Networks: Lecture 1

Amedeo R. Odoni
November 15, 2006

* Thanks to Prof. R. C. Larson for some of the slides

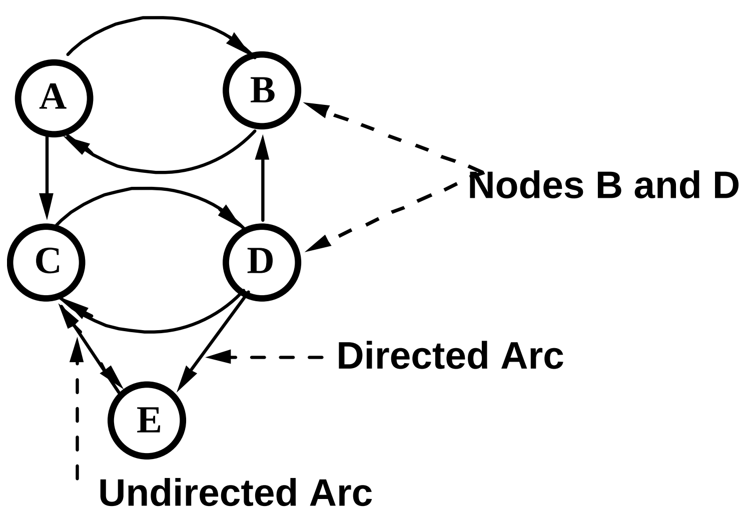
General Comments

- From continuous to a more "discretized" travel environment
- Enormous literature and variety of problems
- Transportation and logistics, urban services just two of the major areas of applications
- Level of detail of model depends on problem
- Numerous interpretations of "nodes" ("points", "vertices") and "arcs" ("links", "edges")
- Will concentrate on routing and location problems
- Will assume that efficient shortest path algorithms are available

Outline and References

- Introduction
- Minimum Spanning Tree (MST)
- Chinese Postman Problem (CPP)
- Skim Sections 6.1 and 6.2, read Sections 6.3- 6.4.4 in Larson and Odoni
- Far more detailed coverage in (among others) Ahuja, R., T. L. Magnanti and J. B. Orlin, *Network Flows,* Prentice-Hall, 1993.

Network with Terminology



Examples of Nodes & Arcs

<u>Nodes/ Vertices/ Points</u>

- Street intersections
- Towns
- Cities
- Electrical junctions
- Project milestones

<u>Arcs/ Edges/ Links</u>

- Street segments
- Country roads
- Airplane travel time
- Circuit components
- Project tasks

Network Terminology

- **N = sets of nodes**
- **A = set of arcs**
- **G(N,A)**
- **Incident arc**
- **Adjacent nodes**
- **Adjacent arcs**
- **Path**
- **Degree of a node**
- **In-degree**
- **Out-degree**
- **Cycle or circuit**
- **Connected nodes**
- **Connected undirected graph**
- **Strongly connected directed graph**
- **Subgraph**

Network Terminology - con't.

- **Tree of an undirected network is a connected subgraph having no cycles**
- **A tree having t nodes contains (t-1) edges**
- **Spanning tree of G(N,A) is a tree containing all n nodes of N**
- **Length of a path S**

$$L(S) = \sum_{(i,j)\in S} l(i,j)$$

- **d(x,y), d(i,j)**

Shortest Path Problem

- **Find the shortest path (more generally, least cost path) between two nodes, starting at Node O and ending at Node D.**
- **Dijkstra's node labeling algorithm (essentially dynamic programming); one-to-all paths; all edge lengths are non-negative; $O(n^2)$.**
- **Floyd's algorithm; negative edge lengths OK (discovers negative cycles); all-to-all paths; non-obvious; $O(n^3)$.**
- **Numerous variations and extensions: all-to-one; critical edge; k-th shortest path; shortest path on stochastic networks; shortest path on stochastic and dynamic networks**

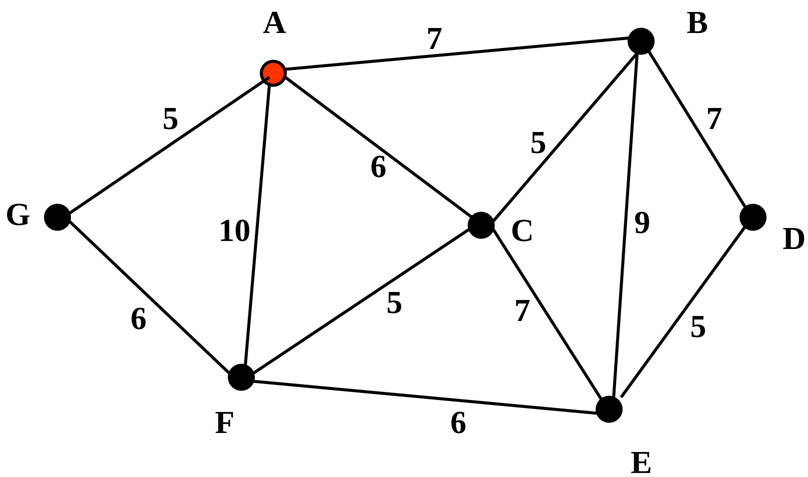
Node Labeling Algorithm: Dijkstra

- Shortest path from a node
- *k*=1, start at origin node
- At the end of iteration *k:*
  - **the set of *k* CLOSED NODES consists of the *k* closest nodes to the origin.**
  - **the label of each OPEN NODE adjacent to one or more closed nodes indicates our current 'best guess' of the minimal distance to that node.**

Minimum Spanning Tree (MST) Problem

- **Assume an undirected graph**
- **Problem: Find a shortest length spanning tree of G(N, A).**
- **Why is this an important problem?**
- **If |N|=n, then each spanning tree contains (n-1) links.**
- **MST may not be unique**

MST Example

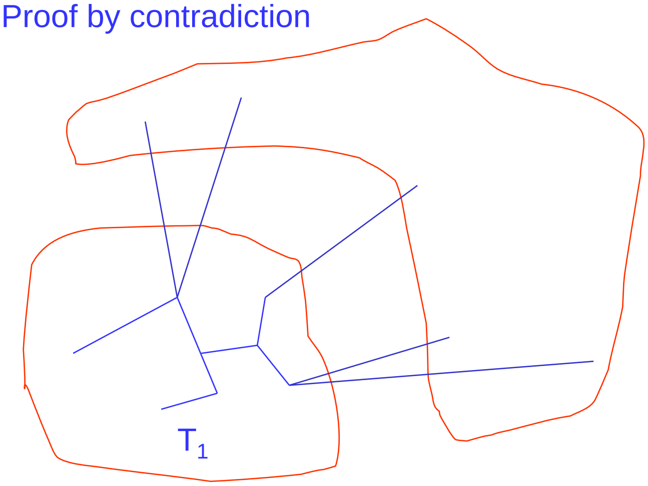


MST

- Greedy algorithm works!
- Algorithm: Start at an arbitrary node. Keep connecting to the growing sub-tree the closest unattached node.
- Fundamental property: The shortest link out of any sub-tree (during the construction of the MST) must be a part of the MST

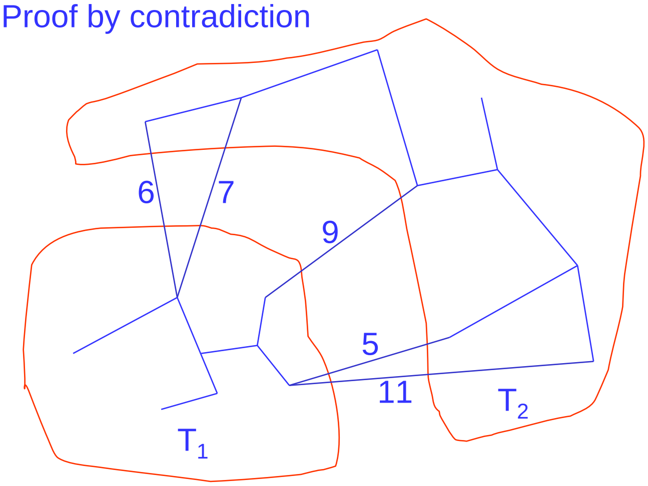
Proof of fundamental property

Proof by contradiction



Proof of fundamental property

Proof by contradiction

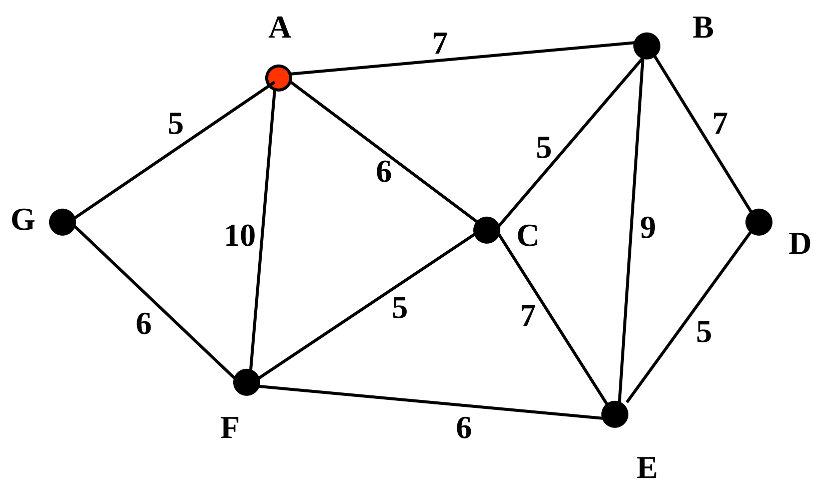


MST = T_1 + T_2 + (one connecting link)

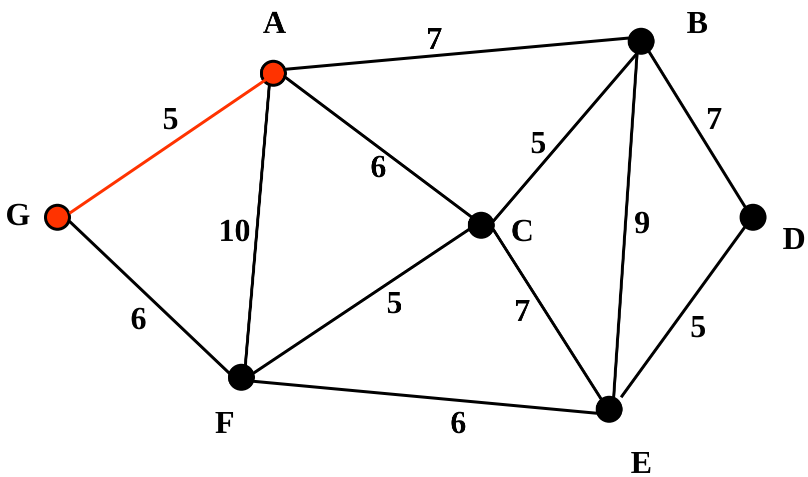
Corollary

- **In an undirected network G, the link of shortest length out of any node is part of the MST.**

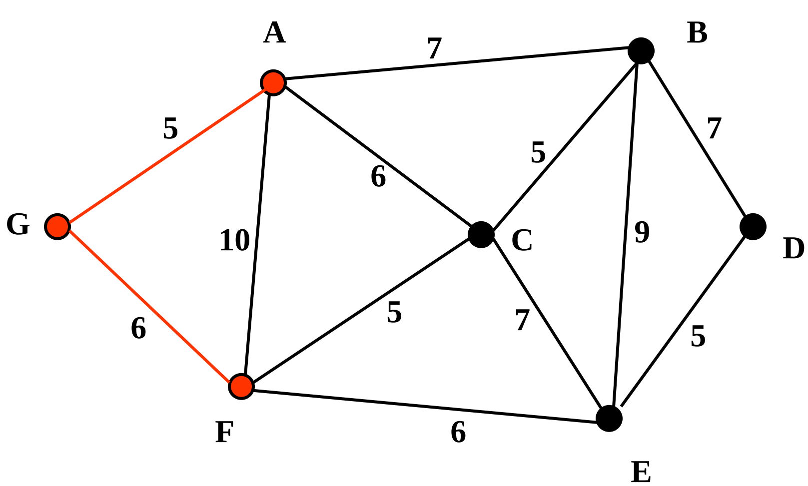
MST Example



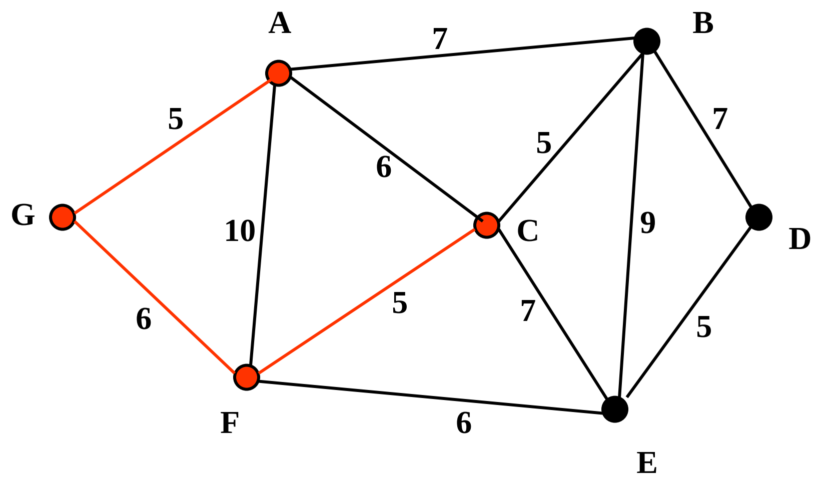
MST Example (continued)



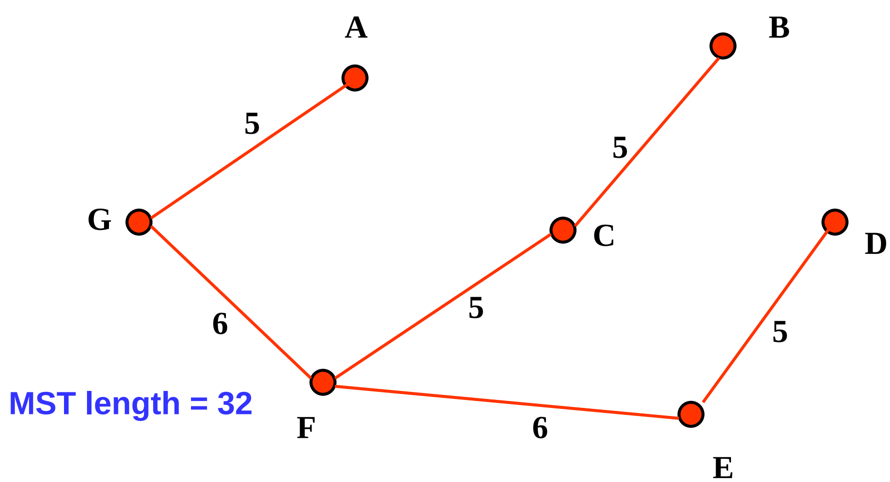
MST Example (continued)



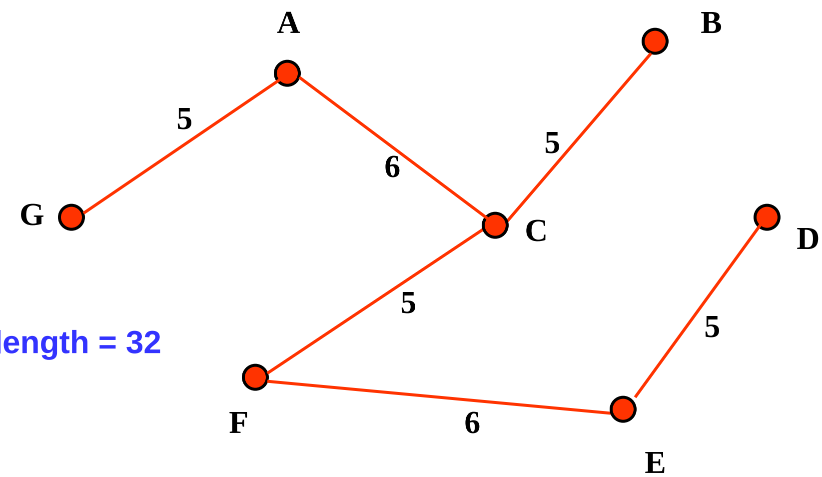
MST Example (continued)



MST Example: A Solution



MST Example: An Alternative Solution

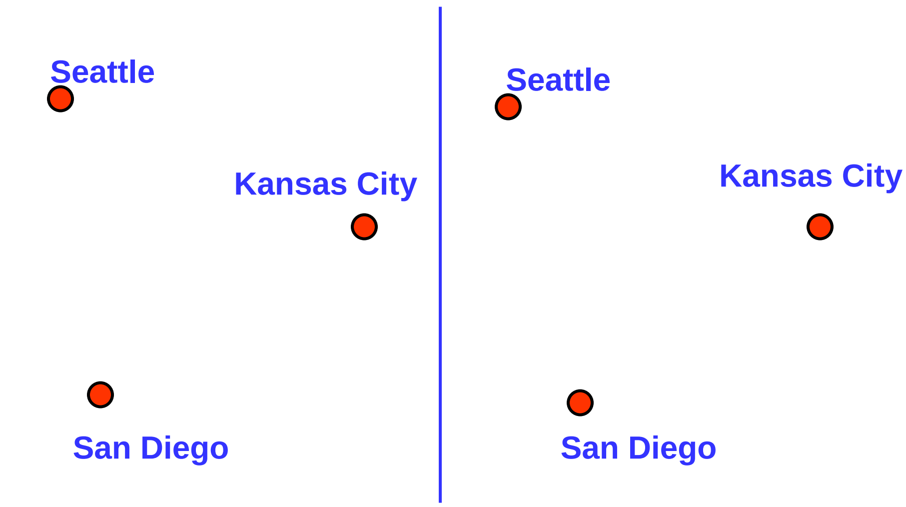


MST length = 32

MST vs. Steiner Problem in the Euclidean Plane

- MST: All links must be rooted in the node set, N, to be connected
- MST is an easy problem
- Steiner problem: Links can be rooted at any point on the plane
- The Steiner problem is, in general, very difficult

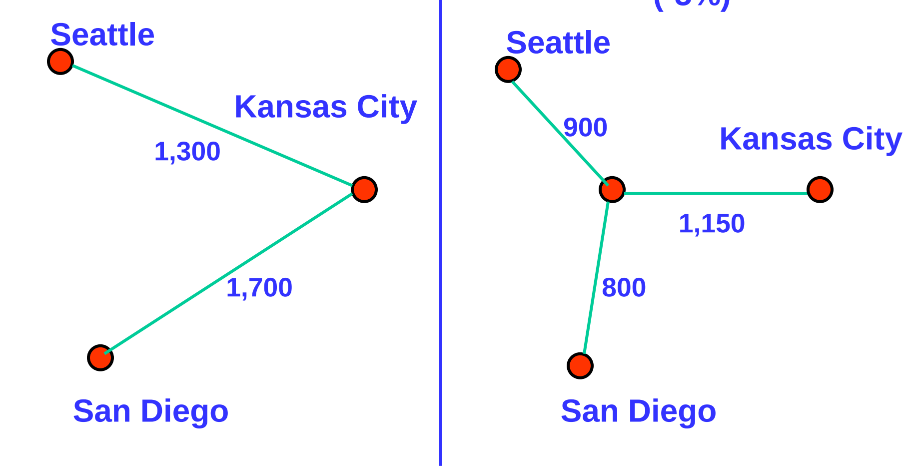
MST vs. Steiner: Example



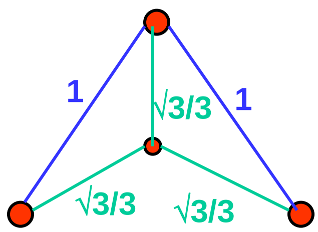
MST vs. Steiner: Example (2)

MST Total = 3,000 miles

Steiner Total = 2,850 miles (-5%)



Equilateral Triangle



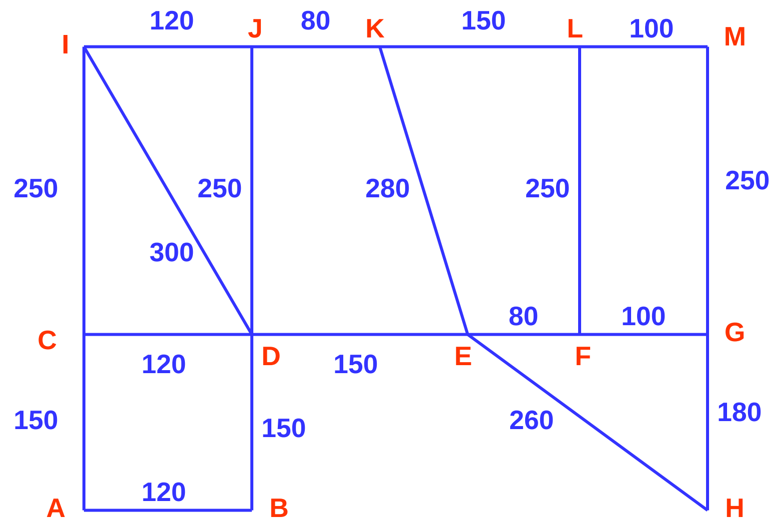
$$\frac{L(STEINER)}{L(MST)} = \frac{\sqrt{3}}{2} \approx 0.87$$

(~13% savings)

Chinese Postman Problem

- Find the minimum length tour (or cycle) that "covers" every link of a network at least once
- Will look at the CPP on an undirected network

Minimize $\sum_{(i,j)\in A} n(i,j)\cdot \ell(i,j)$ [*n(i, j)* is the no. of times (i, j) is "covered"]



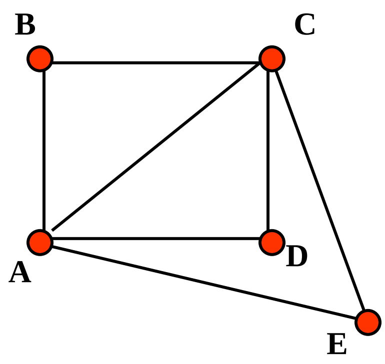
The CPP on undirected graphs: Background

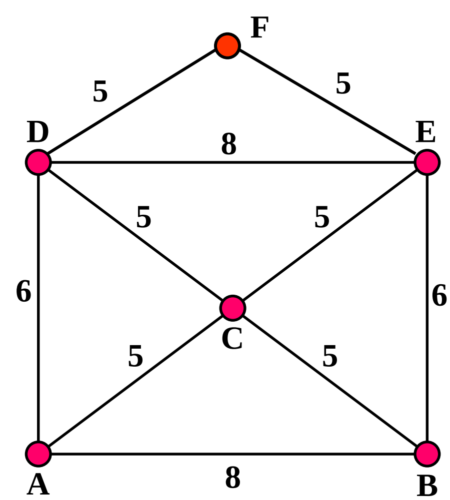
- EULER TOUR: A tour which traverses every edge of a graph exactly once.
- *If we can find an Euler tour on G(N,A), this is clearly a solution to the CPP.*
- The DEGREE of a node is the number of edges that are incident on this node.
- Euler's Theorem (1736): A connected undirected graph, G(N, A), has an Euler tour iff it contains exactly zero nodes of odd degree. [If G(N, A) contains exactly two nodes of odd degree, then an Euler PATH exists.]

The number of odd degree nodes in a graph is always even!

1. Each edge has two incidences.
2. Therefore, the total number of incidences, *P*, is an even number.
3. The total number of incidences, P_e, on the even-degree nodes is an even number.
4. Therefore, the total number of incidences, P_o, on the odd-degree nodes ($P_o = P - P_e$) is an even number.
5. But P_o is the number of incidences on odd-degree nodes. For P_o to be even, it must be that *m*, the number of odd-degree nodes, is also even.

Networks with Euler Tour or Path





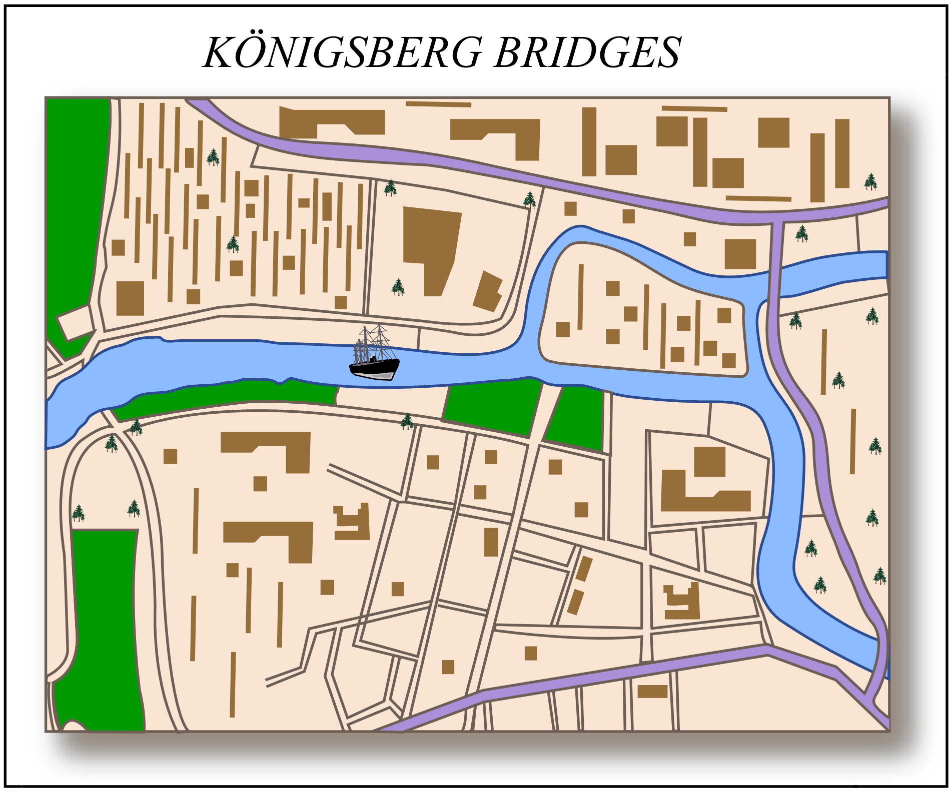


Image by MIT OCW.

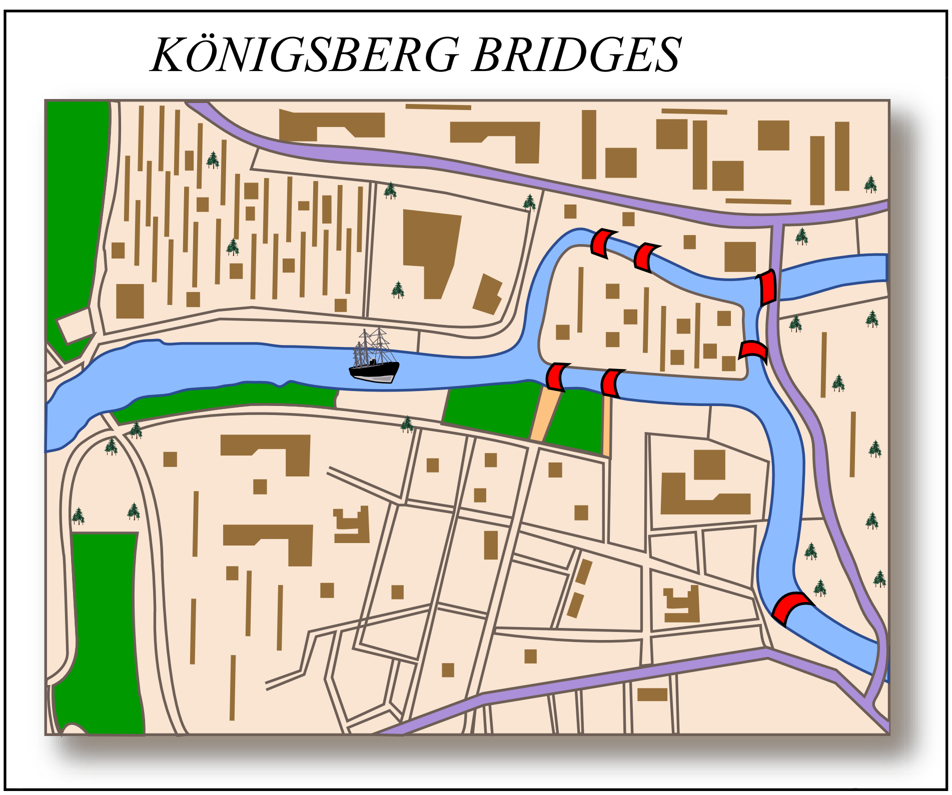
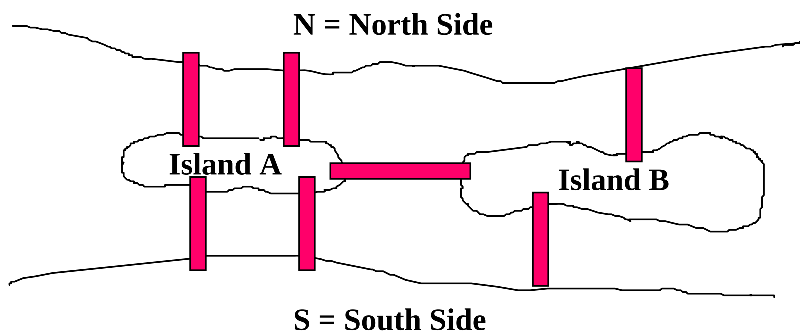


Image by MIT OCW.

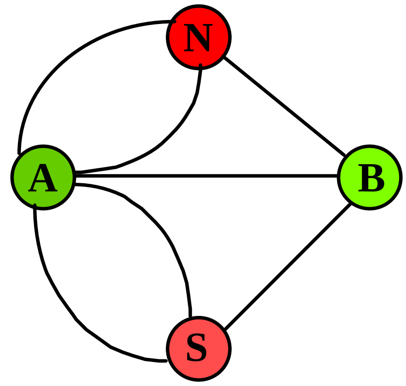
Euler's famous "test problem": the parade route

The Seven Bridges of Konigsberg



… reduced to a network problem

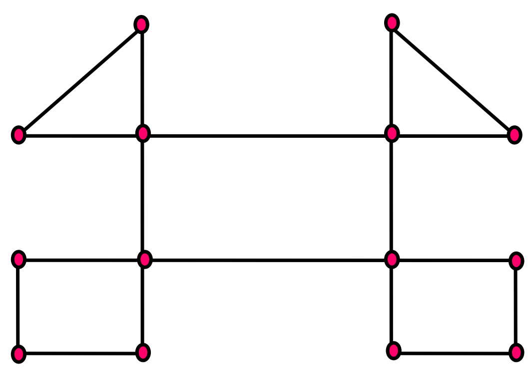
Seven Bridges of Konigsberg as a Network



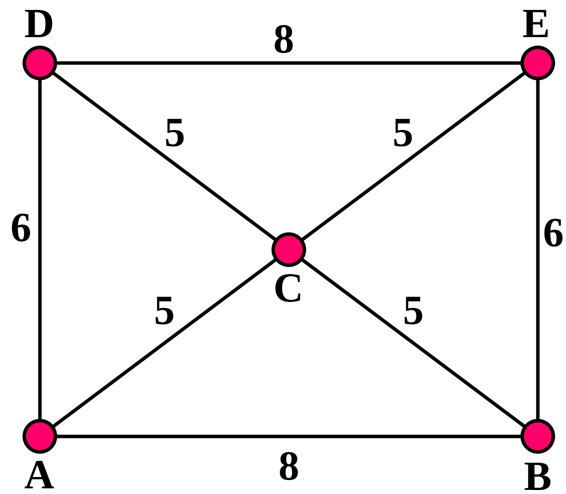
Drawing an Euler Tour

- It is easy to draw manually an Euler tour on a network that has one. Just do not traverse an "isthmus", i.e., an edge whose erasure will divide the *yet uncovered* part of the network into two separate, non-empty sub-networks.

An Easy Chinese Postman Problem



CPP Example



The CPP Algorithm (Undirected Graph)

- **BASIC IDEA: Take the given graph, G(N, A), and add "dummy" edges to it, until G has no odd degree nodes. In adding edges, try to add as little length as possible to G.**

STEP 1: Identify all *m* nodes of odd degree on G(N, A). [Remember *m* is even.]

STEP 2: Find the *minimum-cost, pairwise matching* of the odd-degree nodes. [Apply the "non-bipartite matching" algorithm (a.k.a. "flower and blossom" of Ellis and Johnson (1972) – see Chapter 12 of Ahuja, Magnanti and Orlin.]

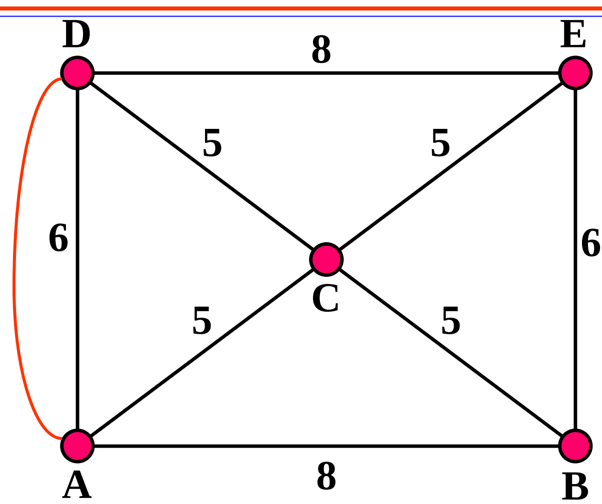
The CPP Algorithm (Undirected Graph) [continued]

STEP 3: Modify G(N, A) by adding to it the set, M, of (dummy) edges corresponding to the minimum-cost pair-wise matching found in STEP 2. Call this augmented graph G'.

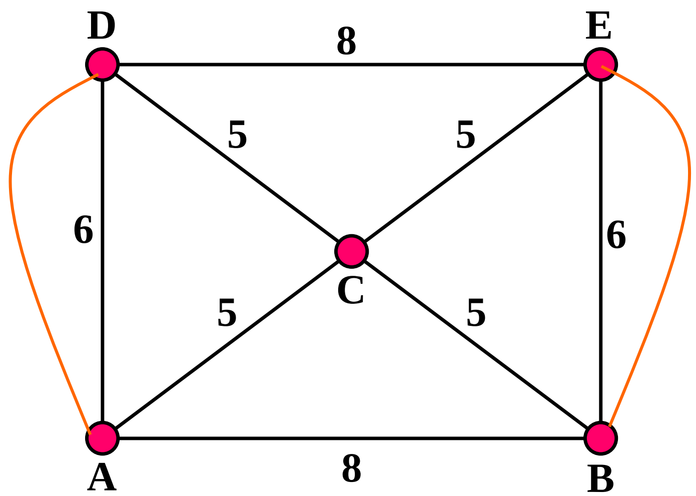
$$[G'(N, A \cup M)]$$

STEP 4: Find an Euler tour on G'. This tour is a solution to the CPP.

CPP Example (2)



CPP Example (3)



The Solution

- Pair-wise matches:

1. {A-D, E-B}, “cost” = 12
2. {A-B, D-E}, “cost” = 16
3. {A-E, B-D}, “cost” = 20

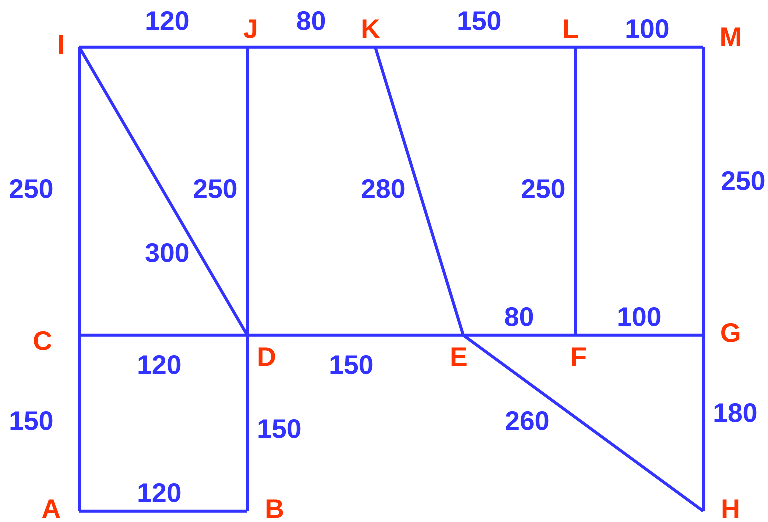
- Select “1”.
- Total CPP tour length = 48 + 12 = 60
- A tour: {A, B, C, A, D, C, E, B, E, D, A}

Number of Matches

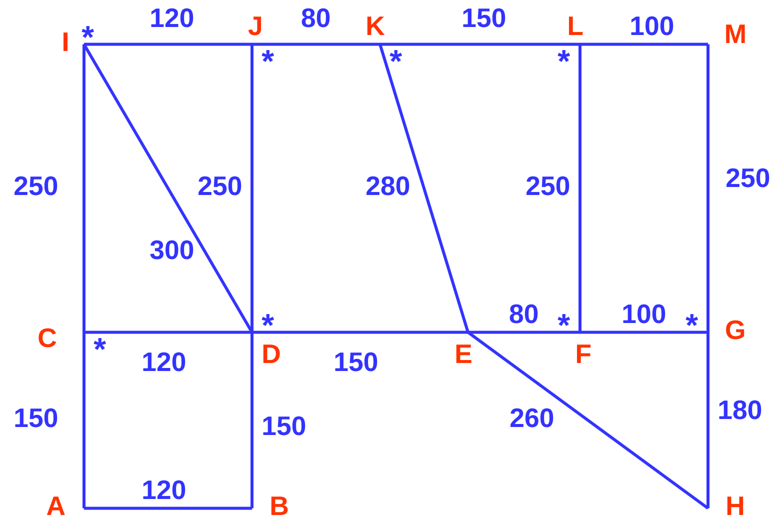
- Given *m* odd-degree nodes, the number of possible pair-wise matches is:

$$(m-1)\cdot(m-3)\cdot\ldots\ldots\cdot 3\cdot 1=\prod_{i=1}^{m/2}(2i-1)$$

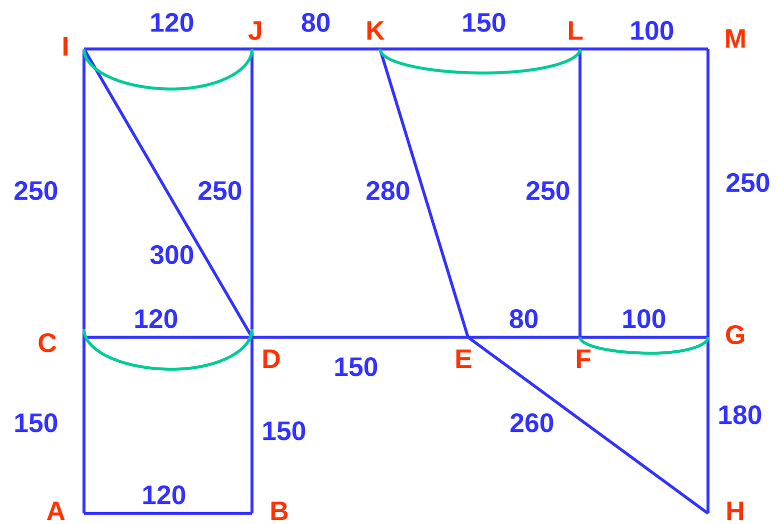
Minimize $\sum_{(i,j)\in A} n(i,j)\cdot\ell(i,j)$ [*n(i, j)* is the no. of times (i, j) is "covered"]



3,140 units of total length; 8 odd-degree nodes; 105 possible pair-wise matching combinations



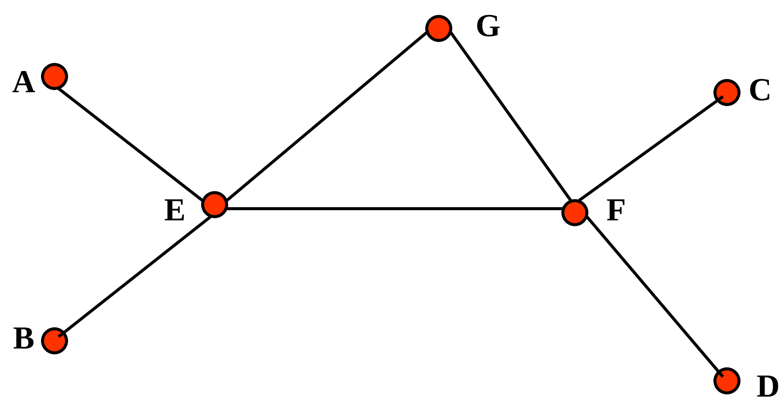
Optimal pair-wise matching can be found by inspection; 490 dummy edge units (double-covered); optimal CPP tour has length of 3,830 units



Solving Manually on a Graph

- Given a good “map”, it is possible to solve manually, to near-optimality, large CPPs on planar graphs.
- KEY OBSERVATION: In a minimum-cost, pair-wise matching of the odd degree nodes, no two shortest paths in the matching can have any edges in common.
- IMPLICATIONS:
  - _ Eliminate large no. of potential matches
  - _ Search only in “neighborhood” of each odd-degree node

Solving Manually (2)



Related CPP Problems

- CPP on directed graphs can also be solved efficiently (in polynomial time) [Problem 6.6 in L+O]
- CPP on mixed graph is a "hard" problem [Papadimitriou, 1976]
- Many variations and applications:
  - Snow plowing
  - Street sweeping
  - Mail delivery => "multi-postmen"
  - CPP with time windows
  - Rural CPP

Applications

- Each of these problem types has been greatly refined and expanded over the years
- Each can be implemented via computer in complex operating environments
- The Post office, FedEx, truckers, even bicycled couriers use these techniques