CSE 326: Data Structures

Topic #15: Cool Graphs n' Pretty Pictures

Ashish Sabharwal
Autumn, 2003

Today's Outline

- Admin
  - Project 3 in-progress checkin due tonight!
- **Graph Algorithms**
  - Representation
  - Applications
  - Topological sort

2

A Great Mathematician

- Number theory
- Numerical Analysis
- Graph Theory
- Physical sciences
- See the History of Mathematics biography on Euler:
  - http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Euler.html

Leonhard Euler 1707-1783

3

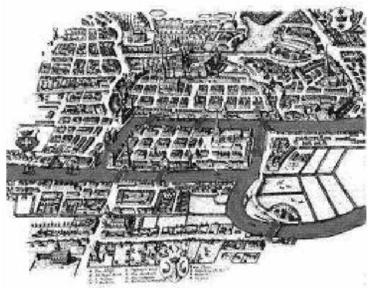
The Bridges of Königsberg

(now Kaliningrad, Russia)

Can we walk around Konigsberg, crossing each bridge *exactly once*?

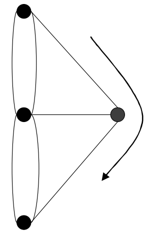
4

The (Graffitied) Bridges of Königsberg



5

The Bridges of Königsberg, Formally



- Each part of town is a vertex
- Each bridge is an edge
- **<u>Eulerian path</u>**: a path that visits each edge exactly once

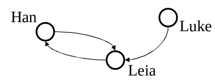
Does there exist an Eulerian path in the graph?

6

Graph… ADT?

- Not quite an ADT… operations not clear
- A formalism for representing relationships between objects

  Graph `G = (V,E)`
  - *Set of vertices*:
    $V = \{v_1, v_2, \ldots, v_n\}$
  - *Set of edges*:
    $E = \{e_1, e_2, \ldots, e_m\}$
    where each e_i connects two vertices (v_{i1}, v_{i2})

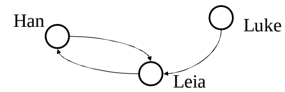



```
V = {Han, Leia, Luke}
E = {(Luke, Leia),
     (Han, Leia),
     (Leia, Han)}
```

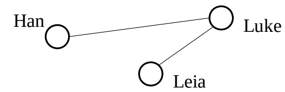
7

Graph Definitions

In *directed* graphs, edges have a specific direction:



In *undirected* graphs, they don't (edges are two-way):



v is *adjacent* to u if $(u,v) \in E$

8

More Definitions: Simple Paths and Cycles

A *simple path* repeats no vertices (except that the first can be the last):

p = {Seattle, Salt Lake City, San Francisco, Dallas}
p = {Seattle, Salt Lake City, Dallas, San Francisco, Seattle}

A *cycle* is a path that starts and ends at the same node:

p = {Seattle, Salt Lake City, Dallas, San Francisco, Seattle}
p = {Seattle, Salt Lake City, Seattle, San Francisco, Seattle}

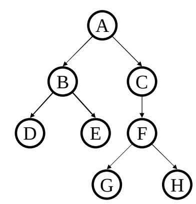
A *simple cycle* is a cycle that repeats no vertices except that the first vertex is also the last (in undirected graphs, no edge can be repeated)

9

Trees as Graphs

- Every tree is a graph!
- Not all graphs are trees!

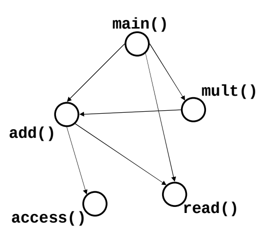
  A graph is a tree if
  - There are *no cycles* (directed or undirected)
  - There is a *path* from the root *to every node*



10

Directed Acyclic Graphs (DAGs)

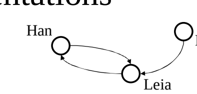
DAGs are directed graphs with no (directed) cycles.



Aside: *If program call-graph is a DAG, then all procedure calls can be in-lined*

11

Graph Representations



0. List of vertices + list of edges
1. 2-D matrix of vertices (marking edges in the cells)
   "adjacency matrix"
2. List of vertices each with a list of adjacent vertices
   "adjacency list"

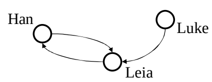
Vertices and edges may be labeled

Things we might want to do:
- iterate over vertices
- iterate over edges
- iterate over vertices adj. to a vertex
- check whether an edge exists

12

Representation 1: Adjacency Matrix

A **|V| x |V|** array in which an element **(u,v)** is true if and only if there is an edge from **u** to **v**



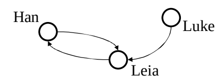
	Han	Luke	Leia
Han			
Luke			
Leia			

space requirements: *runtime:*

13

Representation 2: Adjacency List

A **|V|**-ary list (array) in which each entry stores a list (linked list) of all adjacent vertices

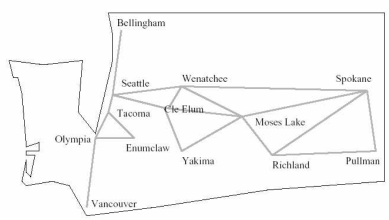


Han	
Luke	
Leia	

space requirements: *runtime:*

14

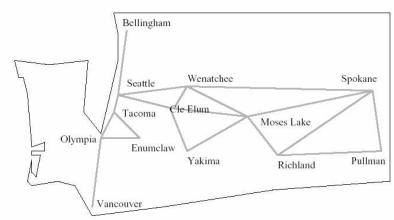
Some Applications: Moving Around Washington



What's the *shortest way* to get from Seattle to Pullman?
Edge labels:

15

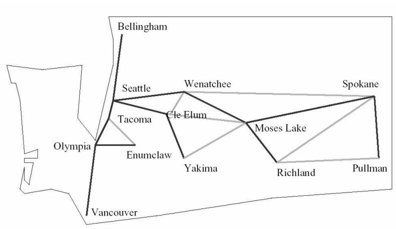
Some Applications: Moving Around Washington



What's the *fastest way* to get from Seattle to Pullman?
Edge labels:

16

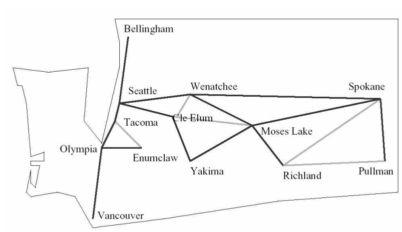
Some Applications: Communication in Washington



What's the *cheapest* inter-city public network?
Edge labels:

17

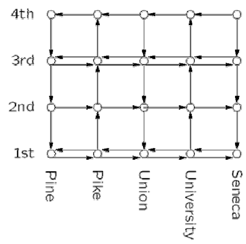
Some Applications: Reliability of Communication



If Wenatchee's phone exchange *goes down*, can Seattle still talk to Pullman?

18

Some Applications: Bus Routes in Downtown Seattle



If we're at 3rd and Pine, how can we get to 1st and University using Metro?

19

Food for Thought

Before next class, *think how you would efficiently solve each of the preceding problems!*

20

More Applications: Orderings and Dependencies

Okay, everybody, get up and stretch!
Pretend as if you were about to leave!!

What actions did you take?

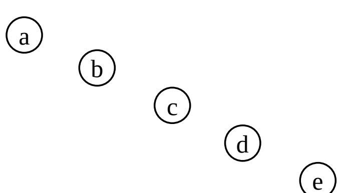
21

Ordering of Actions: Dependency Graph

Let's order the actions by what you did first:

22

Ordering on Graphs: Total and Partial

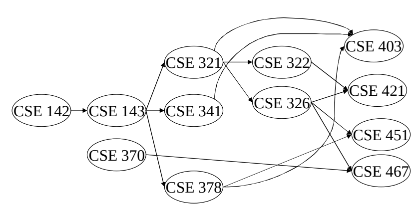


means A must go before B

23

Application: Topological Sort

Given a directed graph, **`G = (V,E)`**, output all the vertices in **`V`** such that no vertex is output before any other vertex with an edge to it.



Is the output unique?

24

Topological Sort: Take One

1. Label each vertex with its *in-degree* (# of inbound edges)
2. While there are vertices remaining
   a. Choose a vertex *v* of in-degree zero; output *v*
   b. Reduce the in-degree of all vertices adjacent to *v*
   c. Remove *v* from the list of vertices

Runtime:

Topological Sort: Take Two

1. Label each vertex with its in-degree
2. Initialize a queue *Q* to contain all in-degree zero vertices
3. While *Q* not empty
   a. *v* = *Q*.dequeue; output *v*
   b. Reduce the in-degree of all vertices adjacent to *v*
   c. If new in-degree of any such vertex *u* is zero *Q*.enqueue(*u*)

Note: could use a stack, list, set, box, … instead of a queue

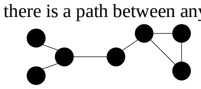
Runtime:

Graph Traversals

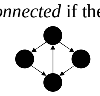
- Breadth-first search (and depth-first search) work for arbitrary (directed or undirected) graphs - not just mazes!
  - Must mark visited vertices so you do not go into an infinite loop!
- Either can be used to determine connectivity:
  - Is there a path between two given vertices?
  - Is the graph (weakly) connected?
- Which one:
  - Uses a queue?
  - Uses a stack?
  - Always finds the shortest path (for unweighted graphs)?

Graph Connectivity

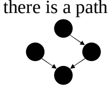
Undirected graphs are *connected* if there is a path between any two vertices



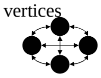
Directed graphs are *strongly connected* if there is a path from any one vertex to any other



Directed graphs are *weakly connected* if there is a path between any two vertices, *ignoring direction*



A *complete* graph has an edge between every pair of vertices



To Do

- Project 3: in-progress turnin tonight!
- Think of solutions to problems posed in today's lecture
- Read Chapter 9, sections 9.1-9.3