Introduction to Graphs

15-211
Fundamental Data Structures and Algorithms

Peter Lee

April 1, 2003

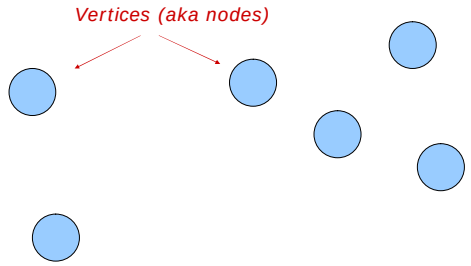
Announcements

Homework 5

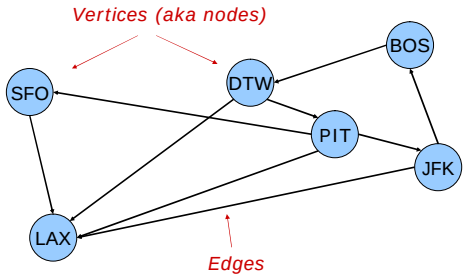
- Homework Assignment #5 is due on ***Tuesday, April 15, 11:59pm***!
- The next quiz (#3) will be postponed until ***Friday, April 4***.
- Reading: ***Chapter 14***.
- Many of you haven't started HW5!
  - ***Get started today!***

Introduction to Graphs

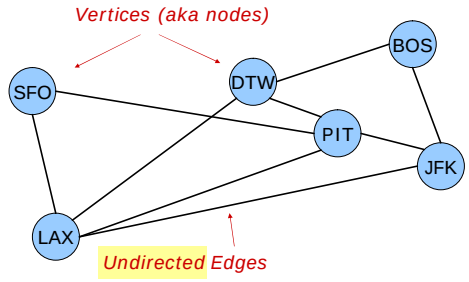
Graphs — an overview



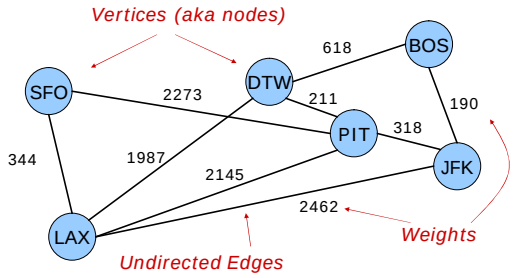
Graphs — an overview



Graphs — an overview



Graphs — an overview



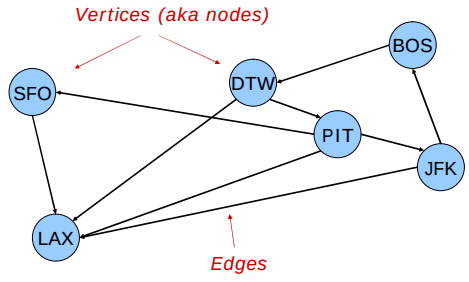
Terminology

- **Graph** G = (V,E)
  - Set V of **vertices** (**nodes**)
  - Set E of **edges**
    - Elements of E are pair (v,w) where v,w Î V.
    - An edge (v,v) is a **self-loop**. (Usually assume no self-loops.)
- **Weighted** graph
  - Elements of E are (v,w,x) where x is a **weight**.

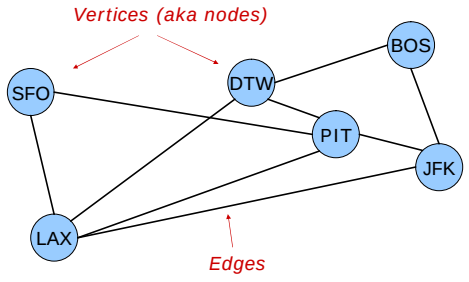
Terminology, cont'd

- **Directed** graph (**digraph**)
  - The edge pairs are ordered
  - Every edge has a specified direction
- **Undirected** graph
  - The edge pairs are unordered
    - E is a symmetric relation
    - (v,w) Î E *implies* (w,v) Î E
    - In an undirected graph (v,w) and (w,v) are usually treated as though they are the *same* edge

Directed Graph (digraph)



Undirected Graph



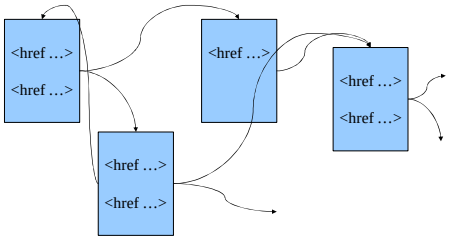
Terminology, cont'd

- **Connected graph**

  a graph where for every pair of nodes there exists a sequence of edges starting at one node and ending at the other.

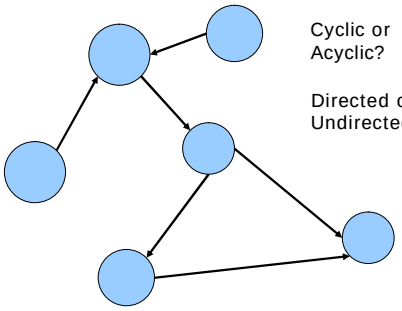
- **The web graph**
  - A directed graph where :
    - ***v*** = (all web pages) and
    - ***w*** = (all HTML-defined links from one web page to another)

Web Graph

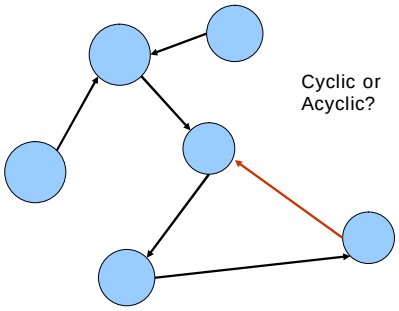


- Web Pages are nodes (vertices)
- HTML references are links (edges)

So, is this a connected graph?



Directed graph (unconnected)



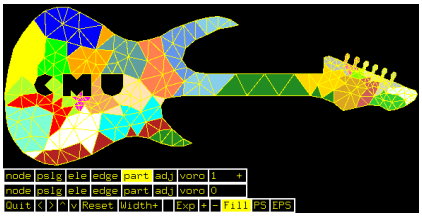
Graphs are Everywhere

Graphs as models

- *The Internet*
  - Communication pathways
  - DNS hierarchy
  - The WWW
- *The physical world*
  - Road topology and maps
  - Airline routes and fares
  - Electrical circuits
  - Job and manufacturing scheduling

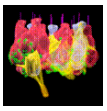
Graphs as models

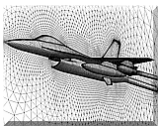
- Physical objects are often modeled by meshes, which are a particular kind of graph structure.



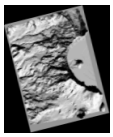
By Jonathan Shewchuk

More graph models





NASA CFD labs

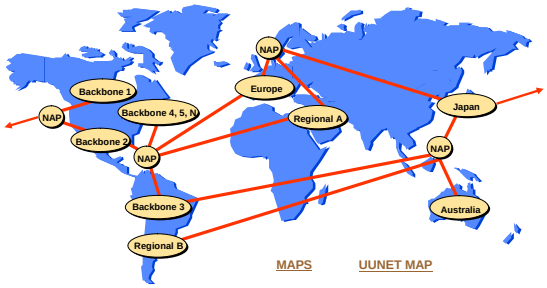


By Paul Heckbert and David Garland

See also
http://java.sun.com/applets/jdk/1.1/demo/WireFrame/index.html
and
http://www.mapquest.com

And yet more graph models…

Structure of the Internet

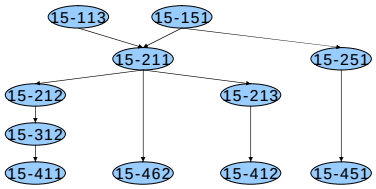


MAPS UUNET MAP

SOURCE: CISCO SYSTEMS

Relationship graphs

- Graphs are also used to model *relationships* among entities.
  - Scheduling and resource constraints.
  - Inheritance hierarchies.



Where are we right now?

Representing Graphs

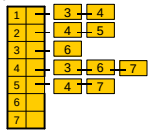
Graph operations

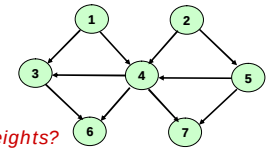
- Navigation in a graph.
  - (v,w)ÎE
  - {w | (v,w)ÎE}
  - {v | (v,w)ÎE}
- Enumeration of the elements of a graph.
  - E
  - V
- Size of a graph
  - |V|
  - |E|

Representing graphs

- Adjacency matrix
  - 2-dimensional array
  - For each edge (u,v), set A[u][v] to **true**; otherwise false
- Adjacency lists
  - For each vertex, keep a list of adjacent vertices

	1	2	3	4	5	6	7
1			x	x			
2				x	x		
3						x	
4			x			x	x
5				x			x
6							
7							





- *Q: How to represent weights?*

Choosing a representation

- Size of V relative to size of E is a primary factor.
  - *Dense*: E/V is large
  - *Sparse*: E/V is small
  - *Adjacency matrix is expensive if the graph is sparse.*
  - *Adjacency list is expensive if the graph is dense.*
- Dynamic changes to V.
  - *Adjacency matrix is expensive to copy/extend if V is extended. WHY?*

Quiz Break

Edit sequences

- A word can be changed to another word by a one-character substitution.
- Give an algorithm to determine whether a word A can be transformed into a word B by a series of one-character substitutions, and if so, outputs the sequences of words.
- For example:
  - bleed -> blend -> blond -> blood
- You may assume that you have a dictionary

Graphs : Application Search Engines

Search Engines

What are they?

- Tools for finding information on the Web
  - Problem: "hidden" databases, e.g. *New York Times (ie, databases of keywords hosted by the web site itself. These cannot be accessed by Yahoo, Google etc.)*
- Search engine
  - A machine-constructed index (usually by keyword)

What are they?

- Search engines: key tools for ecommerce
  - Buyers and sellers must find each other
- How do they work?
- How much do they index?
- Are they reliable?
- How are hits ordered?
- Can the order be changed?

The Process

1. Acquire the collection, i.e. all the documents
   [*Off-line process*]
2. Create an inverted index
   [*Off-line process*]
3. Match queries to documents
   [*On-line process, the actual retrieval*]
4. Present the results to user
   [*On-line process: display, summarize, ...*]

SE Architecture

- Spider
  - Crawls the web to find pages. Follows hyperlinks. Never stops
- Indexer
  - Produces data structures for fast searching of all words in the pages (ie, it updates the lexicon)
- Retriever
  - Query interface
  - Database lookup to find hits
    - 1 billion documents
    - 1 TB RAM, many terabytes of disk
  - Ranking

Did you know?

- The concept of a Web spider was developed by **Dr.Fuzzy Mouldin**
- Implemented in 1994 on the Web
- Went into the creation of Lycos
- Lycos propelled CMU into the top 5 most successful schools
  - Commercialization proceeds
- Tangible evidence
  - Newel-Simon Hall

Dr. Michael L. (Fuzzy) Mauldin

LYCOS

Did you know?

- Vivisimo was developed here at CMU
- Developed by **Prof. Raul Valdes-Perez**
- Developed in 2000

Vivisimo

A look at Google

- >20,000 servers (WOW!) ☺
- Web site traffic grew over 20% per month
- Spiders over 1.5 Billion URLs
- Supports 28 language searches and 25 countries
- Over 150 million searches per day
- "Even CMU uses it!" ☺

Gooooooooogle

Google's server farm

Web Crawlers

- Start with an initial page P_0. Find URLs on P_0 and add them to a queue
- When done with P_0, pass it to an indexing program, get a page P_1 from the queue and repeat
- Can be specialized (e.g. only look for email addresses)
- Issues
  - Which page to look at next? (Special subjects, recency)
  - Avoid overloading a site
  - How deep within a site do you go (depth search)?
  - How frequently to visit pages?

So, why Spider the Web?

User Perceptions

- **Most annoying:** Engine finds nothing (too small an index, but not an issue since 1997 or so).
- **Somewhat annoying:** Obsolete links
  - Refresh Collection by deleting dead link
    - OK if index is slightly smaller
  - Done every 1-2 weeks in best engines
- **Mildly annoying:** Failure to find new site

  => Re-spider entire web

  => Done every 2-4 weeks in best engines

So, why Spider the Web?, cont'd

Cost of Spidering

- Semi-parallel algorithmic decomposition
- Spider can (and does) run in hundreds of severs simultaneously
- Very high network connectivity (e.g. T3 line)
- Servers can migrate from spidering to query processing depending on time-of-day load
- Running a full web spider takes days even with hundreds of dedicated servers

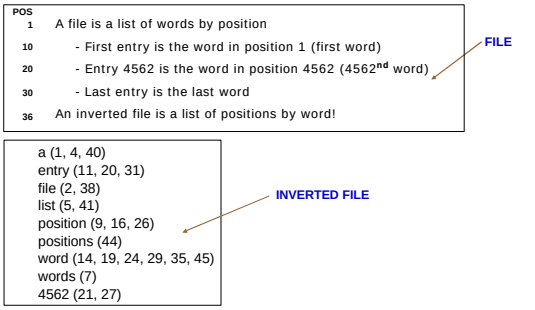
Indexing

- Arrangement of data (data structure) to permit fast searching
- Which list is easier to search?

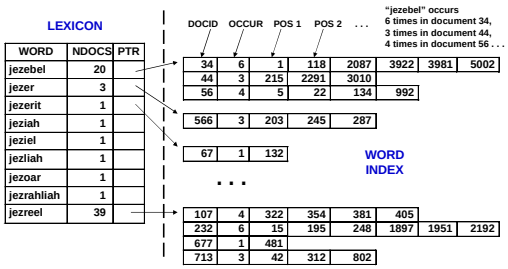
  sow fox pig eel yak hen ant cat dog hog

  ant cat dog eel fox hen hog pig sow yak
- Sorting helps. Why?
  - Permits binary search. About $\log_2 n$ probes into list
    - $\log_2$(1 billion) ~ 30
  - Permits interpolation search. About $\log_2(\log_2 n)$ probes
    - $\log_2 \log_2$(1 billion) ~ 5

Inverted Files



Inverted Files for Multiple Documents



Ranking (Scoring) Hits

- Hits must be presented in some order
- What order?
  - Relevance, recency, popularity, reliability?
- Some ranking methods
  - Presence of keywords in title of document
  - Closeness of keywords to start of document
  - Frequency of keyword in document
  - Link popularity (how many pages point to this one)

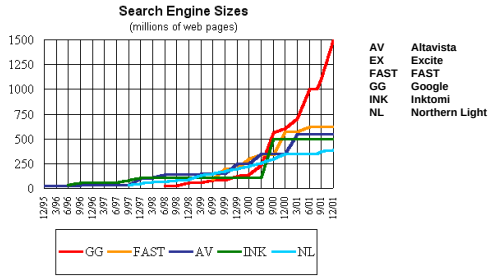
Ranking (Scoring) Hits, cont'd

- Can the user control? Can the page owner control?
- Can you find out what order is used?
- Spamdexing: influencing retrieval ranking by altering a web page. (Puts "spam" in the index)

Link Popularity

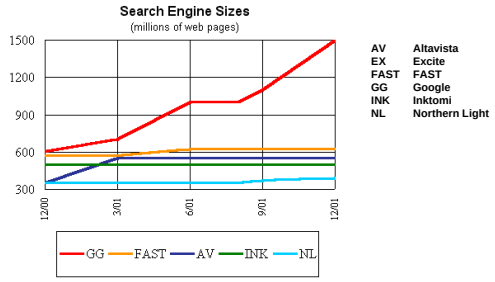
- How many pages link to this page?
  - on the whole Web
  - in our database?
- www.linkpopularity.com
- Link popularity is used for ranking
  - Many measures
  - Number of links in (In-links)
  - Weighted number of links in (by weight of referring page)
- PositionAgent

Search Engine Sizes



SOURCE: SEARCHENGINEWATCH.COM

Search Engine Sizes, cont'd



SOURCE: SEARCHENGINEWATCH.COM

Current Status of Web Spiders

Historical Notes

- WebCrawler: first documented spider
- Lycos: first large-scale spider
- Top-honors for most web pages spidered: First Lycos, then AltaVista, then Google…

Current Status of Web Spiders

Enhanced Spidering

- In-link counts to pages can be established during spidering.
- Hint: Hmmm… hw5?
- In-link counts are the basis for Google's page-rank method

Current Status of Web Spiders

Unsolved Problems

- Most spidering re-traverses stable web graph

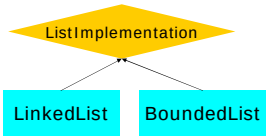
  => on-demand re-spidering when changes occur
- Completeness or near-completeness is still a major issue
- Cannot Spider information stored in local-DB

Key Takeaways

- Engines are a critical Web resource
- Very sophisticated, high technology, but secret
- Most spidering re-traverses stable web graph
- They don't spider the Web completely
- Spamdexing is a problem
- New paradigms needed as Web grows
  - Keywords are not enough. Try hyperbolic trees
- What about images, music, video?
  - Google image search

A Bit About Inner Classes

List implementation ADT




```
public interface ListImplementation
{
  Object get (int i);
  void set (int i, int x);
  int size ();
…
}
```

Simple bounded lists

```
public class BoundedList implements
                        ListImplementation {
…
  public Object get (int i) {
    return elts[i];
  }

  public void set (int i, Object x) {
    elts[i] = x;
  }
…
  private Object elts[];
}
```

Simple linked lists

```
public class LinkedList implements
                        ListImplementation {
  public Object get (int i) {
    if (i == 0) return value;
    else return next.get(i-1);
  }

  public void set (int i, Object x) {
    if (i == 0) value = x;
    else next.set(i-1, x);
}
…
  private Object value;
  private LinkedList next;
}
```

Iterators

- For Collection objects, it is often very useful to get each element of the object, one at a time.
- For example, to print out the elements of a list.
- The JDK's Iterator interface captures this.

Iterator interface

```
public interface Iterator {

  boolean hasNext ();
  Object next ();
  void remove (); //optional

}
```

A simple list iterator

```
public class ListIterator implements Iterator {

  public ListIterator (ListImplementation l) {
    theList = l;    current = 0;
  }

  public boolean hasNext() {
    return current < theList.size(); }

  public Object next() {
    return theList.get(current++);
  }

  private ListImplementation theList;
  private int current;
}
```

Using an iterator

```
…
BoundedList s = new BoundedList(100);
…
ListIterator it = new ListIterator(s);
…
while (it.hasNext()) {
    System.out.println (it.next());
}
…
```

What is the running time for the while loop?

Using an iterator, cont'd

```
…
LinkedList s = new LinkedList();
…
ListIterator it = new ListIterator(s);
…
while (it.hasNext()) {
    System.out.println (it.next());
}
…
```

What is the running time for the while loop?

Inner classes

- To get a good iterator for LinkedList, the iterator needs access to the private fields of the LinkedList class.
- This can be accomplished by defining the iterator class *inside* the LinkedList class.
  - This is called an *inner class*.

LinkedList iterator

```
public class LinkedList implements
                        ListImplementation {
…
  public Object get (int i) { … }
  public void set (int i, Object x) { … }
…
  private Object value;
  private LinkedList next;
…
  class ListIterator implements Iterator { … }
}
```

An inner class

LinkedList iterator, cont'd

```
public class LinkedList implements ListImplementation {
…
  class ListIterator implements Iterator {
    ListIterator () { current = value; theRest = next }

    boolean hasNext () { return value != null; }

    Object next () {
      if (value != null) {
        Object temp = current;
        current = next.value;
        theRest = next.next;
        return temp;
      } else …;
    }

    Object value;   ListImplementation theRest;
  }
…
}
```

Inner classes

- By putting the ListIterator class inside the LinkedList class, ListIterator gets access to the private variables of LinkedList.
- This allows an efficient iterator for LinkedList to be implemented.
- Inner classes are not always needed, but are useful in such cases.