EECS 203 Lecture 19

Graphs
Admin stuffs

- Last homework due tomorrow

- Office hour changes starting Friday (also in Piazza)
  - Friday 6/17: 2-5 Mark in his office.
  - Sunday 6/19: 2-5 will be Jasmine in the UGLI
  - Monday 6/20: 10-12: Mark in his office.
  - Monday 6/20: 5-7 Emily in the UGLI.
  - Tuesday 6/21: 1-3 Mark in his office.
  - Wednesday 6/22: 10-12 Emily in the Beyster Learning Center.
  - Wednesday 6/22: 1:30-3:00 Mark in his office.
  - Thursday 6/23: 1:30-3:00 Jasmine in the Beyster Learning Center.

- Discussion is still on for Thursday and Friday.
Today

- Dijkstra’s algorithm
  - Using slides from
    - http://math.ucsd.edu/~fan/teach/202/
Dijkstra's Algorithm

Dijkstra's algorithm.
- Maintain a set of explored nodes $S$ for which we have determined the shortest path distance $d(u)$ from $s$ to $u$.
- Initialize $S = \{ s \}$, $d(s) = 0$.
- Repeatedly choose an unexplored node $v$ which minimizes

$$\pi(v) = \min_{e = (u, v) : u \in S} d(u) + \ell_e,$$

add $v$ to $S$, and set $d(v) = \pi(v)$.

shortest path to some $u$ in explored part, followed by a single edge $(u, v)$
Dijkstra's Algorithm: Proof of Correctness

Invariant. For each node \( u \in S \), \( d(u) \) is the length of the shortest \( s-u \) path.

Pf. (by induction on \( |S| \))

Base case: \( |S| = 1 \) is trivial.

Inductive hypothesis: Assume true for \( |S| = k \geq 1 \).

- Let \( v \) be next node added to \( S \), and let \( u-v \) be the chosen edge.
- The shortest \( s-u \) path plus \( (u, v) \) is an \( s-v \) path of length \( \pi(v) \).
- Consider any \( s-v \) path \( P \). We'll see that it's no shorter than \( \pi(v) \).
- Let \( x-y \) be the first edge in \( P \) that leaves \( S \), and let \( P' \) be the subpath to \( x \).
- \( P \) is already too long as soon as it leaves \( S \).

\[
\ell(P) \geq \ell(P') + \ell(x,y) \geq d(x) + \ell(x,y) \geq \pi(y) \geq \pi(v)
\]

- \( \ell \): nonnegative weights
- \( \ell(P') + \ell(x,y) \): inductive hypothesis
- \( d(x) + \ell(x,y) \): defn of \( \pi(y) \)
- \( \pi(v) \): Dijkstra chose \( v \) instead of \( \gamma \)
Dijkstra's Shortest Path Algorithm

Find shortest path from s to t.
Dijkstra's Shortest Path Algorithm

\[ S = \{ \} \]
\[ PQ = \{ s, 2, 3, 4, 5, 6, 7, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ \} \]
\[ PQ = \{ s, 2, 3, 4, 5, 6, 7, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s \} \]
\[ PQ = \{ 2, 3, 4, 5, 6, 7, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s \} \]
\[ PQ = \{ 2, 3, 4, 5, 6, 7, t \} \]

Diagram showing a network with nodes labeled 0, 2, 3, 4, 5, 6, 7, and t. The algorithm is in progress, with nodes being added to the set S and the priority queue PQ. The distance label for node 7 is 15.
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2 \} \]
\[ PQ = \{ 3, 4, 5, 6, 7, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2 \} \]
\[ PQ = \{ 3, 4, 5, 6, 7, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2 \} \]
\[ PQ = \{ 3, 4, 5, 6, 7, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 6 \} \]

\[ PQ = \{ 3, 4, 5, 7, t \} \]
Dijkstra's Shortest Path Algorithm

$S = \{ s, 2, 6 \}$
$PQ = \{ 3, 4, 5, 7, t \}$
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 6, 7 \} \]
\[ PQ = \{ 3, 4, 5, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 6, 7 \} \]
\[ PQ = \{ 3, 4, 5, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{s, 2, 3, 6, 7\} \]
\[ PQ = \{4, 5, \dagger\} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 3, 6, 7 \} \]
\[ PQ = \{ 4, 5, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 3, 5, 6, 7 \} \]
\[ PQ = \{ 4, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 3, 5, 6, 7 \} \]
\[ PQ = \{ 4, t \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 3, 4, 5, 6, 7 \} \]
\[ PQ = \{ + \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 3, 4, 5, 6, 7 \} \]
\[ PQ = \{ + \} \]
Dijkstra's Shortest Path Algorithm

\[ S = \{ s, 2, 3, 4, 5, 6, 7, t \} \]

\[ PQ = \{ \} \]
Dijkstra's Shortest Path Algorithm

$S = \{ s, 2, 3, 4, 5, 6, 7, t \}$

$PQ = \{ \}$
Dijkstra’s Algorithm animated

Dijkstra’s algorithm

www.combinatorica.com
Let’s try an induction proof on a graph

- First we’ll define what it means for a graph to be connected.
  - Then we’ll show that in any connected graph there are at least two vertices you can remove from the graph (along with its incident edges) that leave the graph still connected.

- We’ll use strong induction.

- First let’s (re)define some terms
Graphs

A **graph** $G = (V,E)$ consists of
- a non-empty set $V$ of **vertices** (or nodes),
- and a set $E$ of **edges**.

Each edge is associated with two vertices (possibly equal), which are its endpoints.

A **path** from $u$ to $v$ of length $n$ is:
- a sequence of edges $e_1, e_2, \ldots, e_n$ in $E$, such that
- there is a sequence of vertices $u = v_0, v_1, \ldots, v_n = v$
  such that each $e_i$ has endpoints $v_{i-1}$ and $v_i$.

A path is a **circuit** when $u = v$.

A graph is **connected** when there exists a path from every vertex to every other vertex.
Theorem 2. Every connected graph $G$ with $|V(G)| \geq 2$ has at least two vertices $x_1, x_2$ so that $G - x_i$ is connected for $i = 1, 2$.

Proof: We proceed by induction on $|V(G)|$. As a base case, observe that if $G$ is a connected graph with $|V(G)| = 2$, then both vertices of $G$ satisfy the required conclusion. For the inductive step, let $G$ be a connected graph with $|V(G)| \geq 2$ and assume that the theorem holds for every graph with $< |V(G)|$ vertices. If $G - x$ is connected for every vertex $x \in V(G)$, then we are done, so we may assume this is not so, and choose $x \in V(G)$ so that $G - x$ has components $H_1, H_2, \ldots, H_m$ where $m \geq 2$. For every $1 \leq i \leq m$ let $H'_i$ be the graph obtained from $H_i$ by adding back the vertex $x$ and all edges with one end $x$ and the other end in $V(H_i)$. So every $H'_i$ is a connected graph with at least two vertices. Furthermore, $|V(H'_i)| < |V(G)|$, so by induction, $H'_i$ must have at least one vertex $x_i \neq x$ so that $H'_i - x_i$ is connected. It then follows that $G - x_i$ is connected. Since we have such an $x_i$ for every component (and at least two components), this completes the proof. \qed


There are a few nice observations about the proof there as well as a few nice induction examples.
Graphs

A **graph** $G = (V,E)$ consists of

- a non-empty set $V$ of **vertices** (or nodes),
- and a set $E$ of **edges**.

Each edge is associated with two vertices (possibly equal), which are its endpoints.

A **path** from $u$ to $v$ of length $n$ is:

- a sequence of edges $e_1, e_2, \ldots e_n$ in $E$, such that
- there is a sequence of vertices $u = v_0$, $v_1$, $\ldots v_n = v$

such that each $e_i$ has endpoints $v_{i-1}$ and $v_i$.

A path is a **circuit** when $u = v$.

A graph is **connected** when there exists a path from every vertex to every other vertex.
Paths and Circuits

Given a graph $G = (V,E)$:

Is there a path/circuit that crosses each edge in $E$ exactly once?

If so, $G$ is an Eulerian graph, and you have an Eulerian path, or an Eulerian circuit.

Is there a path/circuit that visits each vertex in $V$ exactly once?

If so, $G$ is a Hamiltonian graph, and you have a Hamiltonian path or circuit.
Leonhard Euler lived in Königsberg

He liked to take walks. A famous local puzzle was whether you could take a walk that would cross each bridge exactly once.

Euler solved this problem (in 1736) and thus founded graph theory.
The Graph Abstraction

In the physical world:
- Each bridge connects exactly two land-masses.
- Each land-mass may have any number of bridges.

Suggests a graph abstraction:
- Represent a bridge by an edge in the graph.
- Represent a land-mass by a vertex.
Is there an Euler circuit/path?

Circuit

Path

Not Traversable
Does G have an Euler Path/Circuit?

Theorem: A connected multigraph has an Euler path iff it has exactly **zero or two vertices** of odd degree.

Why?

What if it has only one?

When does it have an Euler circuit?
Examples

Do these graphs have Euler paths/circuits?

(A) Yes, it has a circuit.
(B) Yes, it has a path (but no circuit).
(C) No, it has neither path nor circuit.
In his article on the Königsberg bridges, Euler considered another bridge problem, which is illustrated below.

Two islands, A and B, are surrounded by water that leads to four rivers. Fifteen bridges cross the rivers and the water surrounding the islands. Is it possible to make a trip that crosses each bridge exactly once?

(A) Yes          (B) No
Applications of Euler Paths

Planning routes through graphs that provide efficient coverage of the edges in the graph, without multiple traversals.

- Postal delivery routes
- Snowplowing routes
- Testing network connections
  - Utility transmission network
  - Communication network
Hamiltonian Paths and Circuits

Given a graph, is there a path that passes through each vertex in the graph exactly once? (aka a Hamiltonian path)

(a) dodecahedron  
(b) Isomorphic graph  
(solution)
Hamilton circuit

Do these graphs have Hamilton circuits?

(A) Yes, both.  
(B) G does, H doesn’t. 
(C) G doesn’t, H does.  
(D) No, neither.
Deciding whether a graph is **Eulerian** is a simple examination of the degrees of the vertices.

Simple linear-time algorithms exist for finding the path or circuit.

Deciding whether a graph is **Hamiltonian** is, in general, much more difficult ("NP complete").

Finding a Hamiltonian path or circuit is equally hard.
A **weighted graph** has numbers (weights) assigned to each edge.

The **length** of a path is the sum of the weights in the path.
Traveling Salesperson’s Problem

What is the shortest route in a given map for a salesperson to visit every city exactly once and return home at the end?

Mathematically:
Given an graph with weighted edges, what is the Hamiltonian circuit of least weight?

Applies to both directed and undirected graphs.
The Traveling Salesman Problem

**Problem:** Given a graph $G$, find the shortest possible length for a Hamilton circuit.

What is the solution to TSP for the following graph?

(A) 16  (B) 17  (C) 18  (D) 19  (E) 20
Traveling Salesperson’s Problem

The TSP is NP Complete: intractable in general. Very large instances have been solved with heuristics. An instance with 13,509 cities, solved in 2003.
Applications

The Traveling Salesperson’s Problem and the Hamilton Path/Circuit Problem have many applications.

- Planning and logistics (of course!)
- Manufacture of microchips
- DNA sequencing