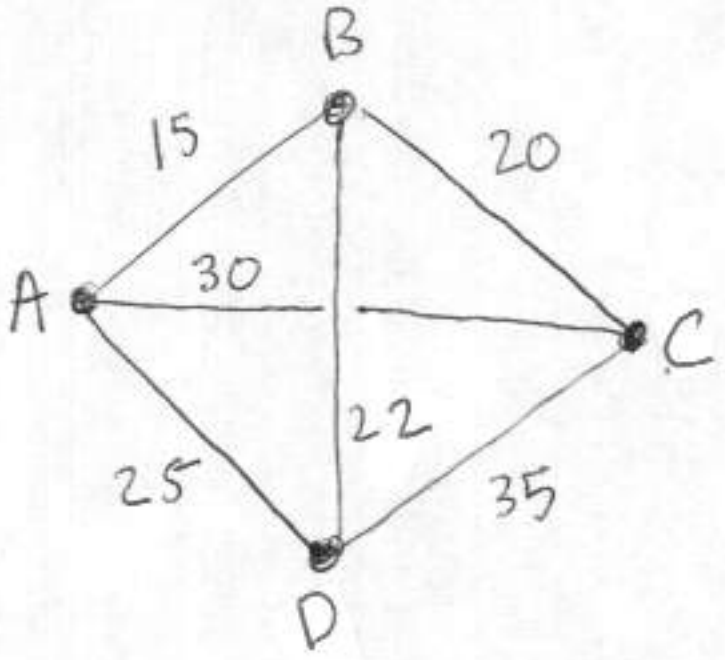
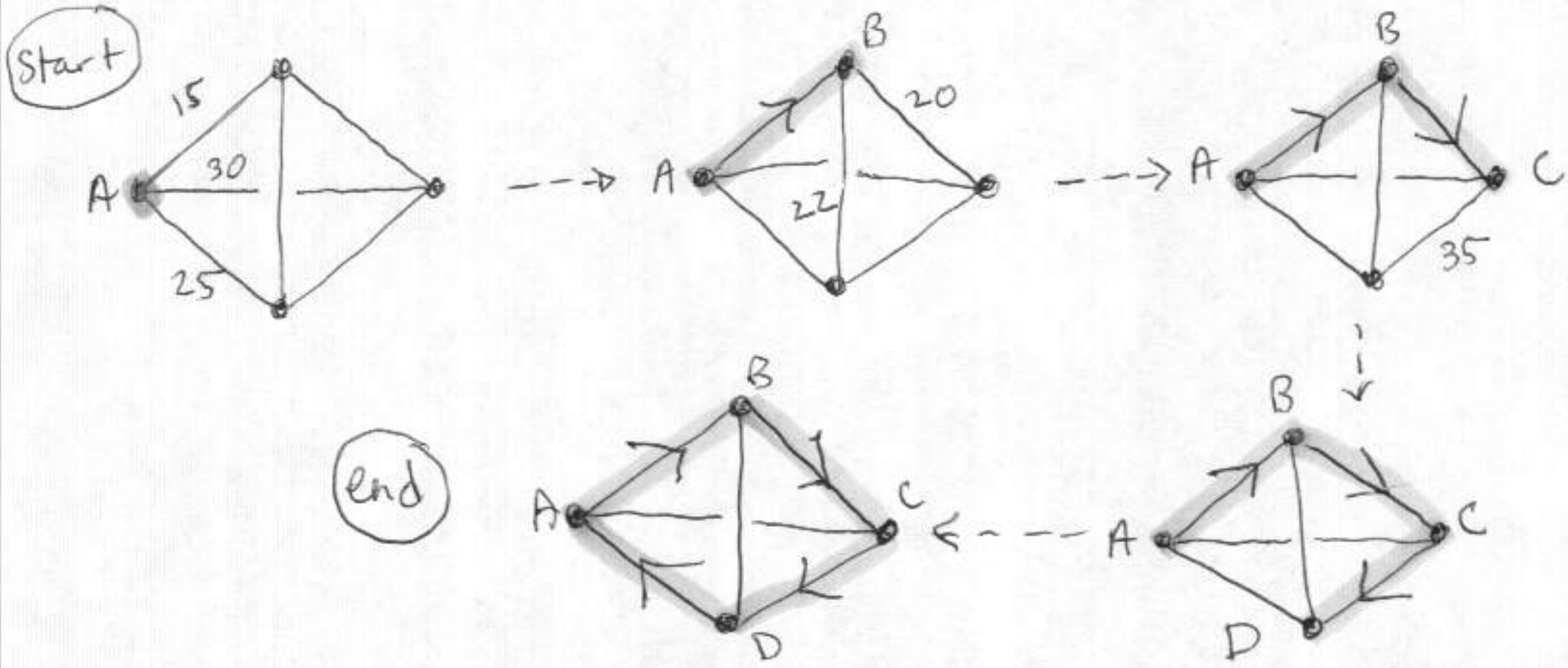
More NNA and RNNA

Suppose we are given the weighted graph



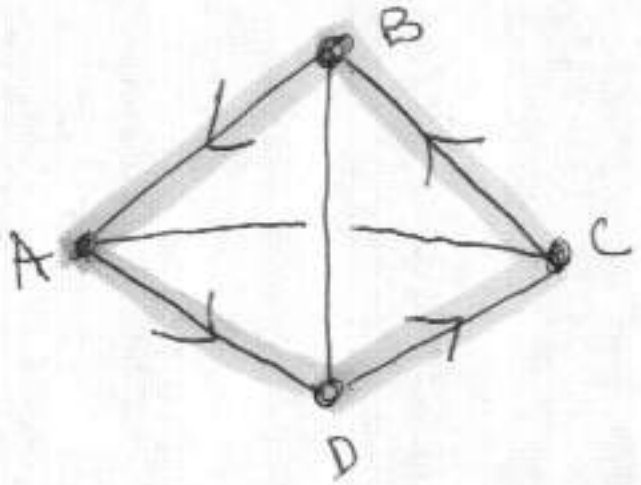
We want to apply the Repetitive Nearest Neighbor Algorithm (RNNA) to this weighted graph. This involves applying the NNA with every possible starting vertex, and then taking the best (cheapest) answer amongst all outputs. Start with the NNA applied by setting A as the starting vertex:



The total cost of this tour/trip is

$$15 + 20 + 35 + 25 = 95$$

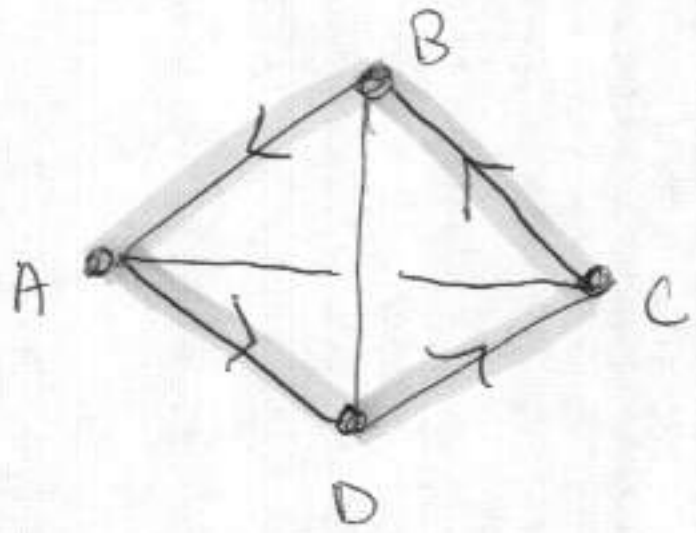
Applying NNA with B as the starting vertex yields:



total cost = 95

(same result as above, but direction is reversed)

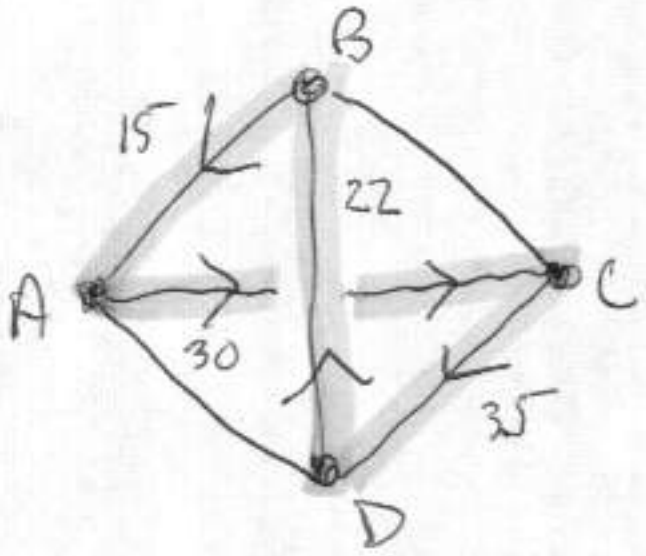
Now apply NNA with C as starting vertex:



total cost = 95

(same as above)

And finally, apply NNA with D as the start:

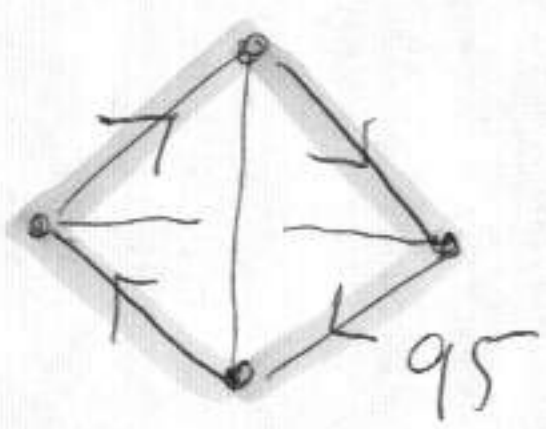


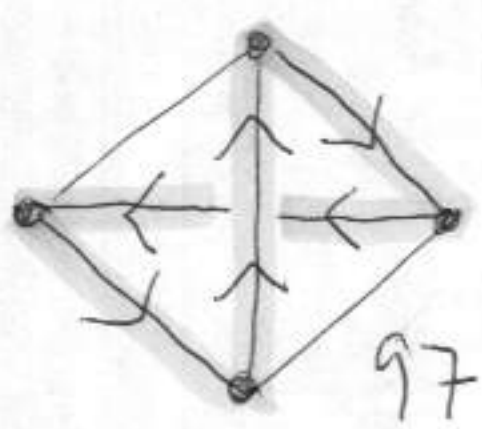
total cost = 102

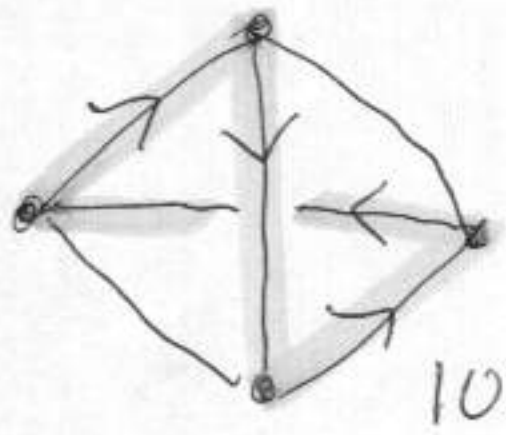
(larger than the above!)

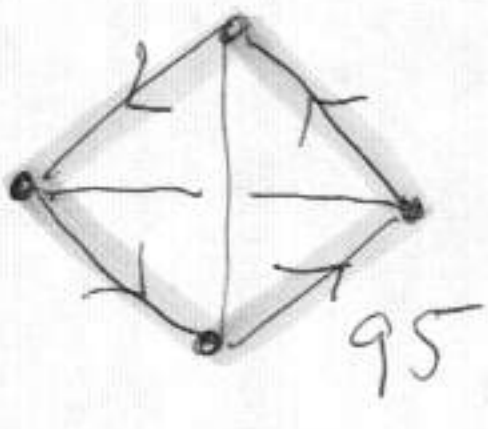
We see that if we only applied the NNA w/ D as the starting vertex then we would have ended up with a nonoptimal solution.

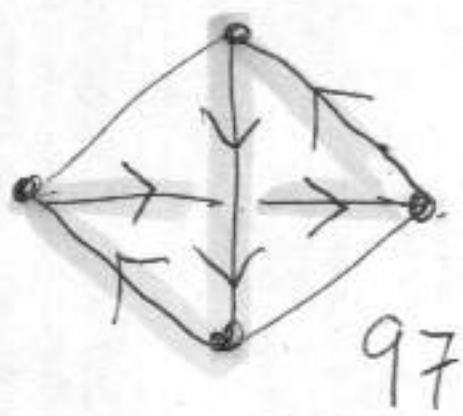
However, RNNA yields a result of total cost 95, which we can check is optimal using the Brute Force Algorithm; here are all Hamilton circuits with their total costs:

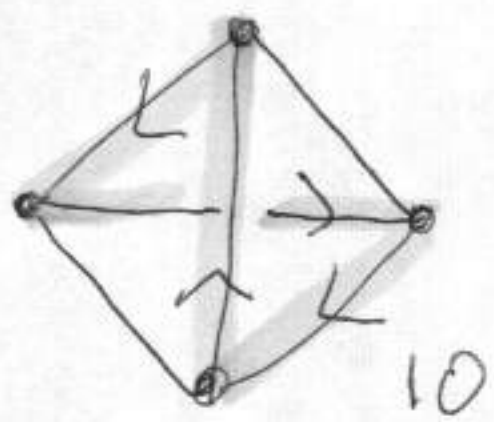












So indeed the Hamilton circuit A, B, C, D, A (or its reverse A, D, C, B, A) is the optimal (lowest cost).

Let's finish Ch. 6 with one more algorithm.

The Cheapest Link Algorithm:

Step 1: Pick the "cheapest link", i.e. the edge with minimal weight/cost.

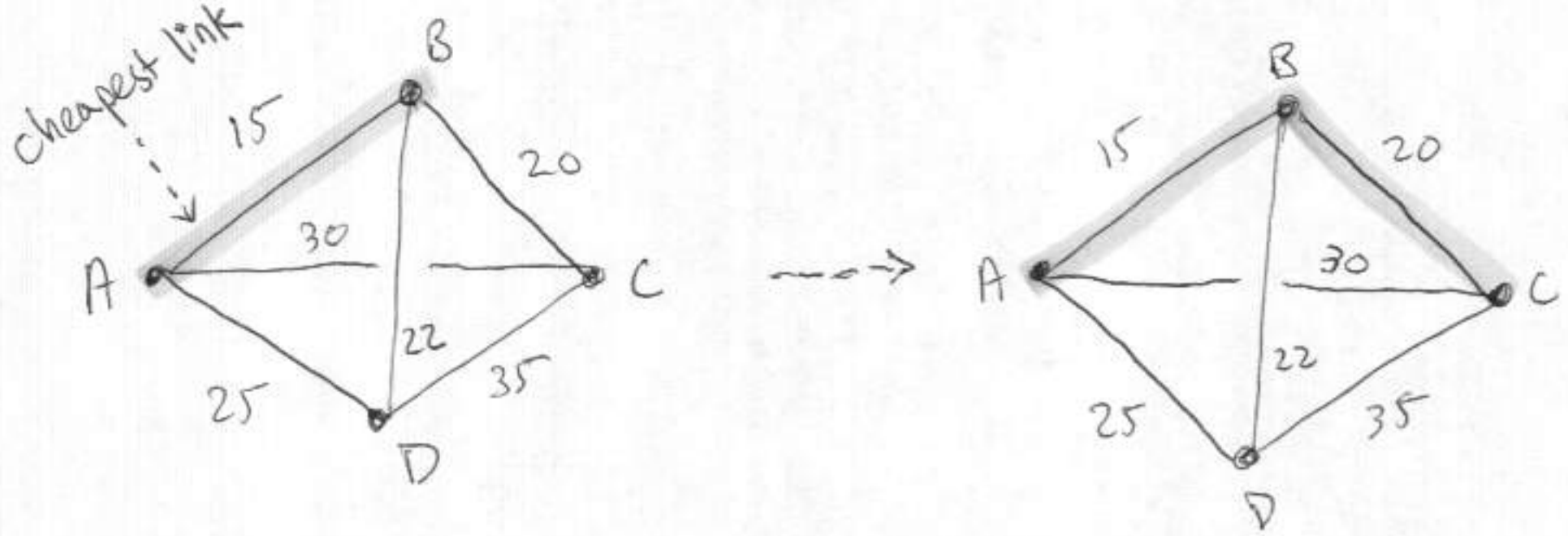
Step 2: Continue to pick, at each stage, the cheapest link available (i.e. one that has not been chosen) such that the following hold:

- the edge does not close a partial circuit made up of previously chosen edges;
- the edge does not meet a vertex where already two previously chosen edges meet.

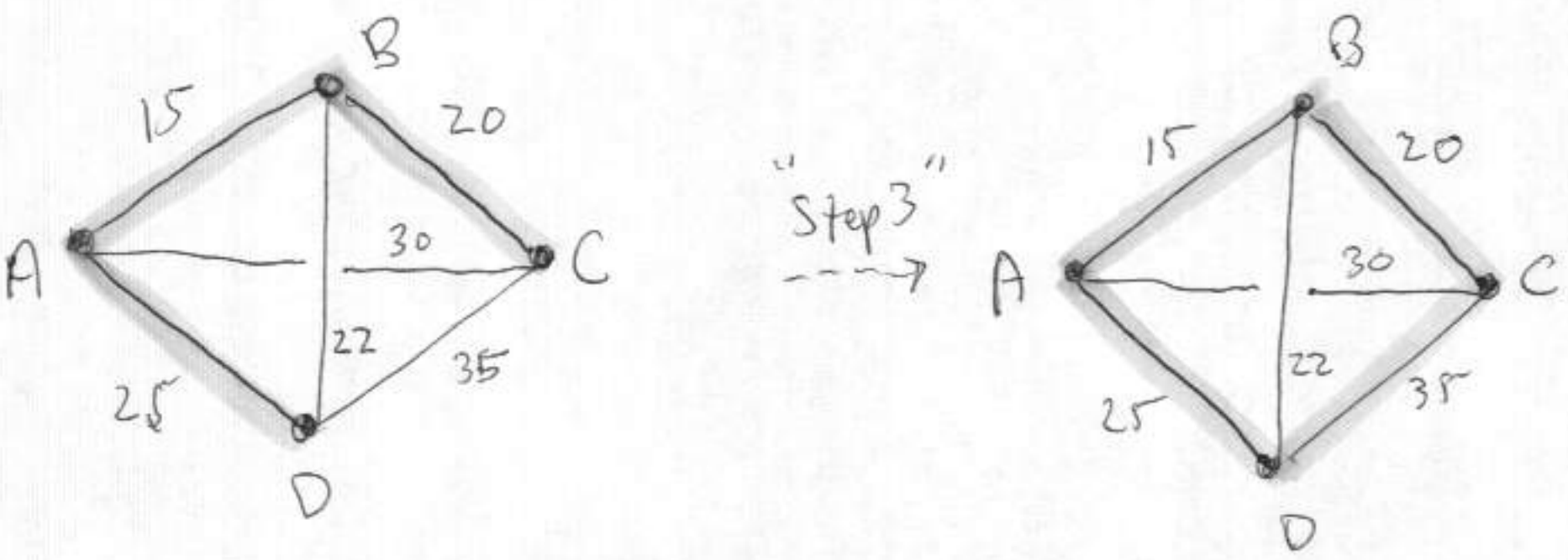
Step 3: After repeating Step 2 so that eventually all vertices are touched by chosen edges, there will be two vertices that have only one chosen edge meeting — connect these two vertices by the unique edge connecting them.

We are left with a Hamilton Circuit, and we can then choose a direction of travel.

Example Use the weighted graph we started with.



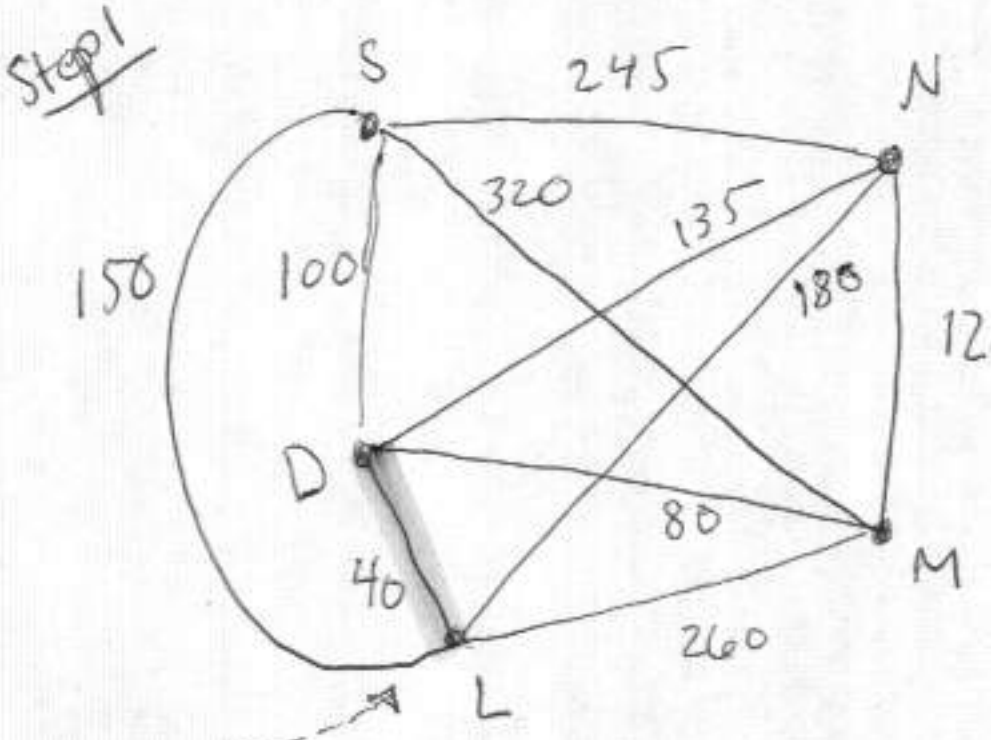
The next cheapest link is the edge BD with weight 22, but if we add that edge then there are 3 red (shaded) edges coming out of vertex B; the algorithm says this is not allowed. So we choose the next cheapest link, AD:



So we get the optimal tour in this example.

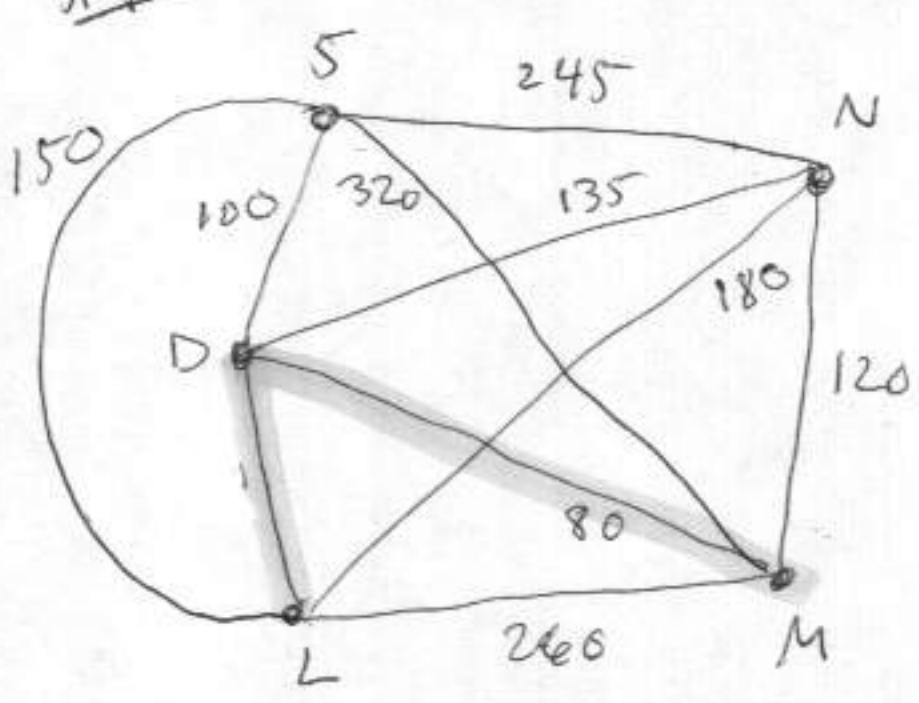
Let's apply the Cheapest Link Algorithm to our problem about flying to different cities:

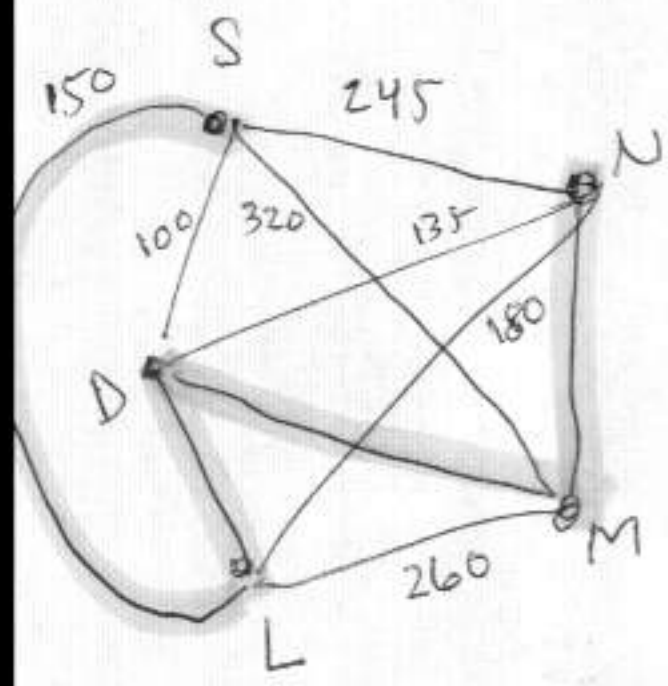
Step 1

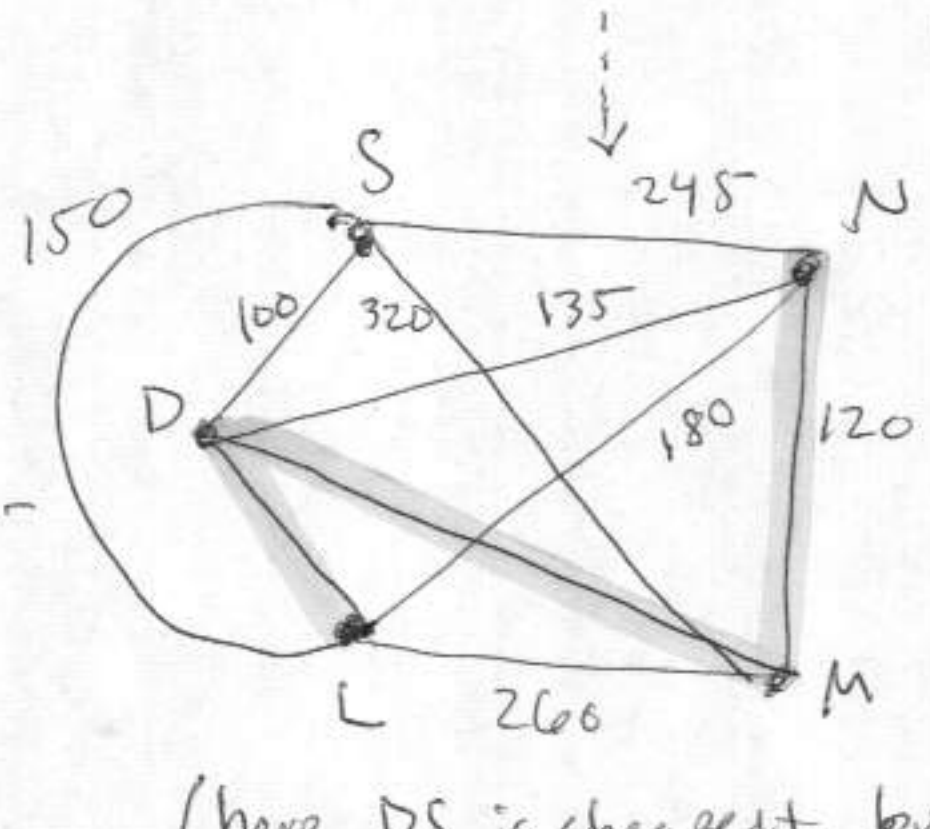


Cheapest link

Step 2

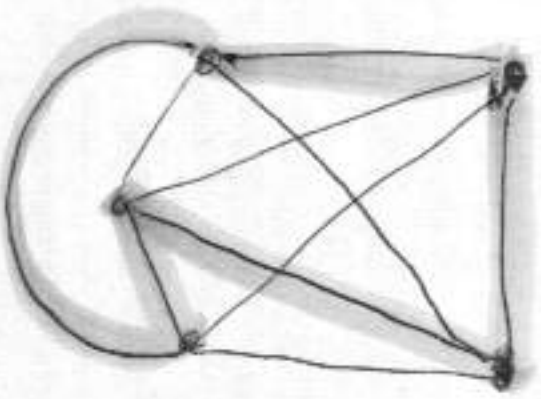






(here DS is cheapest, but violates the rule about 3 edges at 1 vertex, so the next cheapest is NM

Step 3



the end

N, M, D, L, S, N

total cost = $635

We saw from the Brute Force Algorithm that this is not quite the optimal tour: there is a trip that costs $630 instead of $635. In fact we obtained this same tour using the NNA.

Thus the Cheapest Link Algorithm, just like the NNA and RNNA, is only an approximate algorithm.