CISC 1100: Structures of Computer Science

Chapter 9
Graphs

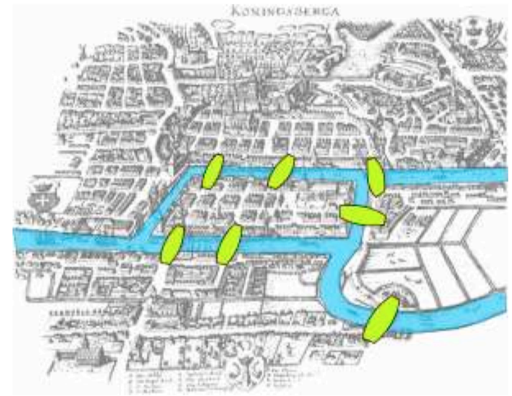
Arthur G. Werschulz

Fordham University Department of Computer and Information Sciences

Summer, 2016

Introduction

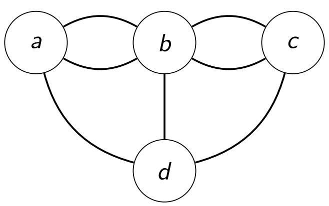
The city of Königsberg:



Can one "einen Spaziergang machen" that crosses each bridge in Königsberg exactly once?

Introduction (cont'd)

Get rid of all the "non-essentials", get a (multi)graph:



Introduction (cont'd)

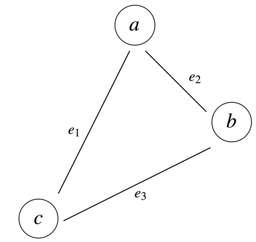
- Situations we can visualize using graphs
  - People in a social network
  - Cities in a country
  - Jobs in a "to-do" list
  - Electrical connections
  - The Internet
- Questions we can ask:
  - Can we visit every vertex and end up where we started?
  - Is there any vertex we cannot reach from other places?
  - What is the shortest distance between two vertices?
  - How to connect all vertices using the fewest number of edges? the minimal cost?

Outline

- Graph notation
- Euler trails and circuits
- Weighted graphs
- Minimum spanning trees
- Matrix notation for graphs

Graph notation: vertices and edges

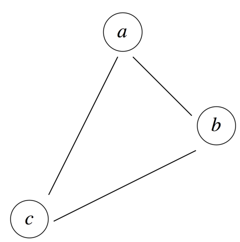
- Graph $G = (V, E)$
  - V: set of *vertices*
  - E: set of *edges*
  - $\{v, w\} \in E$ means that $v, w \in V$ are *connected*
- For the graph



  - $V = \{a, b, c\}$,
  - $E = \{e_1, e_2, e_3\}$.

Graph notation: vertices and edges (cont'd)

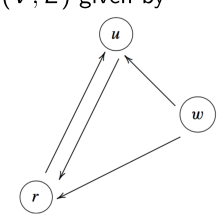
- Sometimes we simply indicate edges by the vertices they connect.
- For the graph



  - $V = \{a, b, c\}$,
  - $E = \{\{a, b\}, \{b, c\}, \{a, c\}\}$.

Graph notation: directed and undirected graphs

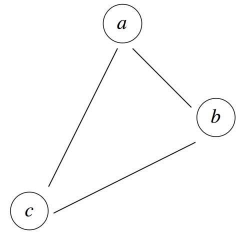
- $G = (V, E)$ is a *directed graph* (or *digraph*) if the edge from v_1 to v_2 can only be traversed in that direction.
- The graphs in previous examples were *undirected*.
- For an undirected graph, edge connecting distinct $v, w \in V$ is $\{v, w\}$.
- For a directed graph, edge connecting $v, w \in V$ is (v, w).
- For the graph $G = (V, E)$ given by



  - $V = \{u, r, w\}$,
  - $E = \{(r, u), (u, r), (w, r), (w, u)\}$.

Graph notation: complete graphs

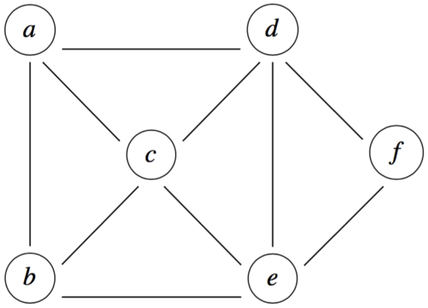
- A graph is *complete* if all possible edges are present.
- An undirected graph $G = (V, E)$ is complete if $\{v, w\} \in E$ for any distinct $v, w \in V$. The graph



  is complete.
  An undirected graph with n vertices has $\frac{1}{2}n(n-1)$ edges.
- A directed graph $G = (V, E)$ is complete if $(v, w) \in E$ for any distinct $v, w \in V$, i.e.if $E = V \times V$.
  A directed graph with n vertices has n^2 edges.

Graph notation

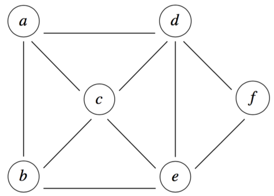
For the graph



- How many vertices are there? Six.
- How many edges? Ten
- Directed or undirected? Undirected
- Is this graph complete? No.

Graph notation

Let $G = (V, E)$. If $V' \subseteq V$ and $E' \subseteq E$, then $G' = (V', E')$ is a *subgraph* of G.
Consider once again the graph



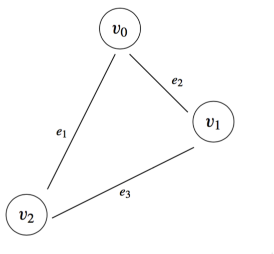
Find the largest n such that this graph has a complete subgraph with n vertices.
The subgraph with vertex set $\{a, b, c\}$ is complete.
There are no 4-element vertex sets yielding a complete subgraph.
So $n = 3$.

Euler trails and circuits

- A *walk* in a graph $G = (V, E)$ is a sequence of vertices $v_0, v_1, \ldots, v_n \in V$ and edges $e_1, e_2, \ldots, e_n \in E$, where each e_i is an edge connecting v_{i-1} and v_i.
- A *trail* is a walk in which no edge is traversed more than once.
- A *path* is a walk in which no vertex is traversed more than once.
- A *circuit* is a trail that begins and ends at the same vertex.
- A *cycle* is a circuit in which the start vertex is the *only* vertex appearing more than once.

Euler trails and circuits (cont'd)

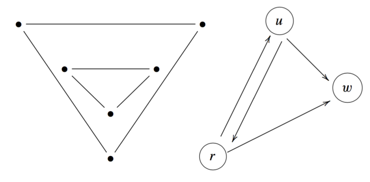
For the graph



- How many trails are there from v_0 to v_2?
  Two: v_0, e_2, v_1, e_3, v_2 and v_0, e_1, v_2.
- How many circuits start at v_0?
  Only one: $v_0, e_2, v_1, e_3, v_2, e_1, v_0$
- How many cycles are there?
  Infinitely many: repeat the circuit above over and over.

Euler trails and circuits (cont'd)

A graph is *connected* if there is a walk from any vertex to any other vertex. Which of these graphs is connected?



Neither of them!

Euler trails and circuits (cont'd)

- An *Euler trail* is a trail that includes every edge in the graph exactly once.
- An *Euler circuit* is a circuit that includes every edge in the graph exactly once.
- When does a graph (or multigraph) have an Euler trail? an Euler circuit?

Euler trails and circuits (cont'd)

(Only covering undirected graphs here; digraphs are a bit more complicated.)

- Every vertex in an Euler circuit must have an entry edge and an exit edge.
- The *degree* of a vertex is the number of edges that it has.
- A vertex is *evern* or *odd* if its degree is even or odd, respectively.
- All the vertices in an Euler circuit must be even (or else you'd be stuck at a vertex).
- So if a graph G has any odd vertices, then G cannot contain an Euler circuit.
- The converse is also true (but we won't prove it here).

Euler trails and circuits (cont'd)

- What about Euler trails?
- Suppose $G = (V, E)$ has Euler trail that's not a circuit, which means that there must be some odd vertices.
- If you add up the degrees of all the vertices, you get $2|E|$.
- Hence the number of odd vertices must be an even number.
- Furthermore, an odd vertex must be an endpoint of a non-circuit Euler trail.
- So there must be exactly two odd vertices.
- Add an extra "artificial" edge between them, getting a new graph.
- Now all the vertices in the new graph are even.
- Hence the new graph has an Euler circuit.
- Remove the artificial edge from the Euler circuit.
- You now have an Euler trail, whose terminal vertices are the two odd vertices.

Euler trails and circuits (cont'd)

Summarizing these results:
Let $G = (V, E)$ be a graph.

- If every vertex in V is even, then G has an Euler circuit.
- If exactly two vertices $v, w \in V$ are odd, then G has an Euler trail, starting at v and ending at w. Moreover, G does not have an Euler circuit.
- Otherwise, G has neither an Euler trail nor an Euler circuit.

Now suppose we change "edge" to "vertex", i.e., we look for a path that visits each *vertex* exactly once, a *Hamiltonian* (perhaps more properly, a *Rudrata*) circuit or trail.
Note the following:

- We can solve Euler circuit/trail problem in polynomial time.
- However:
  - No polynomial-time algorithm exists to solve the Hamiltonian circuit/trail problem in polynomial time, but
  - Nobody has shown that this problem cannot be solved in polynomial time.
  - This problem is NP-complete.

Weighted graphs

We sometimes label edges of a graph by a number:

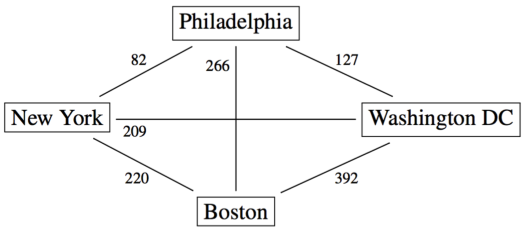
- Mileage between cities.
- Cost of sending a message between cell towers.
- Time to send an internet packet between two hosts.

We call this number a *weight*.
A graph, all of whose edges have weihts, is called a *weighted graph*.
Let $G = (V, E)$ be a weighted graph, and let w_e denote the weight associated with the edge $e \in E$. Then the total *weight* w_G of G is given by

$$w_G = \sum_{e \in E} w_e.$$

Weighted graphs (cont'd)

For example, let G be the weighted graph



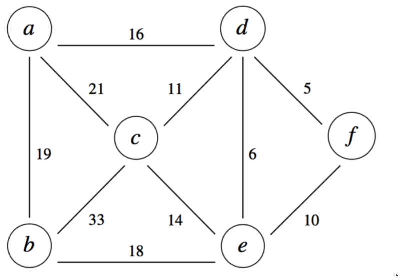
Then $w_G = 82 + 127 + 209 + 220 + 266 + 392 = 1296$. The minimal weight of a trip from New York to Washington, DC is

$$\min\{82 + 127, 209, 220 + 392\} = \min\{209, 209, 612\} = 209$$

and is given either by a direct trip, or a trip through Philadelphia.

Minimum spanning trees

Suppose a building has rooms, and the cost of connecting them by fiber-optic cable is given by the following graph:

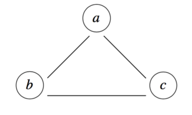


What's the minimal-cost netwrok that connects all the rooms?

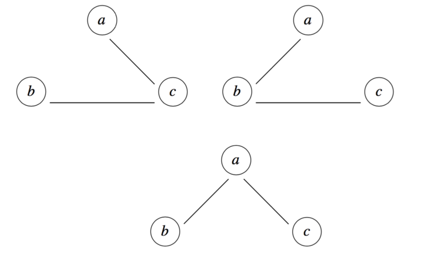
Minimum spanning trees (cont'd)

Let $G = (V, E)$ be a connected graph. A subgraph $T = (V, E')$ is a *spanning tree* for G is T is connected and acyclic.

Example: What are the spanning trees for the graph



Solution: There are three spanning trees, given by

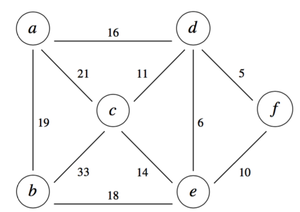


Minimum spanning tree (cont'd)

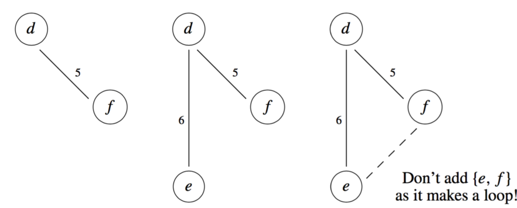
- *Minimal spanning tree* (MST) of a graph is a spanning tree having minimal weight among all spanning trees.
- Exhaustive listing: takes too long!!
- *Prim's algorithm* is far more efficient:
  1. Sort the edges by increasing weight.
  2. Working from smallest to largest, add each edge to the MST iff it doesn't make a cycle.
- How efficient?
  - Step 1 can be done with cost $O(|E| \log |E|)$.
  - Step 2 only requires $\min\{|V| - 1, |E|\}$ iterations.

Minimum spanning trees (cont'd)

Example: Find an MST for

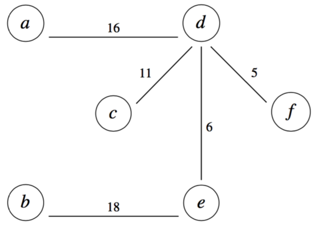


Solution: Order the edges by increasing weight. Now add them one by one, making sure to not introduce any cycles:



Minimum spanning trees (cont'd)

Continuing on, we find that



is an MST for our graph