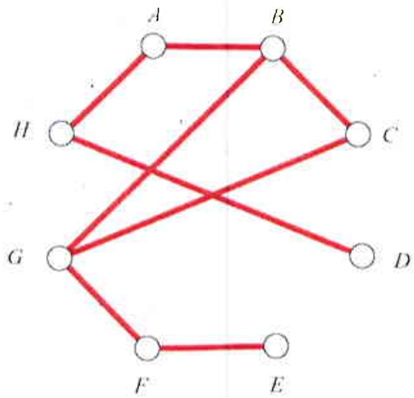
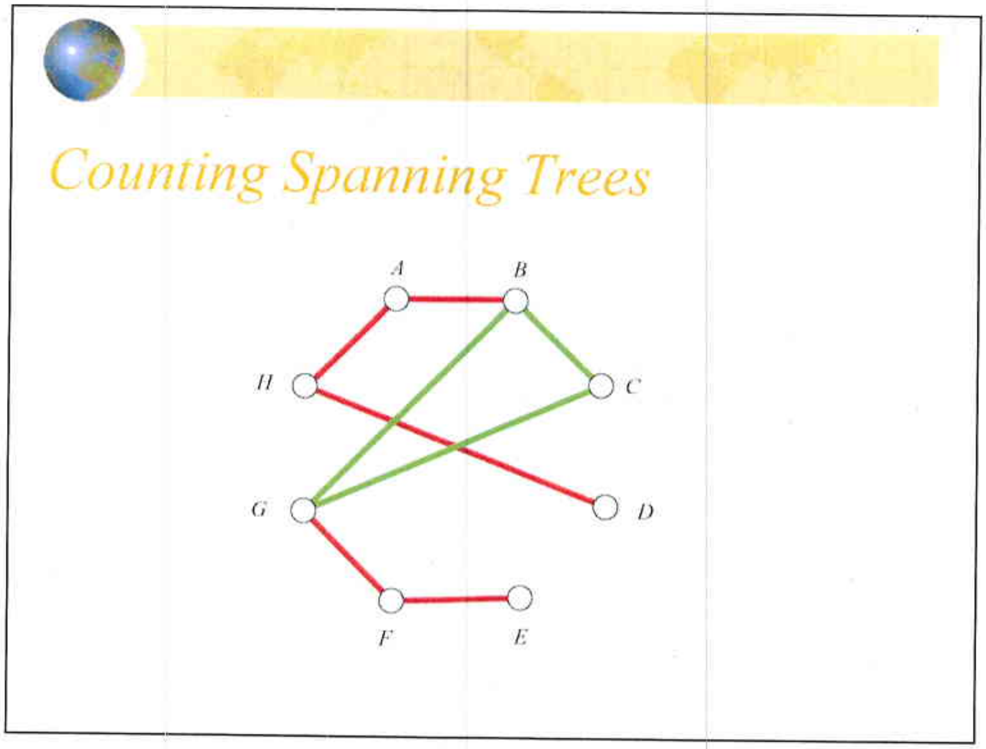
Counting Spanning Trees



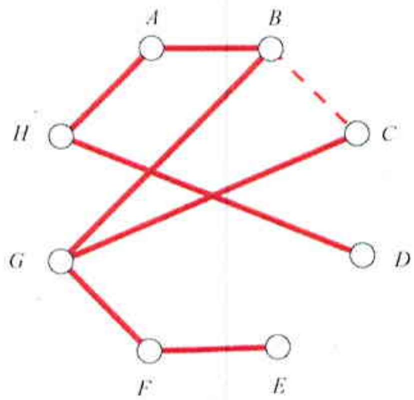
This network has N = 8 vertices and M = 8 edges, so redundancy R = 1

To find a spanning tree, we will have to discard an edge.



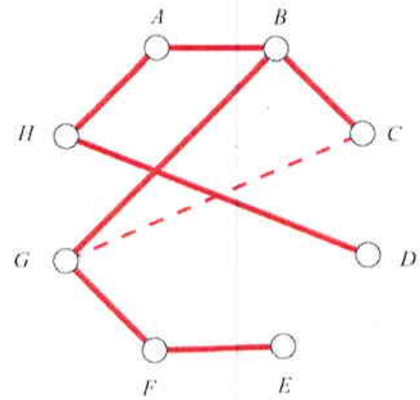
Five edges are bridges (red) and so must remain. The other three edges form a circuit of length 3, and we can discard any edge.

Counting Spanning Trees

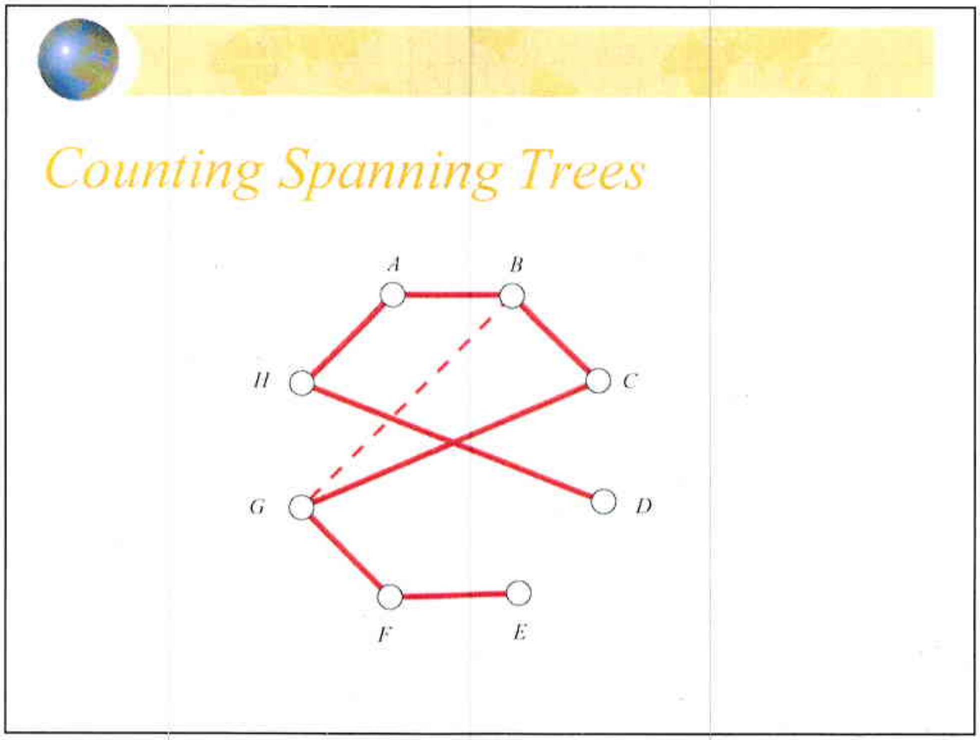


BC, or

Counting Spanning Trees



CG, or



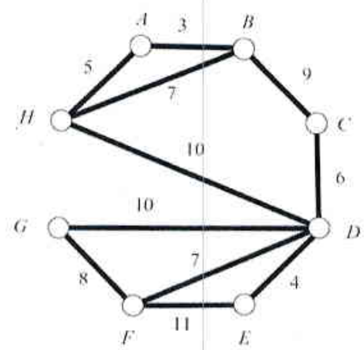
BG.

In more complicated situations, the procedure is the same – we need to bust up circuits.

⌘

Minimum Spanning Trees

- *N* = 8 vertices and *M* = 11 edges in this weighted network.



Here is a weighted network with N = 8 and M = 11, so redudancy R = 4.

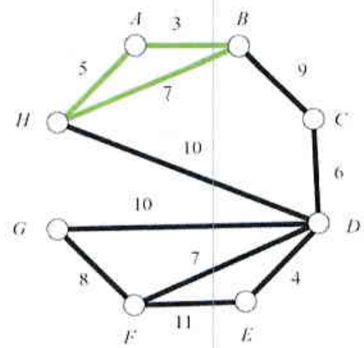
Our goal is to find the minimum spanning tree of the network.

There are many different spanning trees, and we don't want to list them all, so let's develop a strategy here…

⌘

Minimum Spanning Trees

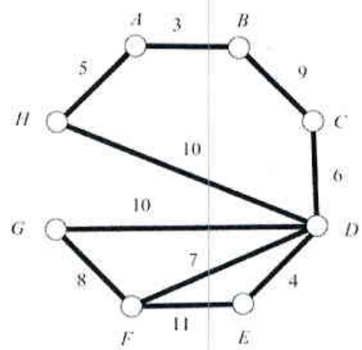
- Circuit *A,B,H,A* can be broken by removing *AB*, *BH*, or *HA*.



Here, we look at the first circuit to bust.

Minimum Spanning Trees

- Clearly, remove *BH*, the most expensive.
- Continuing with each of the circuits, we find...



Clearly, get rid of the most expensive.

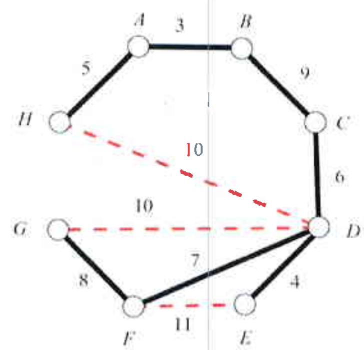
How about in ABCDHA?

DFGD?

DFED?

Minimum Spanning Trees

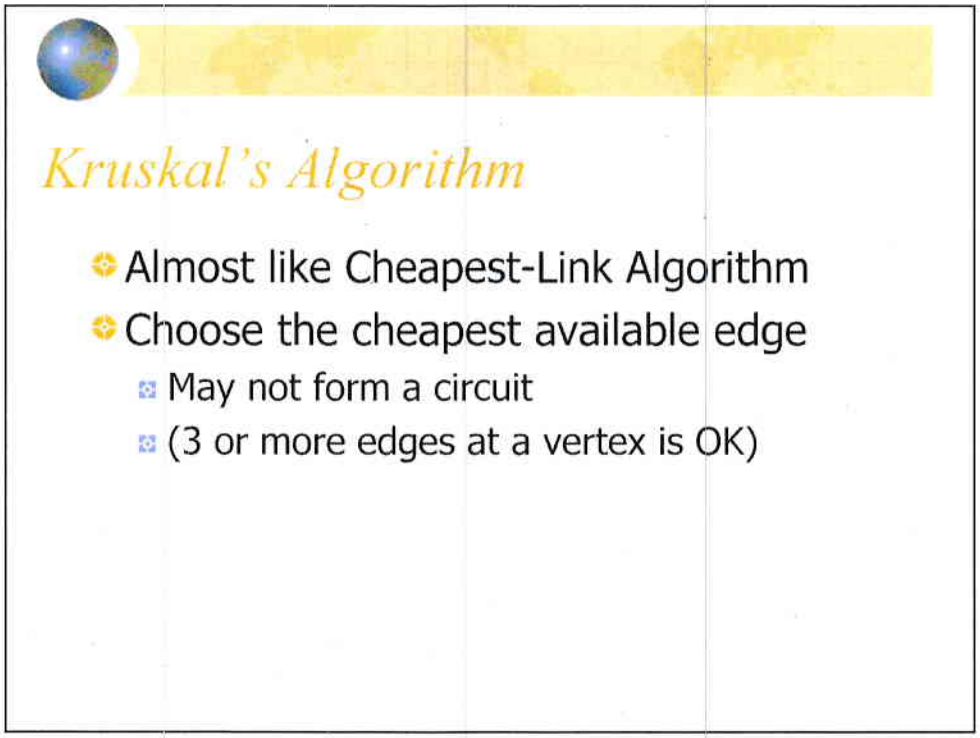
- ...the minimum spanning tree



This is the minimum spanning tree.

But the business of looking for circuits and dropping links is a little tedious. There is an easier way.

⌘



Kruskal, an American mathematician, who first proposed the algorithm in 1956.

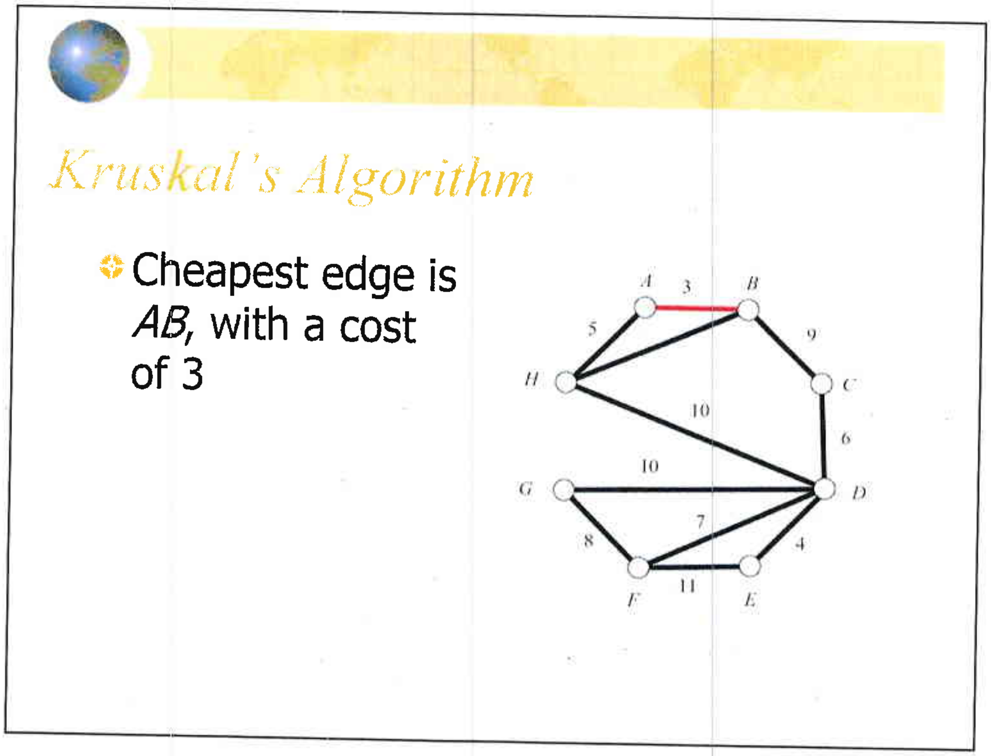
- Recall the cheapest-Link algorithm. This is a bit like that, but there are some essential difference, too.

- Like the cheapest Link algorithm, choose the cheapest link

    - forming no circuits

- But 3 or more edges at a vertex is OK.

Let's try it.

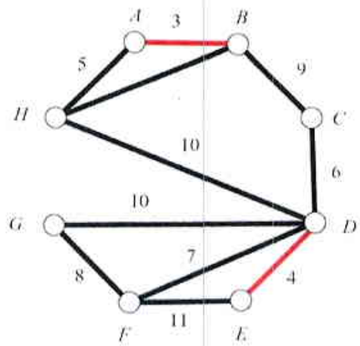


Cheapest edge is AB, with a cost of 3

What is the next edge?

Kruskal's Algorithm

- Cheapest edge is *DE*, with a cost of 4



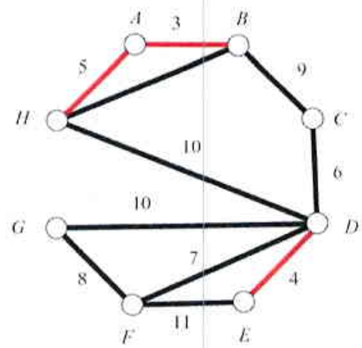
DE, with a cost of 4

And next?

⌘

Kruskal's Algorithm

- Cheapest edge is *AH*, with a cost of 5

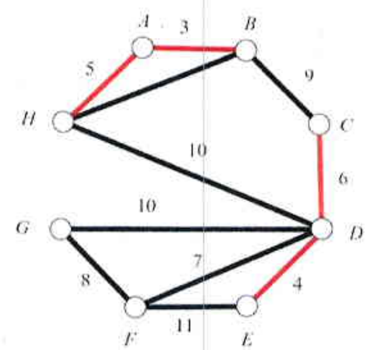


AH for 5

Next?

Kruskal's Algorithm

- Cheapest edge is *CD*, with a cost of 6

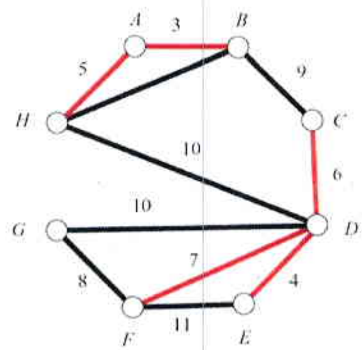


CD costing 6,

And next?

Kruskal's Algorithm

- Cheapest edge is *DF*, with a cost of 7

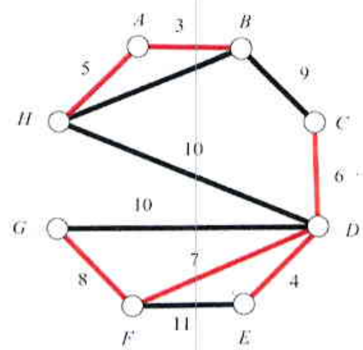


DF at 7,

Next?

Kruskal's Algorithm

- Cheapest edge is *FG*, with a cost of 8

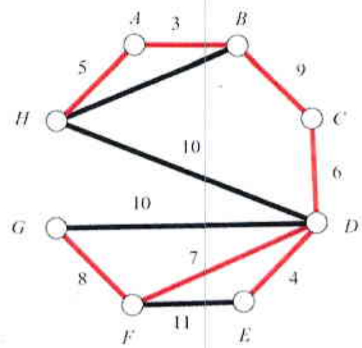


FG at 8

And next?

Kruskal's Algorithm

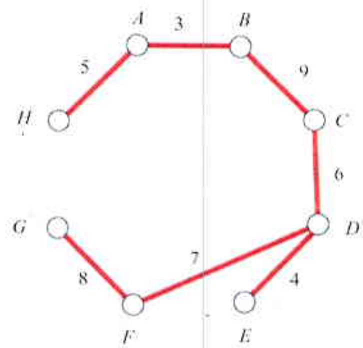
- And finally, edge *BC*, with a cost of 9



Finally, BC with a cost of 9

Kruskal's Algorithm

- So we have the minimum spanning tree, at a cost of 42.



So we have the minimum spanning tree at a cost of 42

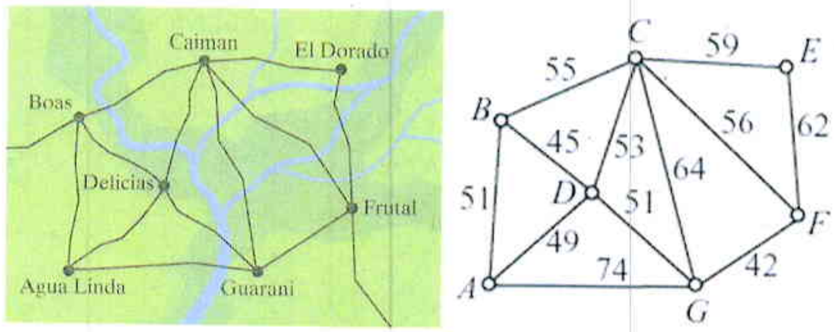
⌘

Kruskal's Algorithm

- What is truly remarkable about Kruskal, is that unlike the Cheapest-Link Algorithm,
  - It always produces an optimal solution!

What is truly remarkable about Kruskal, is that unlike the Cheapest-Link Algorithm, It always produces an optimal solution!

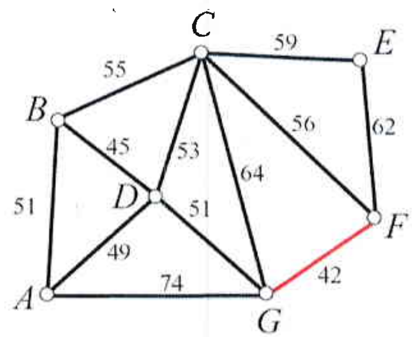
Amazonian Cable Network: ACN



Let's revisit the Amazon…

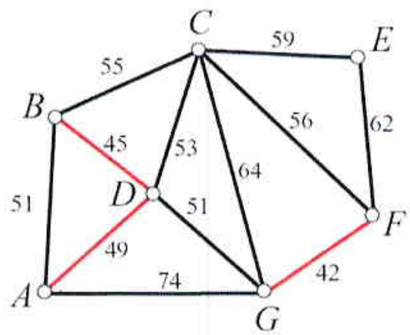
ACN and Kruskal

- We choose the cheapest link: *GF* at a cost of $42 million



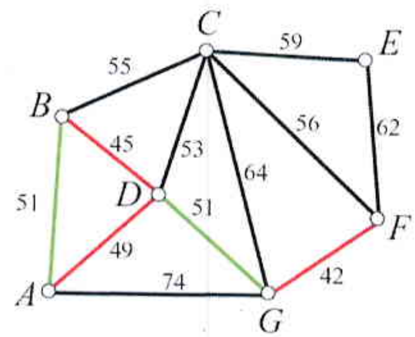
ACN and Kruskal

- Next are *BD* at $45 million and *AD* at $49 million.



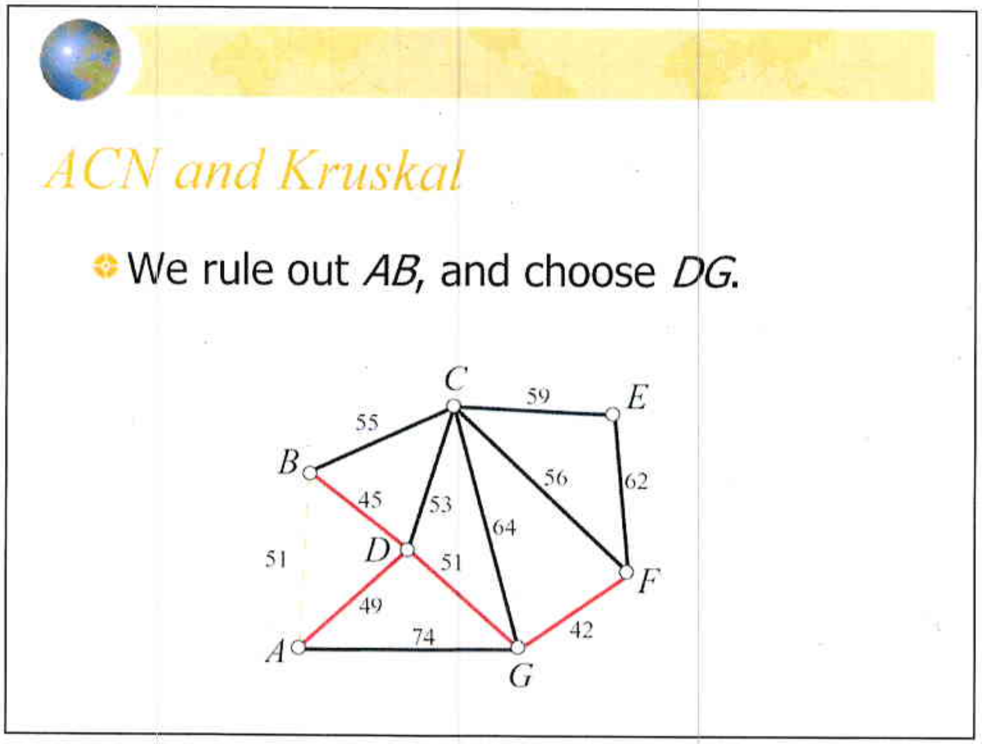
ACN and Kruskal

- Next, there is a tie for $51 million between *AB* and *DG*.



What do we do in this case?

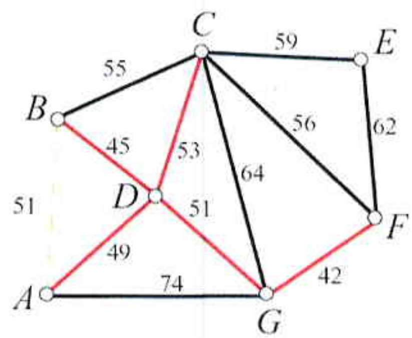
If both are OK, choose one at random. But in this case…



We rule out AB (why?) and instead choose DG.

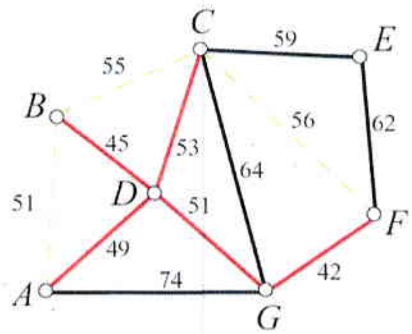
ACN and Kruskal

- Next is *CD* at $53 million.



ACN and Kruskal

- The next cheapest, *BC* ($55 million) and *CF* ($56 million) would each create a circuit.

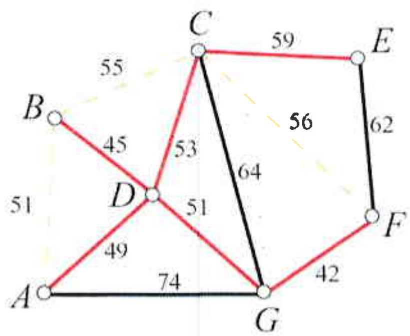


We need to rule out BC and CF,

⌘

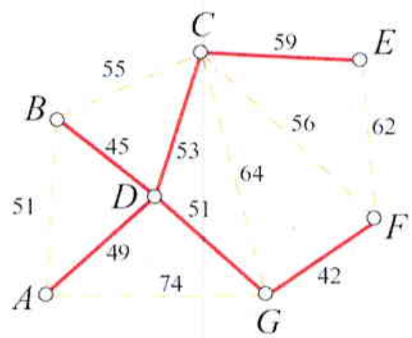
ACN and Kruskal

- Next is *CE* at $59 million.



ACN and Kruskal

- That was the 6th edge, so we are done!



Cost

$ 42m
45m
49m
51m
53m
59m

$299m

And we are done, at a cost of $299 million. We drop the remaining links.

Minimum Spanning Tree

- Kruskal's algorithm is easy to implement.
- Kruskal's algorithm is an efficient algorithm.
- Kruskal's algorithm is an optimal algorithm.

Hooray!