Graphs

Part I:
Introduction

Motivation for Graph Theory

Many, Many Applications

Fundamental Data Structures

Neat Algorithms

Engaging Theory

Novel Mathematics

Types of Graphs

- Simple Graph
- Multigraph
- Directed Graph
- Weighted Graph
- Connected
- Planar
- Bipartite
- Complete

Examples and Animations

http://oneweb.utc.edu/~Christopher-Mawata/petersen/

Representations and Special Graphs

Representations

- Diagrams
- Matrices

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson7.htm

Special Graphs

- $C_n =$ Cycle
- $W_n =$ Wheel
- $K_n =$ Complete Graph

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson11.htm

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson4.htm

Examples

- Web Graph
- Acquaintance Graph
- Telephone Graph
- Road Graph -- Robotics
- Concurrency Graph -- Programming
- Tournament Graph

Applications

- Path Planning (Robotics)
- Shortest Path
  - -- Traveling Salesman Problem
  - -- Cost Minimizing Problems
  - -- Time Minimizing Problems
- Scheduling (Graph Coloring)
- DNA Sequencing

Handshaking Theorem -- Connected Graphs

Theorem: $$2e = \sum_{v \in V} \deg(v)$$

-- e = # edges

-- $\deg(v)$ = # edges incident on the vertex v

Proof: Each edges is counted twice on the RHS, since each edge joins two vertices.

Web Page

http://oneweb.utc.edu/~ChristopherMawata/petersen/lesson2.htm

Consequences

1. It is impossible to connect 15 computers so that each computer is connected to exactly 7 other computers.

2. A country with exactly 3 roads out of every city cannot have 1000 roads.

Graph Isomorphisms and Graph Invariants

Problem

- When are two graphs G, H the same?
- Hard to tell from diagrams.

Graph Isomorphisms

- G and H are said to be isomorphic if we can deform G into H.
- G and H are said to be isomorphic if there is function $f : G \to H$ such that:

  -- f maps vertices to vertices

  -- f maps edges to edges

  -- each vertex of H corresponds to a unique vertex of G

  -- each edge of H corresponds to a unique edge of G

  -- if e connects u and v, then $f(e)$ connects $f(u)$ and $f(v)$

Examples and Animations

Graph Isomorphisms

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson3.htm

Graph Invariants

Definition

- A graph invariant is a number or property that is the same for all isomorphic graphs.

Examples

- Number of Vertices and Edges
- Number of Paths between Vertices
- Vertex Degrees
- Circuit Length
- Connectedness
- Chromatic Number

Applications

- Determining that two graphs are not isomorphic

Examples and Animations

Graph Coloring

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson8.htm

Part II:

Navigation

Paths

- Simple -- Contains Each Edge at Most Once

- Circuit -- Starts and Ends at Same Vertex

- Euler Path/Circuit -- Simple Path Contains Every Edge (Edges, Easy)

- Hamiltonian Path/Circuit -- Simple Path Contains Every Vertex (Vertices, Hard)

Euler Paths

Applications

- Layout of Circuits
- DNA Sequencing

Theorems

- Euler Circuit Theorem: Every vertex has even degree.
- Euler Path Theorem: Exactly two vertices of odd degree.

Euler Paths

Google Streetview

- Eulerian path to drive for taking pictures

Routing

- Snow plow routes

DNA Sequencing

- Shortest sequence of nucleotides representing a gene.

<u>DNA Sequencing</u>

Biochemistry (A, G, C, T)

- Find all nucleotides of a fixed small length N in a gene.

Graph Theory (Reassemble the entire gene)

- Vertices = Strands of DNA of Fixed Length $N-1$
- Edges = Connect two vertices u,v if there is a Strand of DNA of Length N whose first $N-1$ nucleotides correspond to u and the last nucleotides correspond to v
- Construct an Euler Path to reassemble the gene

DNA Sequencing

Example

- Sequence of Nucleotides -- *AGT, TAG, GTA*
- Vertices -- AG, GT, TA
- Edges -- AGT, GTA, TAG
- Reconstructed Gene

  -- Juxtaposition = *AGTTAGGTA or AGTAGTA*

  -- Euler Path = *AGTAG*

Euler Animations

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson12.htm

http://www.cut-the-knot.org/Curriculum/Combinatorics/GraphPractice.shtml

http://www.cut-the-knot.org/Curriculum/Combinatorics/FleuryAlgorithm.shtml

http://cauchy.math.okstate.edu/~wrightd/1493/euler/index.html

http://www.cut-the-knot.org/Curriculum/Combinatorics/GraphPractice.shtml

Euler Circuit Theorem

Theorem

Euler Circuit Exists $\Leftrightarrow$ Every Vertex has Even Degree.

Proof

$\Rightarrow$: Assume Euler Circuit Exists.

- Let v be any vertex.
- For every entry, there must be an exit, so v has even degree.
- (Note the first vertex is special, but the statement is still true.)

$\Leftarrow$: Assume Every Vertex is Even.

- Start anywhere and Go as far as you can.
- You must return to start vertex, since every vertex is even.
- If all edges traversed, then done.
- Otherwise, remove traversed edges, and pick an unused edge connected to a vertex in the first path.
- Again go as far as you can and form another circuit.
- Splice circuits together.
- Continue until all edges are traversed.

Euler Path Theorem

Theorem

Euler Path Exists $\Leftrightarrow$ Exactly Two Vertices have Odd Degree.

Proof

$\Rightarrow$: Assume Euler Path Exists.

- Let v be any vertex, except first and last.
- For every entry, there must be an exit, so v has even degree.
- The first vertex has an exit with no entry.
- The last vertex has an entry with no exit,
- Hence there are exactly two vertices with odd degree.

$\Leftarrow$: Assume Exactly Two Vertices have Odd Degree.

- Add an edge between the two odd vertices a,b.
- Now every vertex has even degree.
- Therefore an Euler circuit exists, starting with the new edge exiting a.
- Therefore an Euler path exists starting from b and ending at a.

Hamilton Circuits

Examples

- The complete graph K_n has many Hamilton circuits.
- The more edges in the graph, the more likely the graph has a Hamilton circuit.

Applications

- Traveling Salesman Problem

  -- Shortest Hamilton Circuit in K_n

  -- FED EX, Garbage Collection

<u>Hamilton Circuits</u>

Graph

- Each House is a Vertex
- Each Road Segment is an Edge

Hamiltonian Paths

- Mail Routes
- Garbage Pickup

Hamilton Animations

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson12b.htm

Hamilton Circuits -- Theorems

Ore's Theorem (Necessary Conditions)

- $\deg(u)+\deg(v)\geq \#G=n$ for all nonadjacent u,v $\Rightarrow$ Hamilton Circuit Exists

Proof: Homework

NP-Completeness Theorem

- Determining if a graph has a Hamilton Circuit is an NP-Complete problem.

  -- The existence of a Hamilton Circuit can be verified in Polynomial Time.

  -- If a polynomial time algorithm exists that can determine for every graph whether or not there exists a Hamilton Circuit, then every problem that can be verified in polynomial time can be solved in polynomial time!

Proof: Comp 482

Comparisons

Euler Circuits

- Easy
- Linear Time Algorithm -- $O(n)$

Hamilton Circuits

- Hard
- NP-Complete problem -- $O(2^n)$

Shortest Paths

Problem

- Find the Shortest Path between two arbitrary Vertices in a Weighted Graph.
- Typically all Weights are assumed to be Positive.

Applications

- Minimizing *Cost*, *Time*, *Distance* for Travel between Cities.
- Minimizing *Cost* or *Response Time* in a Computer Network.

Dijkstra's Shortest Path Algorithm

Problem

Find the Shortest Path in a Weighted Graph G from Vertex a to Vertex z, where all the Weights are assumed to be Positive.

Dijkstra's Algorithm

Base Case: $S_1 = \{a\}$

Recursion: $S_{k+1} = S_k \cup \{v\}$, where v is the vertex closest to S_k.

Terminate when $z \in S_k$

Proof

By Induction on k: S_k contains the shortest path from a to vertices in S_k.

Dijkstra Animations

http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/DijkstraApplet.html

http://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html

http://www.unf.edu/~wkloster/foundations/foundationsLinks.html

Trees

Tree

- A simple graph with no circuits.

Spanning Tree

- A tree containing every vertex of a simple graph G, that is also a subgraph of G.

Minimal Spanning Tree

- A spanning tree on a weighted graph with the smallest sum of weights.

Minimal Spanning Trees

Applications

- Minimizing Network Cost

Algorithms

- Prim's Algorithm
- Kruskal's Algorithm

Prim's Algorithm

Problem

Given a Weighted Graph G, find a Minimal Spanning Tree.

Prim's Algorithm (Greedy Algorithm)

Base Case:

- $T_1 = \{edge\ \ with\ \ smallest\ \ weight\}$

Recursion:

- $T_{k+1} = T_k \cup \{e\}$
- $e = edge\ \ with\ \ smallest\ \ weight\ \ connected\ \ to\ \ T_k\ \ not\ \ forming\ \ a\ \ circuit$

Termination Condition

- $k = n-1$

Proof

By Contradiction: T_k is a subtree of the minimal spanning tree.

Complexity

$O\big(e\,log(v)\big)$

Kruskal's Algorithm

Problem

Given a Weighted Graph G, find a Minimal Spanning Tree.

Kruskal's Algorithm (Greedy Algorithm)

Base Case:

- $T_1 = \{edge\ with\ smallest\ weight\}$

Recursion:

- $T_{k+1} = T_k \cup \{edge\ with\ smallest\ weight\ not\ forming\ a\ circuit\}$

Termination Condition

- $k = n - 1$.

Complexity

- $O\left(e\, log(e)\right)$

Animations

Prim's Algorithm

http://www.unf.edu/~wkloster/foundations/foundationsLinks.html

Kruskal's Algorithm

http://www.unf.edu/~wkloster/foundations/foundationsLinks.html

Planar Graphs

Definition

- A graph G is called *planar* if G can be drawn on the plane with no crossing edges.

Examples

- K_4 is planar
- K_5 and $K_{3,3}$ are not planar

Application

- Printed Circuits

Examples

http://www.personal.kent.edu/~rmuhamma/GraphTheory/MyGraphTheory/planarity.htm

<u>**Euler's Formula**</u>

Planar Graphs

- $v - e + r = 2$
  - -- v = # vertices,
  - -- e = # edges
  - -- r = # regions

Proof

By induction on the number of edges.

Base Case:

- One edge: $v = 2,\ e = 1,\ r = 1 \Rightarrow v - e + r = 2$

Inductive Hypothesis:

- Euler's formula is valid for planar graphs with n edges.
- Must show that Euler's formula is valid for planar graphs with $n + 1$ edges.
- Consider two cases:
  - i. Connect an edge to one vertex: $v \to v + 1,\quad e \to e + 1$
  - ii. Connect an edge to two vertices: $e \to e + 1,\quad r \to r + 1$
- In both cases $v - e + r$ does not change.

Euler's Formulas

Planar Graphs

- $v - e + r = 2$
  - -- v = # vertices,
  - -- e = # edges
  - -- r = # regions

Polyhedra

- $V - E + F - H = 2(C - G)$
  - -- V = # vertices,
  - -- E = # edges
  - -- F = # faces
  - -- H = # holes in faces
  - -- C = # connected components
  - -- G = # holes in the solid (genus).

Kuratowski's Theorem

Theorem

Every Non-Planar Graph contains either $K_{3,3}$ or K_5.

Proof

Hard

Graph Coloring

Graph Coloring

- An assignment of colors to the vertices of a graph so that no two adjacent vertices have the same color.

Chromatic Number

- The smallest number of colors needed to color a graph.

Examples

- K_n *requires n colors*
- $K_{m,n}$ *requires 2 colors*
- C_n *requires 2 colors if n is even and 3 colors if n is odd*
- W_n *requires 3 colors if n is even and 4 colors if n is odd*

Applications

- Avoiding Scheduling Conflicts

Graph Coloring Animations

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson8.htm

<u>**Four Color Theorem**</u>

Four Color Theorem

Four colors suffice to color any planar graph.

Proof of Four Color Theorem

Difficult. By Computer!

Graph Coloring Problem

Problem

- Find the chromatic number of an arbitrary graph.

Solution

- Backtracking (Later -- See Trees)

Complexity

- Best known algorithms take exponential time
  in the number of vertices of the graph.

Graph Coloring Animations

http://oneweb.utc.edu/~Christopher-Mawata/petersen/lesson8.htm