$\boxed{3_1}$			

Nothing goes under the boxed zero (that column is finished). Under column with node 2, normally we would put ∞, but because 4_1 is already there, we just bring it down. The other

columns work the way you think they should- the distance from node 3 to node 5 is 3, but we already have used 3 units of distance, so we write $3 + 3 = 6$ (with a subscript of 3):

	1	2	3	4	5	6
1	$\boxed{0}$	4_1	3_1	∞	∞	∞
3		4_1	$\boxed{3_1}$	∞	6_3	∞

We repeat the process. The next smallest unboxed number is the 4, so bring it down and it becomes boxed. Put "2" in the front, and fill in the rest. We might note that the distance from node 2 to node 5 is 2 units, with 4 units already used, so the distance altogether is 6. Six is also the distance used using node 3, so there is a tie. We could keep 6_3, or write 6_2- your choice.

	1	2	3	4	5	6
1	$\boxed{0}$	4_1	3_1	∞	∞	∞
3		4_1	$\boxed{3_1}$	∞	6_3	∞
2		$\boxed{4_1}$		7_2	6_2	∞

Now bring down 6_2 and repeat:

	1	2	3	4	5	6
1	$\boxed{0}$	4_1	3_1	∞	∞	∞
3		4_1	$\boxed{3_1}$	∞	6_3	∞
2		$\boxed{4_1}$		7_2	6_2	∞
5				7_2	$\boxed{6_2}$	8_5

Repeat the process a couple of times, and we get the finished table:

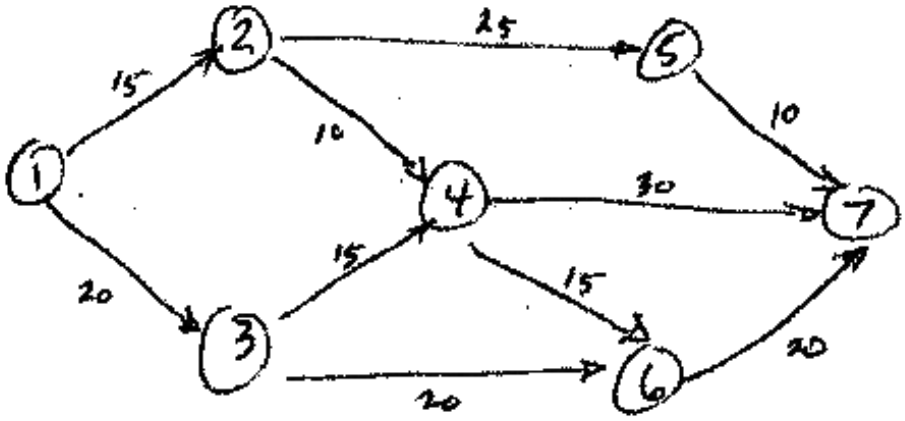
	1	2	3	4	5	6
1	$\boxed{0}$	4_1	3_1	∞	∞	∞
3		4_1	$\boxed{3_1}$	∞	6_3	∞
2		$\boxed{4_1}$		7_2	6_2	∞
5				7_2	$\boxed{6_2}$	8_5
4				$\boxed{7_2}$		8_5
6						$\boxed{8_5}$

To interpret this solution, we see go down the column for node 6, and see that the shortest distance to node 6 came from node 5 and had a value of 8.

To backtrack the path, go over to column 5, and go down to see that you could use either 6 from node 3 or 6 from node 2. Using node 3 first, we came from node 1 (so the path was 1-3-5-6 with a value of 8), or using node 2, the path would be (1-2-5-6, also with a value of 8).

You also note that the boxed value is the shortest path from the source to that node, which may come in handy sometime.

The second example is straightforward. Copy down the graph with the weighted edges and try the algorithm out yourself, then come back to the notes to check your answer against the one below.



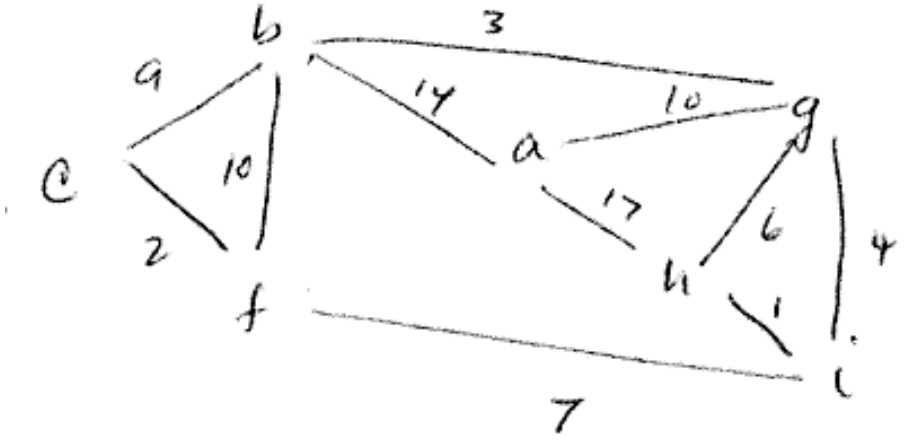
	1	2	3	4	5	6	7
1	$\boxed{0}$	15_1	20_1	∞	∞	∞	∞
2		$\boxed{15_1}$	20_1	25_2	40_2	∞	∞
3			$\boxed{20_1}$	25_2	40_2	40_3	∞
4				$\boxed{25_2}$	40_2	40_4^*	55_4
5					$\boxed{40_2}$	40_4	50_5
6						$\boxed{40_4}$	50_5
7							$\boxed{50_5}$

(*) There's a tie between 3 and 4. We used 40_4, but these could be 40_3 if you didn't want to change.

In conclusion, we had one path for the shortest path (we didn't need node 6), and it had a value of 50:

$$1 - 2 - 5 - 7$$

Lastly, we show that the source doesn't have to be furthest right. Here, the nodes are alphabetical, and we want to go from Node a and list the (shortest) distance to all other nodes. Once again, copy down the graph and see if you get the same answer that is shown below.



SOLUTION:

	a	b	c	f	g	h	i
a	$\boxed{0}$	14_a	∞	∞	10_a	17_a	∞
g		13_g	∞	∞	$\boxed{10_a}$	16_g	14_g
b		$\boxed{13_g}$	22_b	23_b		16_g	14_g
i			22_b	21_i		15_i	$\boxed{14_i}$
h			22_b	21_i		$\boxed{15_i}$	
f			22_b	$\boxed{21_i}$			
c			$\boxed{22_b}$				

Now we can easily determine the distance from node a to all other nodes. This allows us to draw something called a "minimum spanning tree". The way it goes is the following:
Put Node a at the top. Which node is closest to node a? Node g is the only node whose shortest distance comes from node a, so we connect node g to a.
What nodes are connected to g? We look at the boxed values, and see 13_g indicates node b is connected to node g, and the boxed 14_g indicates node i is connected to node g. Next, we see that node h is connected to i, and node f is connected to i (so there's a branching there). And we continue this way to construct the following interesting "tree".