The Mathematics of Touring

Beth Kirby and Carl Lee

University of Kentucky
MA 111

Fall 2009

Course Information

Text: Peter Tannenbaum, *Excursions in Modern Mathematics*, second custom edition for the University of Kentucky, Pearson.
Course Website:
`http://www.ms.uky.edu/~lee/ma111fa09/ma111fa09.html`

6.3 Traveling Salesman Problems

Planning a Road Trip

Use the worksheet to plan a road trip through six cities in Kentucky.

The chart below contains the driving distances (in miles) among six cities in Kentucky.

Planning a Road Trip

	Lex	Lou	Eliz	Cor	Owens	Pad
Lexington		74	85	92	175	256
Louisville	74		45	158	109	218
Elizabethtown	85	45		143	94	175
Corbin	92	158	143		234	315
Owensboro	175	109	94	234		130
Paducah	256	218	175	315	130	

1. Plan a trip that starts in Lexington, visits the other five cities, and then returns to Lexington, with the least amount of driving. Try to develop a strategy that will result in the shortest trip.
2. Do the same as in (1), but now suppose your trip must start and end in Louisville. Does your route change?

The Traveling Salesman Problem

You have just encountered a traveling salesman problem.

The Traveling Salesman Problem

You have just encountered a traveling salesman problem.

Characteristics of traveling salesman problems (TSPs):

- The tour passes through several sites (cities), starting and ending at the same site.
- Each site (other than the starting/ending one) is visited exactly once.
- A cost (distance) is associated to each leg of the tour.
- The goal is to find an optimal tour. We wanted to minimize the total distance.

Algorithms

An algorithm is a problem-solving tool made up of procedural rules that result in some kind of solution to the problem.

The algorithm must have clearly defined rules that eventually come to an end.

Algorithms

An algorithm is a problem-solving tool made up of procedural rules that result in some kind of solution to the problem.

The algorithm must have clearly defined rules that eventually come to an end.

For example, each voting method we discussed was a type of algorithm. After a certain number of steps, a winner of the election was determined.

Algorithms for Solving the KY Road Trip Problem

- If you list out all of the possible routes and calculate each of their total distances, you can pick the shortest route. This is called the Brute Force Algorithm.

Algorithms for Solving the KY Road Trip Problem

- If you list out all of the possible routes and calculate each of their total distances, you can pick the shortest route. This is called the Brute Force Algorithm.
- From the starting city, you can choose to go to the closest city. Repeat this idea, so that from each city, you choose to go to the closest city that has not yet been visited. This is called the Nearest Neighbor Algorithm.

Applications: Other Examples of TSP

Mr. Loman is a traveling salesman. He must start and end in his hometown and then visit five other cities once. He knows the cost of a one-way plane ticket between each of the cities. What is the cheapest route for Mr. Loman?

Applications: Other Examples of TSP

Mr. Loman is a traveling salesman. He must start and end in his hometown and then visit five other cities once. He knows the cost of a one-way plane ticket between each of the cities. What is the cheapest route for Mr. Loman?

Sites: Cities.
Cost: Cost of a one-way plane ticket.
Optimal Tour: Cheapest route.

Applications: Other Examples of TSP

A school bus must drop off children at twenty different stops around town. Each child must be dropped off, and then the bus driver must return the bus to the school. The estimated time between stops is known. What is the fastest route?

Applications: Other Examples of TSP

A school bus must drop off children at twenty different stops around town. Each child must be dropped off, and then the bus driver must return the bus to the school. The estimated time between stops is known. What is the fastest route?

Sites: Bus stops.
Cost: Time between stops.
Optimal Tour: Fastest route.

Applications: Other Examples of TSP

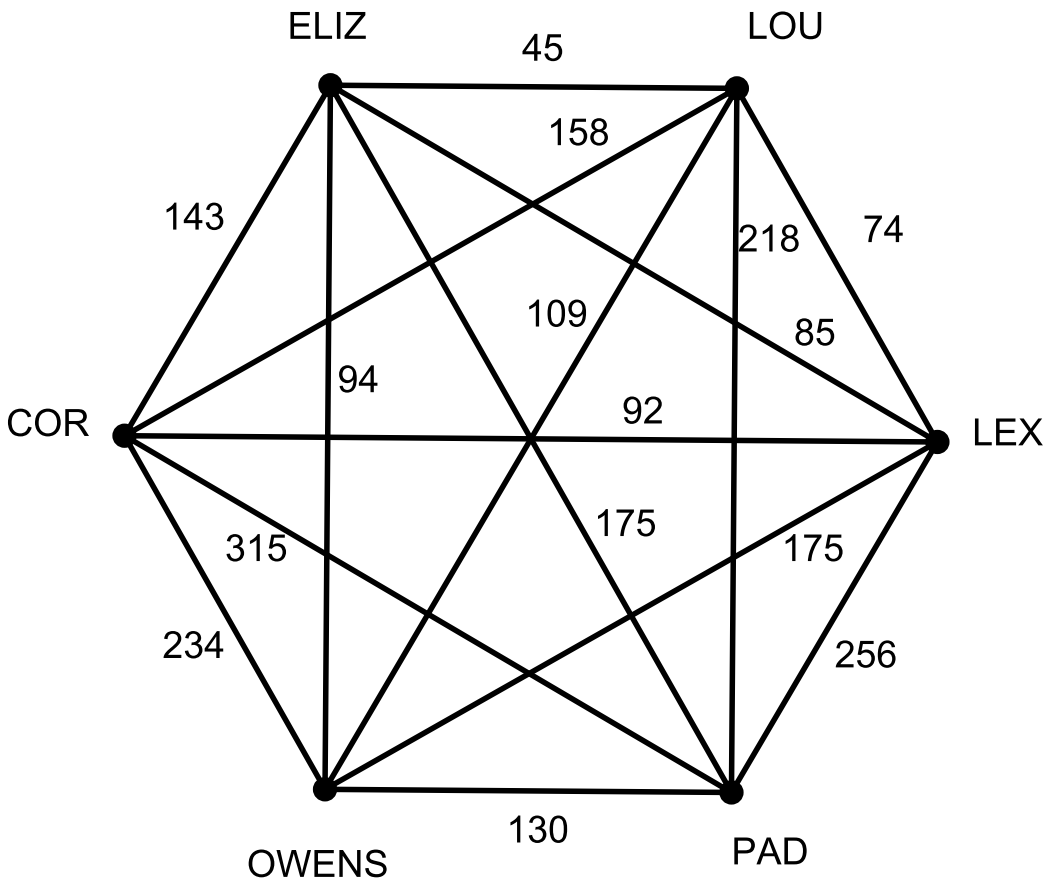
NASA's StarLight program uses a pair of satellites to image celestial objects. Each object should be imaged once. To preserve fuel, NASA must minimize the amount of fuel needed to reposition the satellites in order to take the next picture.

Applications: Other Examples of TSP

NASA's StarLight program uses a pair of satellites to image celestial objects. Each object should be imaged once. To preserve fuel, NASA must minimize the amount of fuel needed to reposition the satellites in order to take the next picture.

Sites: Celestial objects.
Cost: Fuel needed for repositioning satellites.
Optimal Tour: Sequence with minimal fuel usage.

Modeling as a Weighted Graph



Modeling as a Weighted Graph

We can model a Traveling Salesman Problem by a weighted graph. Note that this is a different use of the word graph that you have seen when "graphing a function."

Sites: Vertices of the graph; each one is a vertex.
Cost: Weights of the edges.
Tour: Hamilton circuit of the graph ("closed" circuit that does not re-use any edge and uses every vertex once and only once).
Optimal Tour: Hamilton circuit of least total weight.

Modeling as a Weighted Graph

We can model a Traveling Salesman Problem by a weighted graph. Note that this is a different use of the word graph that you have seen when "graphing a function."

Sites: Vertices of the graph; each one is a vertex.
Cost: Weights of the edges.
Tour: Hamilton circuit of the graph ("closed" circuit that does not re-use any edge and uses every vertex once and only once).
Optimal Tour: Hamilton circuit of least total weight.

The graph used in this model is called a complete graph because every pair of vertices is joined by an edge.

Modeling as a Weighted Graph

Notice that the model does not depict the *actual* relative physical positions of the cities; it is merely a way of *representing* the problem in a concise way.

On the Web

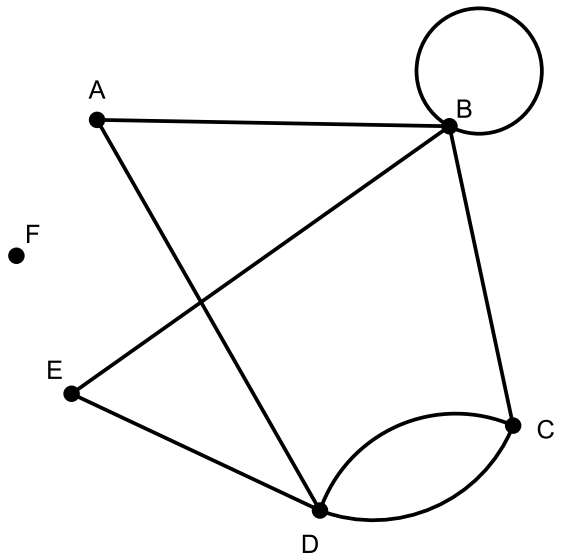
For more background and applications, see:
`http://www.tsp.gatech.edu/index.html`

Graphs

Refer to Section 5.2 in the textbook.

A graph can be thought of as a picture. There is a set of "dots" called the vertices of the graph (each one of which is called a vertex, and a set of "lines" called edges. The particular locations of the vertices does not matter, and it also does not matter if the edges are drawn with straight or curved lines, because all we interested in are the connections or relationships between various pairs of vertices.

Example



This graph has vertex set $\mathcal{V} = \{A, B, C, D, E, F\}$ and edge set $\mathcal{E} = \{AB, AD, BB, BC, BE, CD, CD, DE\}$.

Paths and Circuits

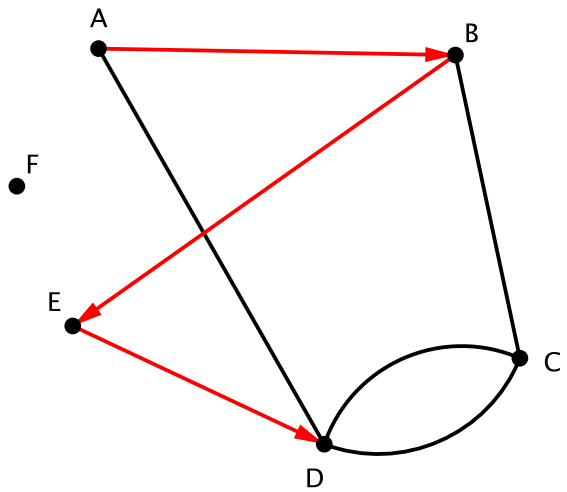
A path is a "trip" from vertex to vertex, following the edges, that *uses no edge more than once* and *does not end where it starts*.

Paths and Circuits

A path is a "trip" from vertex to vertex, following the edges, that *uses no edge more than once* and *does not end where it starts*.

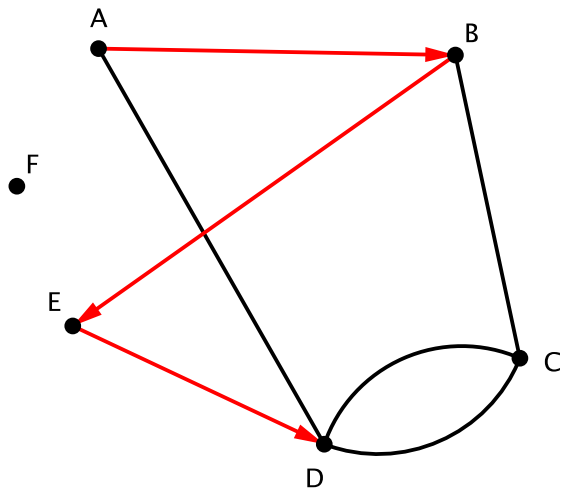
A circuit is a "trip" from vertex to vertex, following the edges, that *uses no edge more than once* and *ends where it starts*.

Example of a Path



An example of a path, denoted A, B, E, D.

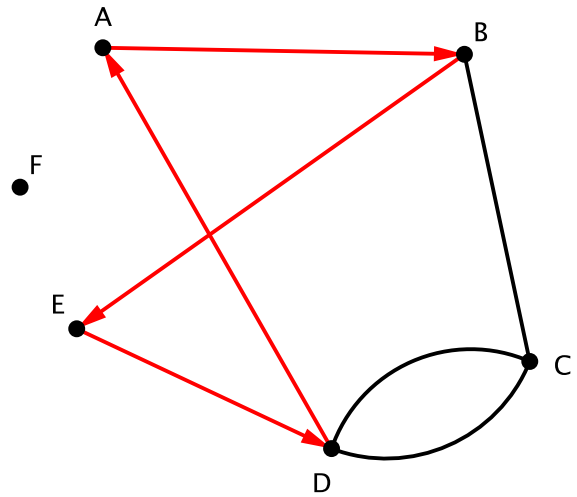
Example of a Path



An example of a path, denoted A, B, E, D.

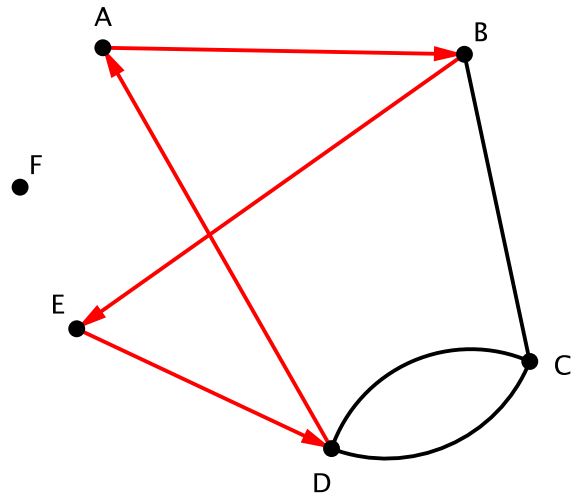
Note that this is regarded as *different* than the path D, E, B, A.

Example of a Circuit



An example of a circuit, denoted A, B, E, D, A.

Example of a Circuit

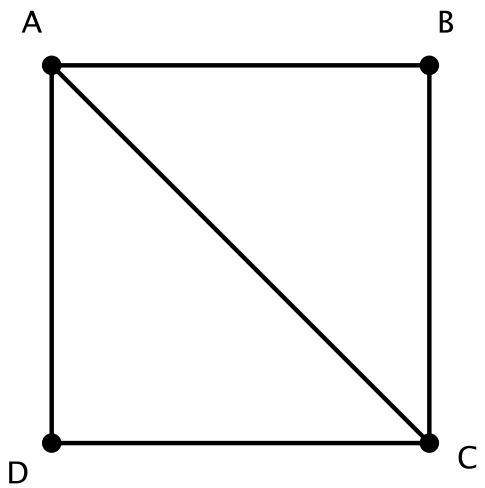


An example of a circuit, denoted A, B, E, D, A.

Note that this is regarded as the *same* circuit as E, D, A, B, E, but *different* than the circuit A, D, E, B, A.

Some Problems

List all of the paths and all of the circuits in this graph. Remember: You are not allowed to use an edge more than once.



GeoGebra

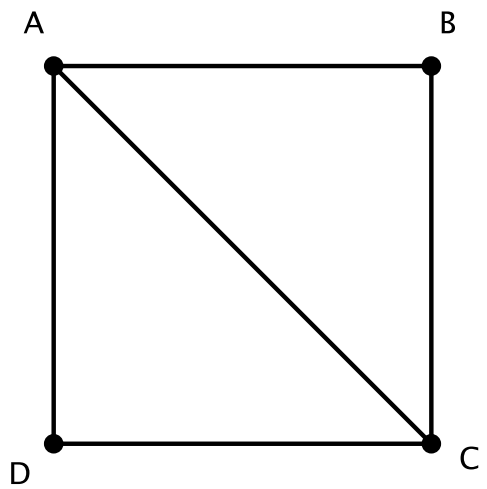
GeoGebra is a nice free drawing tool, and can be downloaded from `http://www.geogebra.org/cms/`.

6.1 Hamilton Paths and Circuits

Hamilton Paths

A Hamilton path is a path that includes every vertex of the graph *once and only once*.

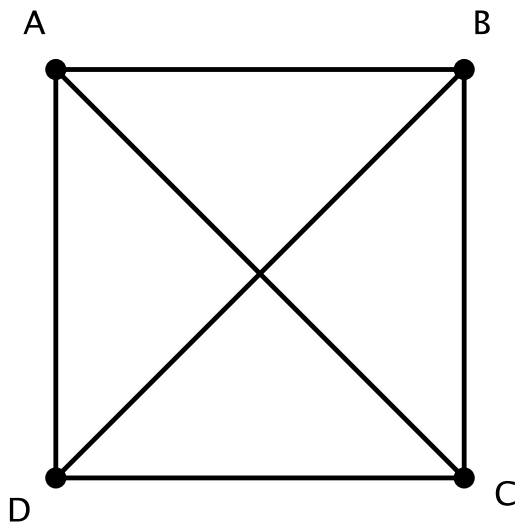
Problem: Find all the Hamilton paths in the graph below:



Hamilton Circuits

A Hamilton circuit is a circuit that includes every vertex of the graph *once and only once* (except that you must return to the starting vertex).

Problem: Find all the Hamilton circuits in the graph below:

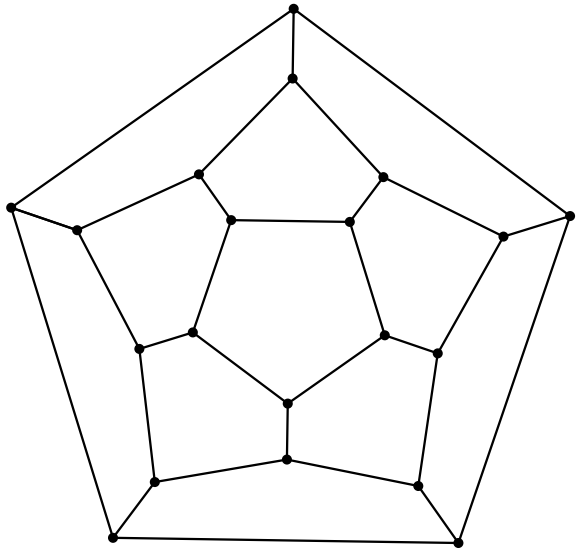


Hamilton Circuits

Remember that you can start the same circuit at different points (the starting vertex is called the reference point), but these are all still regarded *as the same circuit*. But traversing the circuit "backwards"—the mirror circuit—is regarded as *different*.

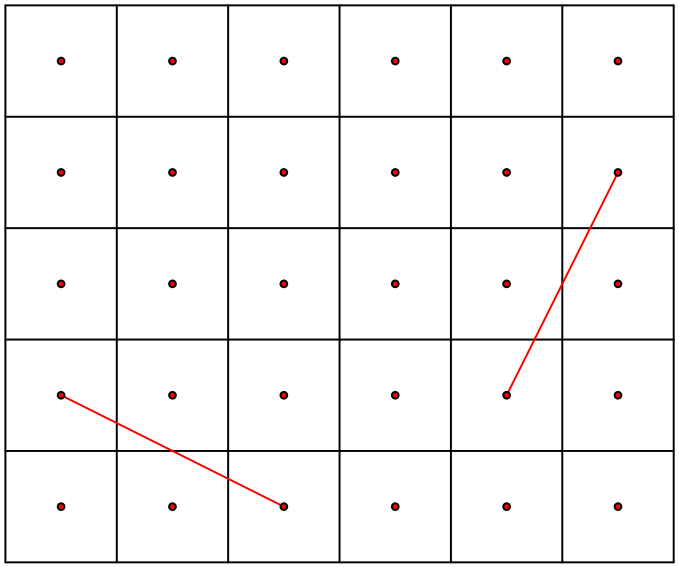
The Dodecahedron Problem

A classic Hamilton circuit problem (in fact, the origin of the name, since this puzzle was proposed by the mathematician Sir William Rowan Hamilton) is to find a Hamilton circuit on the graph of a *dodecahedron*. Can you find one?



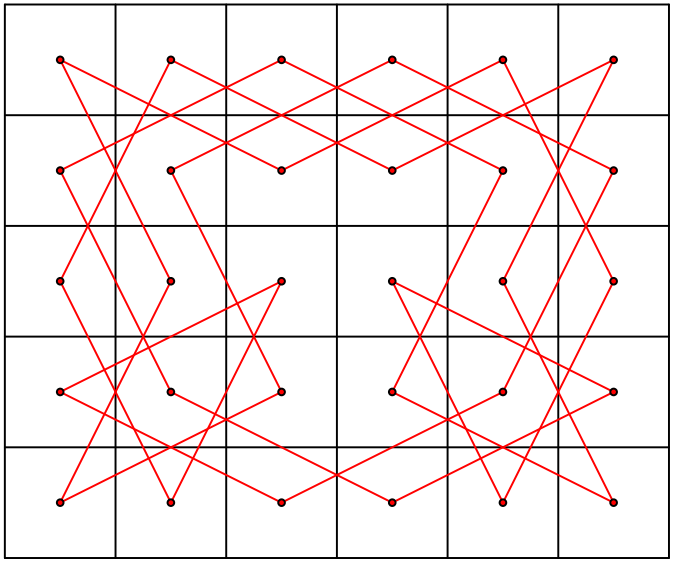
Knight's Tours

Another classic Hamilton circuit problem is to find a circuit of knight's moves on a chessboard, so that the knight moves through every square of the chessboard one and only once, returning to its starting square.
You can warm up on a smaller 5×6 board:



Knight's Tours

Here is one solution for the 5×6 board. Can you find another?



Knight's Tours

Can you find a knight's tour on a 4×4 board?

On a 5×5 board?

On a 6×6 board?

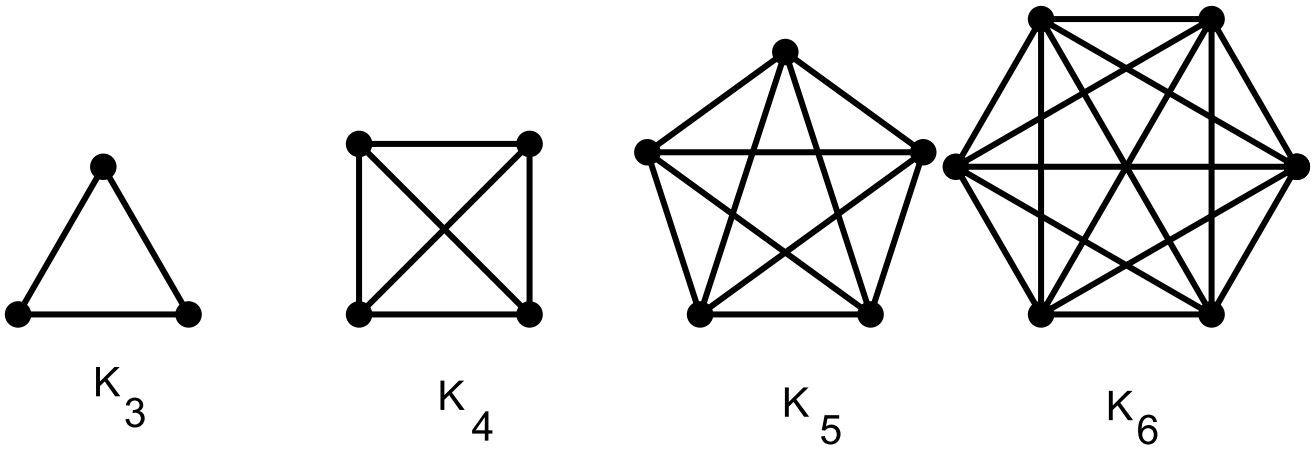
On a standard 8×8 board?

6.2 Complete Graphs

Complete Graphs

A complete graph on N vertices is a graph having N vertices in which *every* pair of vertices is joined by an edge. It is denoted K_N. As we have seen, these are useful in representing Traveling Salesman Problems.

Examples:



Counting Edges in Complete Graphs

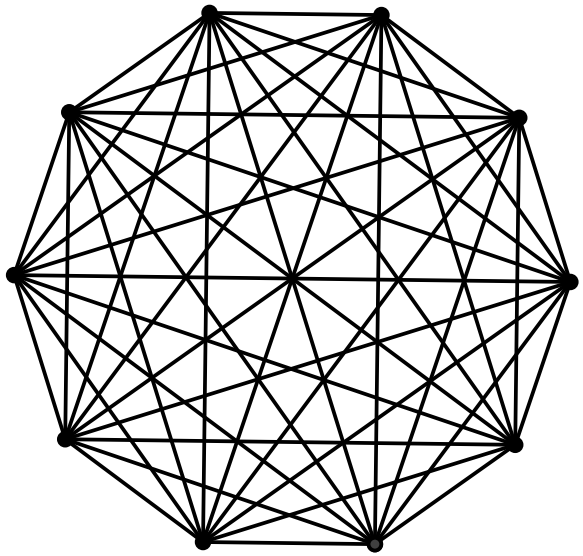
Problem:

How many edges does K_{10} have?

How many edges does K_{100} have?

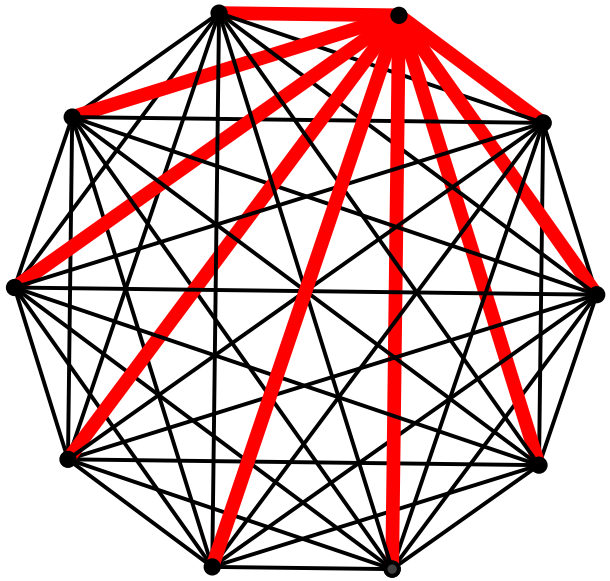
How many edges does K_N have?

Counting Edges in Complete Graphs



K_{10} has 10 vertices.

Counting Edges in Complete Graphs



Each vertex of K_{10} has 9 edges joining it.
You might think at first that therefore the number of edges is $10 \times 9 = 10$, but this counts every edge twice (from both ends), so the actual number of edges is half that, or 45.

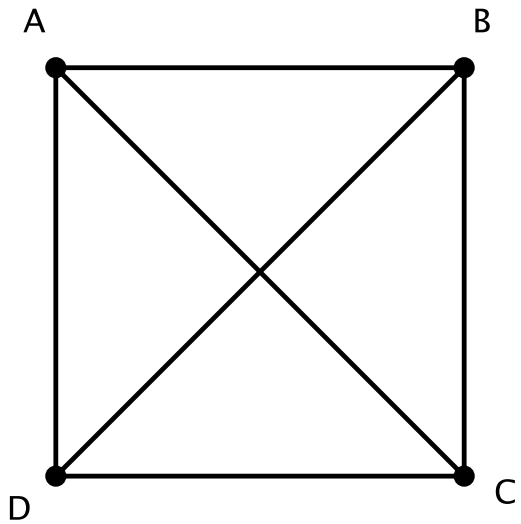
Counting Edges in Complete Graphs

In general, K_N has N vertices.
Each vertex of K_N has $N-1$ edges joining it.
Multiplying $N(N-1)$ counts every edge twice, so the total number of edges is half this quantity:

The graph K_N has $\frac{N(N-1)}{2}$ edges.

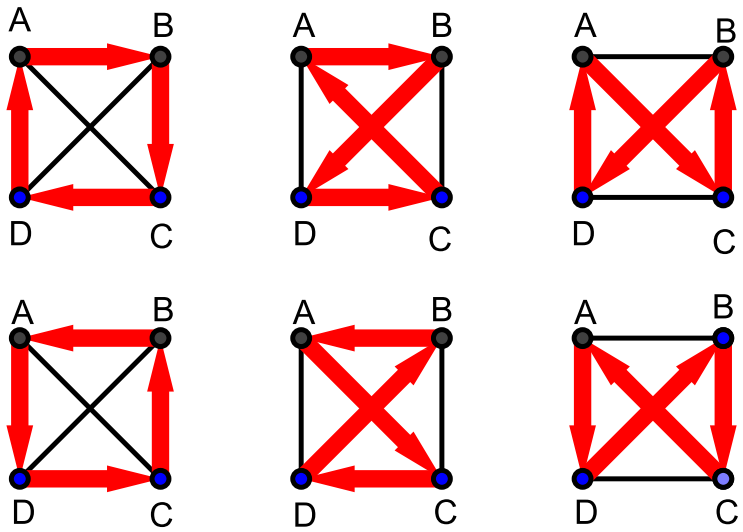
Counting Hamilton Circuits in Complete Graphs

Problem: How many Hamilton circuits does K_4 have?



Suggestion: Since the same circuit can be started at any vertex, choose one particular vertex, say, A, as a reference point when listing the vertices in the circuit.

Counting Hamilton Circuits in Complete Graphs

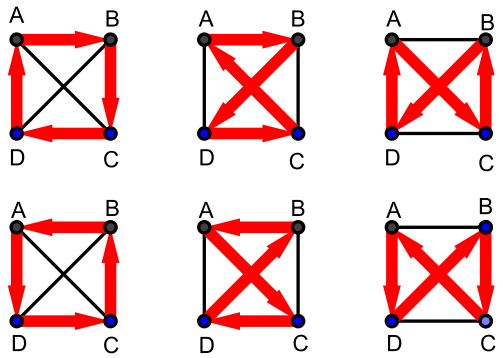


K_4 has 6 Hamilton circuits:

A, B, C, D, A A, B, D, C, A A, C, B, D, A
A, D, C, B, A A, C, D, B, A A, D, B, C, A

Which ones are mirror circuits of each other?

Counting Hamilton Circuits in Complete Graphs



Let's choose A as a reference point. After A, there are 3 choices for the second vertex. After the first two vertices, there are only 2 choices for the third vertex. After the third vertex, there is only 1 choice for the fourth vertex. Then you must return to A.

So the total number of possibilities is $3 \times 2 \times 1 = 6$.

Counting Hamilton Circuits in Complete Graphs

Problem: How many Hamilton circuits does K_5 have?

Counting Hamilton Circuits in Complete Graphs

Problem: How many Hamilton circuits does K_5 have?

$4 \times 3 \times 2 \times 1 = 24 = 4!$. Why does this make sense?

Counting Hamilton Circuits in Complete Graphs

Problem: How many Hamilton circuits does K_N have?

Counting Hamilton Circuits in Complete Graphs

Problem: How many Hamilton circuits does K_N have?

The complete graph K_N has

$$(N-1) \times (N-2) \times (N-3) \times 3 \times 2 \times 1 = (N-1)!$$

Hamilton circuits.

Why does this make sense?

6.4–6.8 Algorithms for the TSP

The Traveling Salesman Problem

Goal: To find a Hamilton circuit of optimal weight in a complete weighted graph.

The Traveling Salesman Problem

Goal: To find a Hamilton circuit of optimal weight in a complete weighted graph.

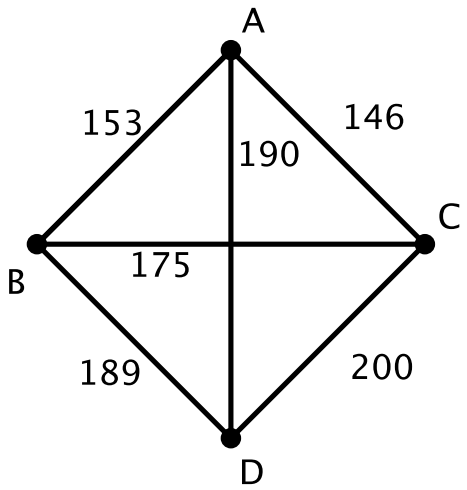
Translation: To find the cheapest tour that visits each site exactly once and returns to the starting position.

The Brute Force Algorithm

We have already seen this algorithm, but to review:

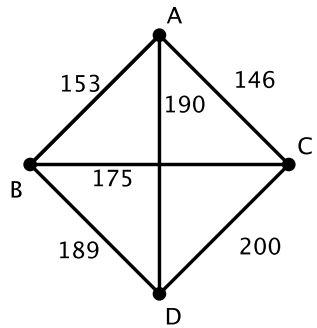
- List *all* possible Hamilton circuits (tours) of the graph.
- Calculate the total weight of each tour.
- The tour(s) with the smallest total weight is an optimal tour.

Applying the Brute Force Algorithm



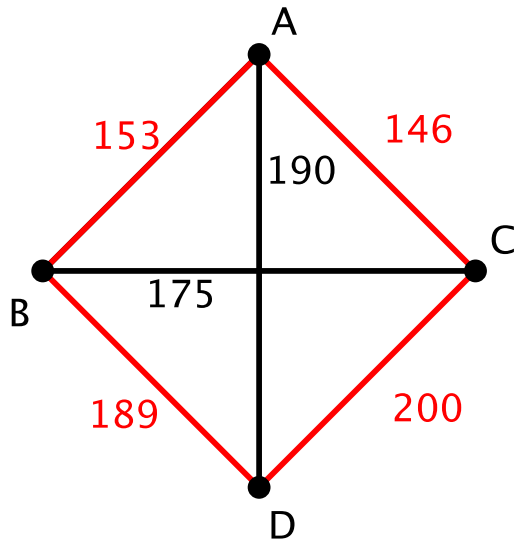
Use the Brute Force Algorithm to find an optimal tour for the above graph. Assume the tour starts and ends at A.

Applying the Brute Force Algorithm



Hamilton Circuit	Total Weight	
A, B, C, D, A	153+175+200+190=	718
A, B, D, C, A	153+189+200+146=	688
A, C, B, D, A	146+175+189+190=	700
A, C, D, B, A	146+200+189+153=	688
A, D, B, C, A	190+189+175+146=	700
A, D, C, B, A	190+200+175+153=	718

Applying the Brute Force Algorithm



Both A, B, D, C, A and A, C, D, B, A are optimal tours for the graph.

How Efficient is the Brute Force Algorithm?

Recall that K_N (the complete graph with N vertices) has $(N-1)!$ Hamilton circuits.

How Efficient is the Brute Force Algorithm?

Recall that K_N (the complete graph with N vertices) has $(N-1)!$ Hamilton circuits.

How long will our list of all possible Hamilton circuits be?

Number of Vertices	Number of Hamilton Circuits
4	$3! = 6$
5	$4! = 24$
6	$5! = 120$
7	$6! = 720$
10	$9! = 362{,}880$
15	$14! = 87{,}178{,}291{,}200$
25	$24! \approx 6 \times 10^{23}$

Even the fastest computers in the world become useless beyond 25 vertices.

How Efficient is the Brute Force Algorithm?

The fastest computer in the world can do about 1 quadrillion (1×10^{15}) computations per second.

To complete the Brute Force Algorithm for K_{25}, the computer needs

$$\frac{6 \times 10^{23}}{1 \times 10^{15}} = 6 \times 10^{8} \text{ seconds.}$$

That's equivalent to almost 20 years of computations.
(Try this out using `http://www.wolframalpha.com`.)

How Efficient is the Brute Force Algorithm?

The Brute Force Algorithm is an inefficient algorithm. The number of steps needed to complete the algorithm grows disproportionately with the number of vertices.

To go from N vertices to $N + 1$ vertices, the number of steps in the Brute Force Algorithm increases by a factor of N.

How Efficient is the Brute Force Algorithm?

The Brute Force Algorithm is an inefficient algorithm. The number of steps needed to complete the algorithm grows disproportionately with the number of vertices.

To go from N vertices to $N + 1$ vertices, the number of steps in the Brute Force Algorithm increases by a factor of N.

Is there an alternative to Brute Force?

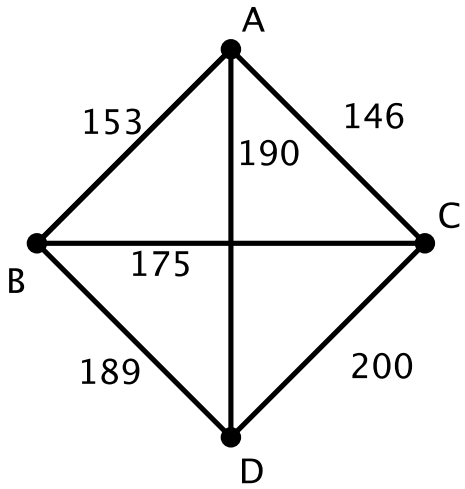
The Nearest Neighbor Algorithm

We have already seen this algorithm, but to review:

- Start at the designated vertex. From the starting vertex, move to its nearest neighbor (the vertex for which the connecting edge has the smallest weight).
- From each vertex, go to the nearest neighbor, choosing among those vertices that have not yet been visited.
- From the last vertex, return to the starting vertex.

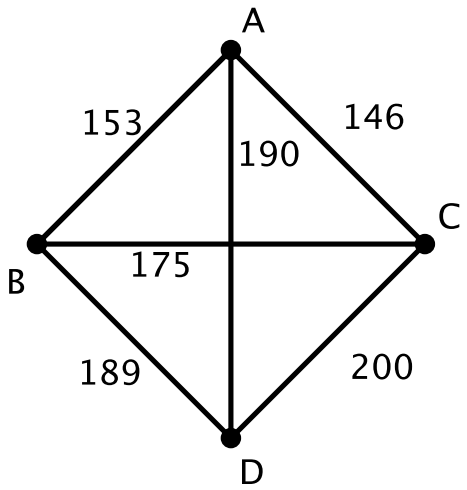
The resulting Hamilton circuit is often called a nearest neighbor tour.

Applying the Nearest Neighbor Algorithm



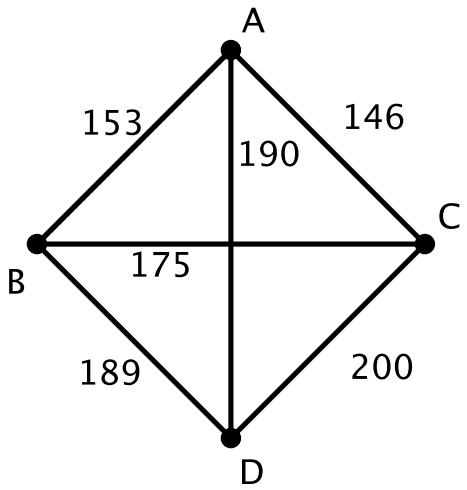
Find the nearest neighbor tour for the above graph, starting at vertex A.

Applying the Nearest Neighbor Algorithm



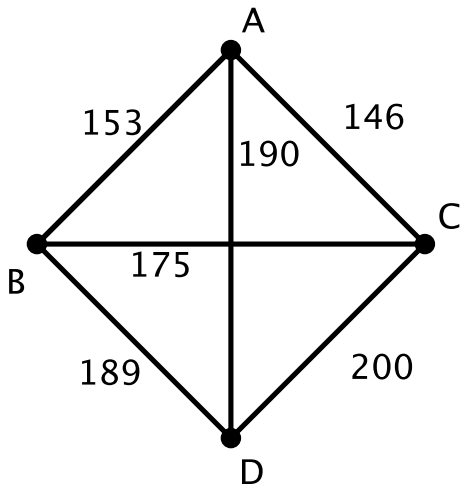
A

Applying the Nearest Neighbor Algorithm



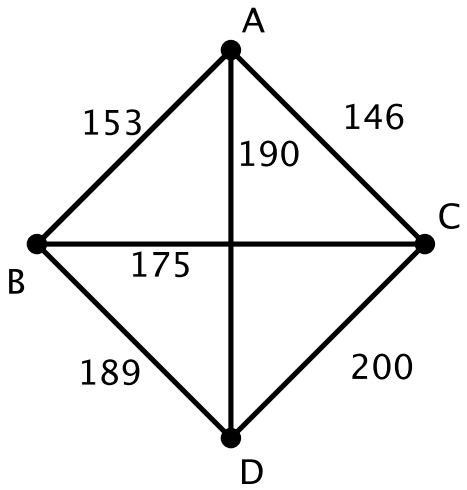
$$A \xrightarrow[146]{} C$$

Applying the Nearest Neighbor Algorithm



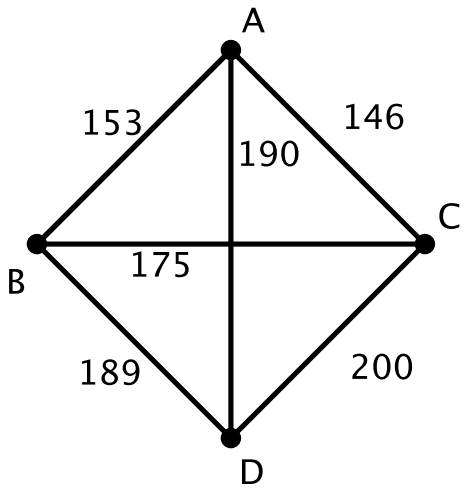
$$A \xrightarrow[146]{} C \xrightarrow[175]{} B$$

Applying the Nearest Neighbor Algorithm



$$A \xrightarrow[146]{} C \xrightarrow[175]{} B \xrightarrow[189]{} D$$

Applying the Nearest Neighbor Algorithm



$$A \xrightarrow[146]{} C \xrightarrow[175]{} B \xrightarrow[189]{} D \xrightarrow[190]{} A$$

The nearest neighbor tour is A, C, B, D, A. Notice that this tour has a total weight of 700.

How Efficient is the Nearest Neighbor Algorithm?

- The Nearest Neighbor Algorithm produced a tour that is not optimal.
  (By the Brute Force Algorithm, we know the optimal tour has weight 688.)

How Efficient is the Nearest Neighbor Algorithm?

- The Nearest Neighbor Algorithm produced a tour that is not optimal.
  (By the Brute Force Algorithm, we know the optimal tour has weight 688.)

- The Nearest Neighbor Algorithm is often used because it is an efficient algorithm.

- For K_N, the Nearest Neighbor Algorithm uses N computations.

How Efficient is the Nearest Neighbor Algorithm?

- The Nearest Neighbor Algorithm produced a tour that is not optimal.
  (By the Brute Force Algorithm, we know the optimal tour has weight 688.)

- The Nearest Neighbor Algorithm is often used because it is an efficient algorithm.

- For K_N, the Nearest Neighbor Algorithm uses N computations.

- The amount of work required for the algorithm grows in a reasonable proportion to the size of the problem.

Approximate Algorithms

The Nearest Neighbor Algorithm is an example of an approximate algorithm. It *might not produce the optimal solution* to the TSP, but it produces a solution that might be "close enough" to the optimal solution.

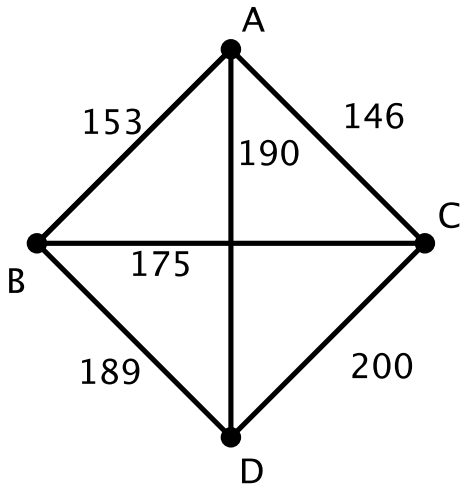
Relative Error of a Tour

In our example the Nearest Neighbor Algorithm produced an answer of 700, while the optimal answer was 688. We can calculate the relative error of our tour:

$$\text{error} = \frac{\text{cost of tour} - \text{cost of optimal tour}}{\text{cost of optimal tour}}.$$

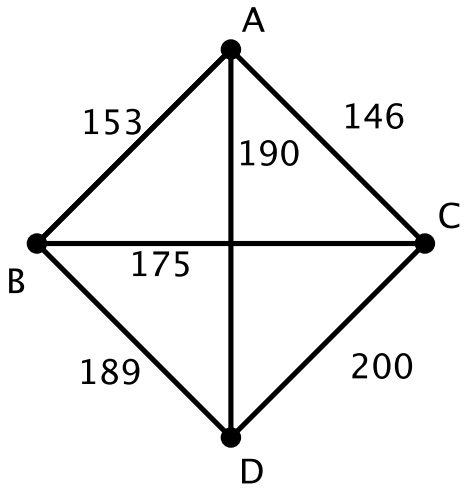
In this case the relative error equals $\frac{700-688}{688} \approx 0.0174$ or 1.74%.

A Variation on the Nearest Neighbor Algorithm



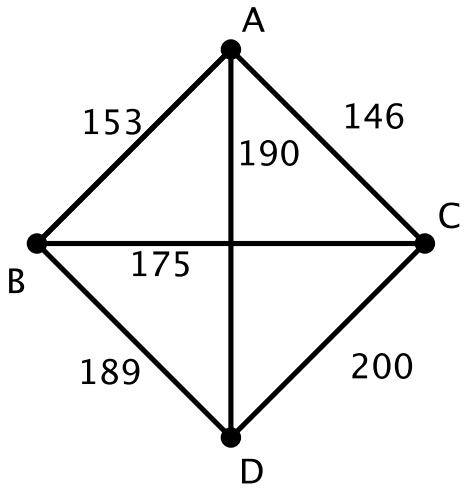
Find the nearest neighbor tour for the above graph, but this time start at vertex C.

A Variation on the Nearest Neighbor Algorithm



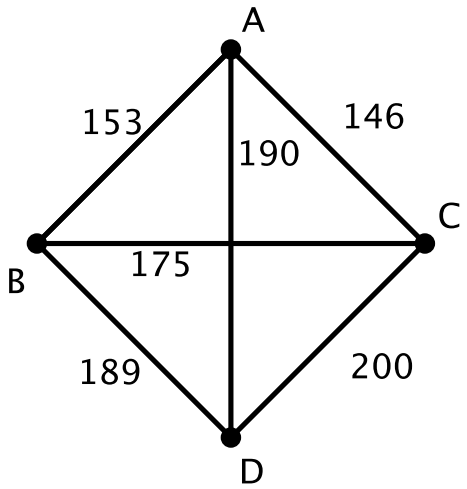
C

A Variation on the Nearest Neighbor Algorithm



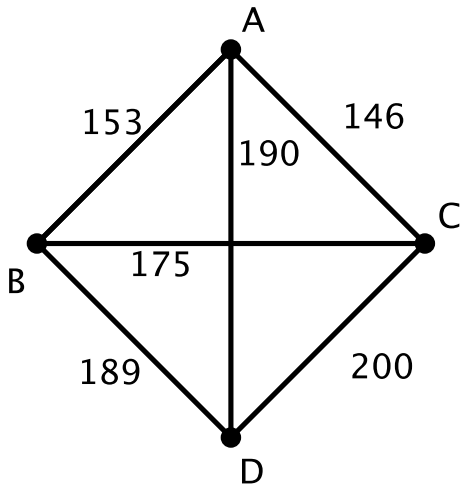
$$C \xrightarrow[146]{} A$$

A Variation on the Nearest Neighbor Algorithm



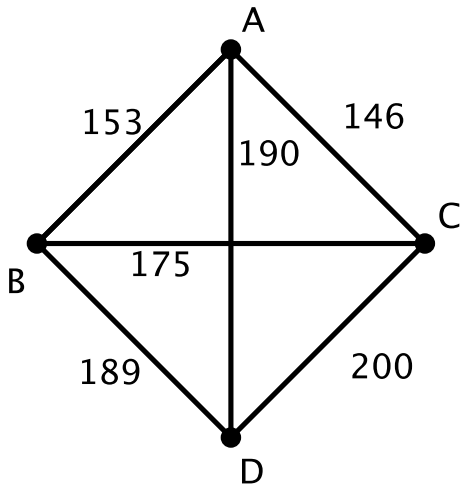
$$C \xrightarrow[146]{} A \xrightarrow[153]{} B$$

A Variation on the Nearest Neighbor Algorithm



$$C \xrightarrow[146]{} A \xrightarrow[153]{} B \xrightarrow[189]{} D$$

A Variation on the Nearest Neighbor Algorithm



$$C \xrightarrow[146]{} A \xrightarrow[153]{} B \xrightarrow[189]{} D \xrightarrow[200]{} C$$

The nearest neighbor tour is C, A, B, D, C. Notice that this tour has a total weight of 688 and is one of the optimal tours produced by the Brute Force Algorithm.

A Variation on the Nearest Neighbor Algorithm

This example shows that the Nearest Neighbor *can* produce an optimal solution sometimes.

It also shows that altering the starting vertex can produce different results when we apply the Nearest Neighbor Algorithm.

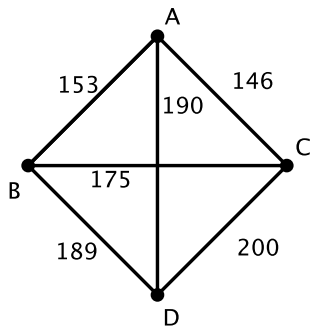
The Repetitive Nearest-Neighbor Algorithm

This suggests an improved approximation algorithm:

- Let X be any vertex. Find the nearest-neighbor tour with X as the starting vertex, and calculate the cost of the tour.
- Repeat the process with each of the other vertices of the graph as the starting vertex.
- Of the nearest-neighbor tours thus obtained, choose one with least cost. If necessary, rewrite the tour so that it starts at the designated starting vertex. We will call this tour the repetitive nearest-neighbor tour.

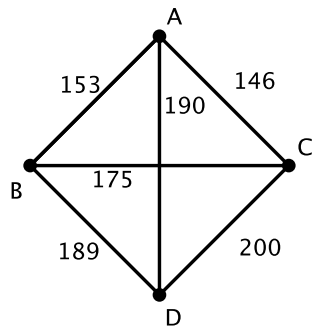
Example

You want to find a low cost tour starting and ending at A in the graph. Do NN four times, once from each vertex.



Example

You want to find a low cost tour starting and ending at A in the graph. Do NN four times, once from each vertex.



Do NN from A: A, C, B, C, A, cost 700.
Do NN from B: B, A, C, D, B, cost 688.
Do NN from C: C, A, B, D, C, cost 688.
Do NN from D: D, B, A, C, D, cost 688.

Example

Do NN from A: A, C, B, C, A, cost 700.
Do NN from B: B, A, C, D, B, cost 688.
Do NN from C: C, A, B, D, C, cost 688.
Do NN from D: D, B, A, C, D, cost 688.

The lowest cost we have found is 688. Take one of these low-cost tours, say, the second one, and rewrite it as it would be described starting at A: A, C, D, B, A. This is our repetitive nearest-neighbor tour.

Web Demos

You can try out the Brute-Force, Nearest-Neighbor, and Repetitive Nearest-Neighbor algorithms at this website:

`http://www.wiley.com/college/mat/gilbert139343/java/java09_s.html`