CPT-S 415

Big Data

Yinghui Wu

EME B45

CPT-S 415
Big Data

Graph Query Processing

- ✓ Basics of Graph Algorithms
  - Graph search (traversal)
  - PageRank
  - Nearest neighbors
  - Keyword search
  - Graph pattern matching (next lecture)

Graph query processing: operators and algorithms

When it comes to graphs …

- Semistructured:
  - No schema
  - No constraints yet
- No standard query languages
  - A variety of queries used in practice
  - Nontrivial
- What is the complexity of the following problems?
  - Subgraph isomorphism — NP-complete
  - Simple path: given a graph G, a pair (s, t) of nodes in G, and a regular expression R, it is to decide whether there exists a simple path from s to t that satisfies R.
- Query optimization techniques, indexing, updates, … preliminary

The study of graph queries is still in its infancy

Basic graph queries and algorithms

- ✓ Graph search (traversal)
- ✓ PageRank
- ✓ Nearest neighbors
- ✓ Keyword search
- ✓ Graph pattern matching (a full treatment of itself)

Widely used in graph algorithms

Graph search (traversal)

Path queries

- ✓ Reachability
  - Input: A directed graph G, and a pair of nodes s and t in G
  - Question: Does there exist a path from s to t in G?
- ✓ Distance
  - Input: A directed weighted graph G, and a node s in G
  - Output: The lengths of shortest paths from s to all nodes in G
- ✓ Regular path
  - Input: A node-labeled directed graph G, a pair of nodes s and t in G, and a regular expression R
  - Question: Does there exist a (simple) path p from s to t that satisfies R?

Reachability queries

- ✓ Reachability
  - Input: A directed graph G, and a pair of nodes s and t in G
  - Question: Does there exist a path from s to t in G?
- ✓ Applications: a routine operation
  - Social graphs: are two people related for security reasons?
  - Biological networks: find genes that are (directly or indirectly influenced by a given molecule

Nodes: molecules, reactions or physical interactions

Edges: interactions

How to evaluate reachability queries?

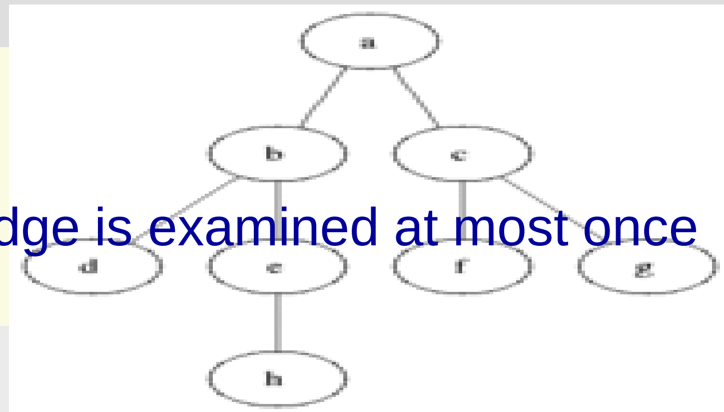
Breadth-first search

- ✓ BFS (G, s, t):
  1. while Que is nonempty do
     a. v ← Que.dequeue();
     b. if v = t then return true;
     c. for all adjacent edges e = (v, u) of v do
        a) if not flag(u)
           then flag(u) ← true; enqueue u onto Que;
  2. return false

Use (a) a queue Que, initialized with s, (b) flag(v) for each node, initially false; and (c) adjacency lists to store G

Breadth-first, by using a queue

Why do we need the test?



- ✓ Complexity: each node and edge is examined at most once

What is the

Breadth-first search

Too costly as a routine operation when G is large

- ✓ Reachability: NL-complete
- ✓ BFS (G, s, t): O(|V| + |E|) time and space
  1. while Que is nonempty do
     a. v ← Que.dequeue();
     b. if v = t then return true;
     c. for all adjacent edges e = (v, u) of v do
        a) if not flag(u)
           then flag(u) ← true; enqueue u onto Que;
  2. return false
- ✓ O(1) time? Yes, adjacency matrix, but $O(|V|^2)$ space

How to trike a balance?

2-hop cover

- For each node v in G,

$$2hop(v) = (L_{in}(v),\ L_{out}(v))$$

  - $L_{in}(v)$: a set of nodes in G that can reach v
  - $L_{out}(v)$: a set of nodes in G that v can reach
- To ensure: node s can reach t if and only if

$$L_{out}(s) \cap L_{in}(t) \neq \varnothing$$

  - Testing: better than $O(|V| + |E|)$ on average
  - Space: $O(|V|\ |E|^{1/2})$

Find a minimum 2-hop cover? NP-hard

Maintenance cost in response to changes to G

A number of algorithms for reachability queries (see reading list)

Distance queries

Edsger Wybe Dijkstra
(1930-2002)

- ✓ Distance: single-source shortest-path problem
  - Input: A directed weighted graph G, and a node s in G
  - Output: The lengths of shortest paths from s to all nodes in G
- ✓ Application: transportation networks
- ✓ Dijkstra (G, s, w):

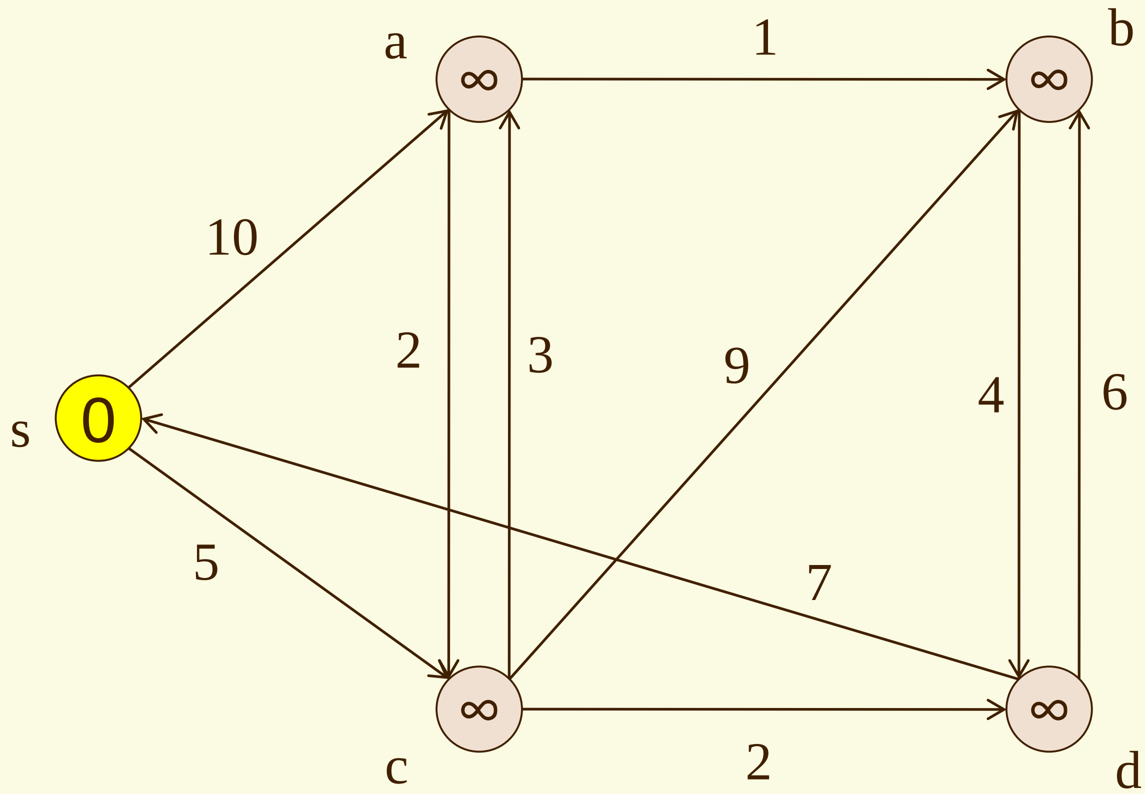
Use a priority queue Que; w(u, v): weight of edge (u, v); d(u): the distance from s to u

1. for all nodes v in V do
   a. d[v] $\leftarrow \infty$; $\leftarrow$
2. d[s] $\leftarrow$ 0; Que $\leftarrow$ V;
3. while Que is nonempty do
   a. u $\leftarrow$ ExtractMin(Que);
   b. for all nodes v in adj(u) do
      a) if d[v] > d[u] + w(u, v) then d[v] $\leftarrow$ d[u] + w(u, v);

Extract one with the minimum d(u)

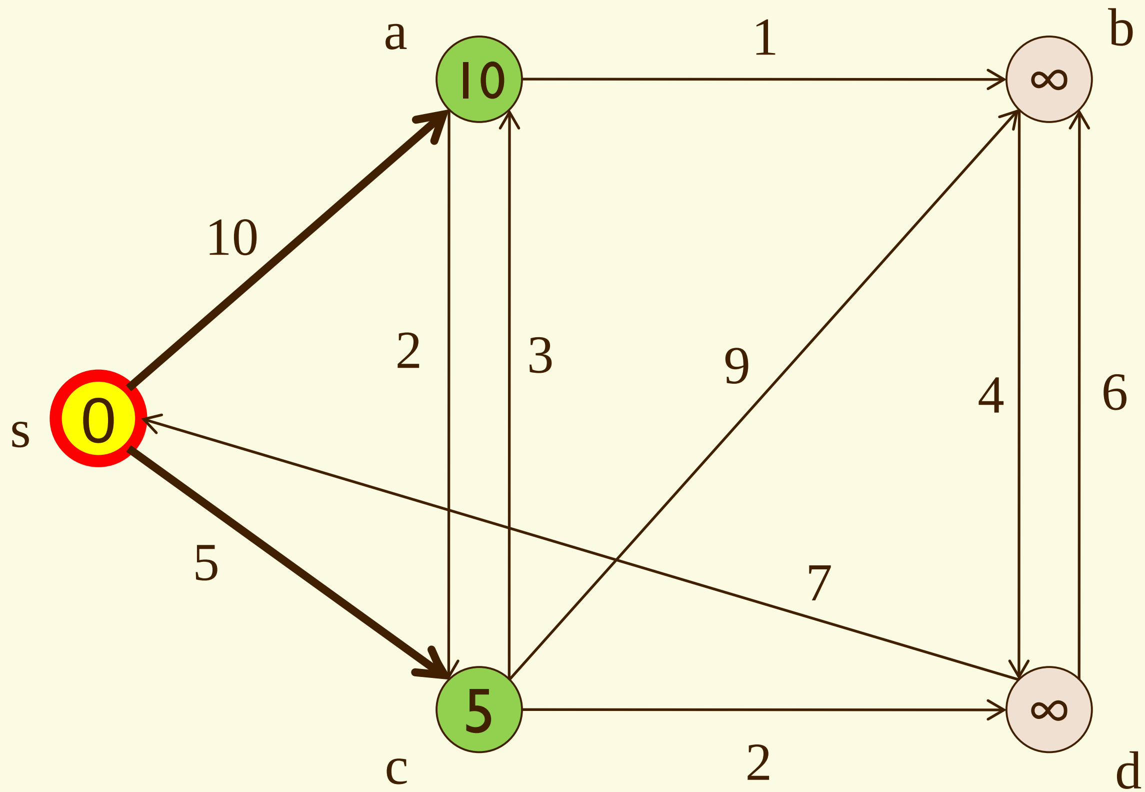
Example from CLR 2nd ed. p. 528

Example: Dijkstra’s algorithm



Q = {s,a,b,c,d}
d: {(a,∞), (b,∞), (c,∞), (d,∞)}

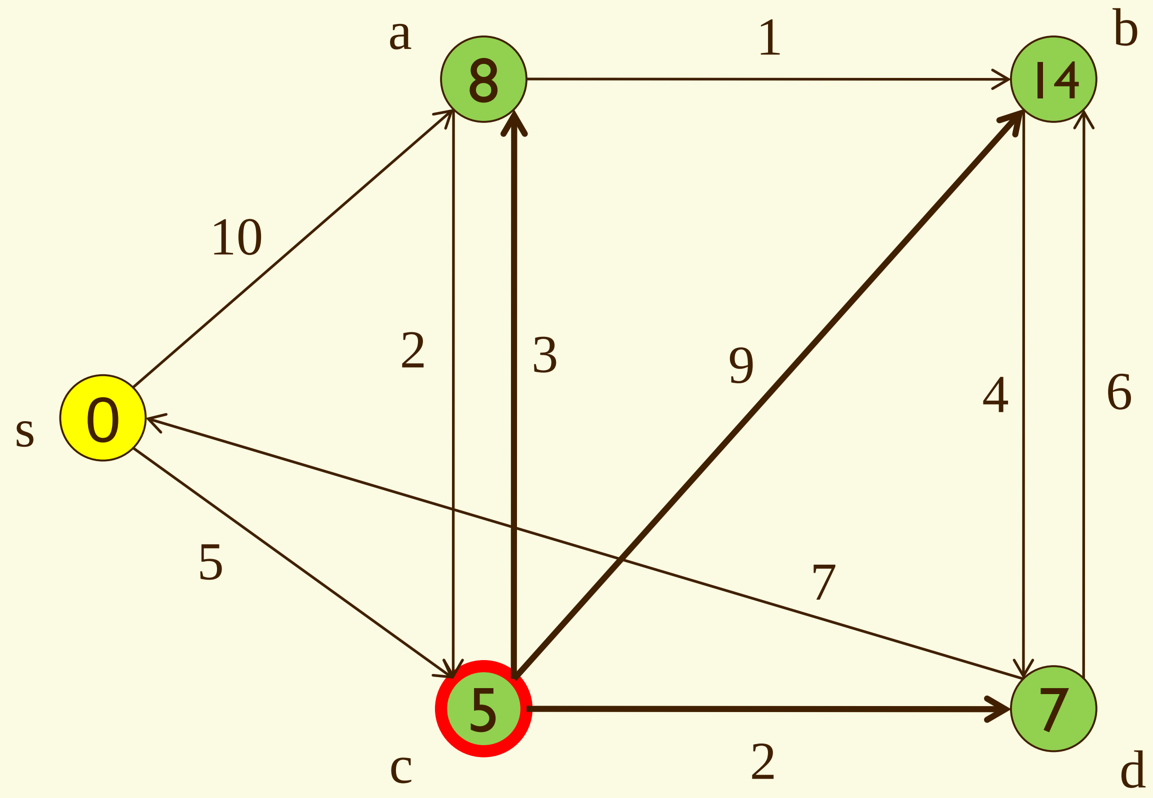
Example: Dijkstra's algorithm



Q = {a,b,c,d}
d: {(a,10), (b,∞), (c,5), (d,∞)}

Example from CLR 2nd ed. p. 528

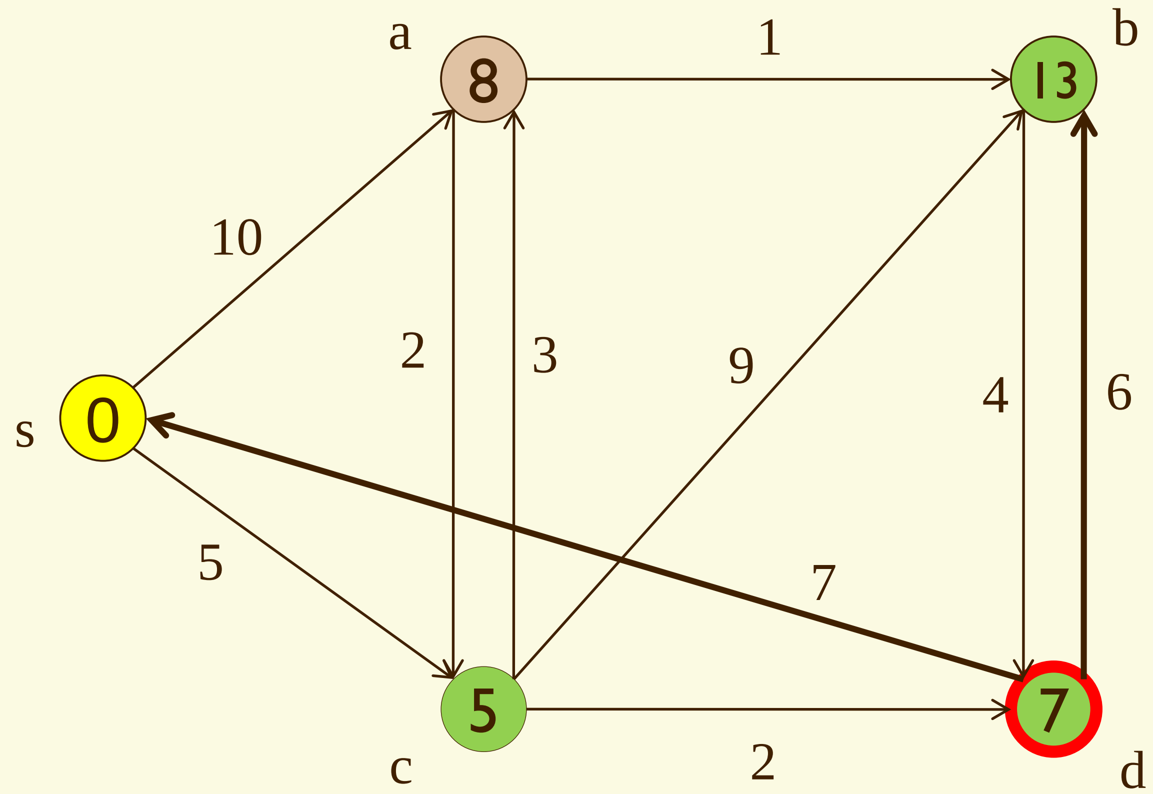
Example: Dijkstra's algorithm



Q = {a,b,d}
d: {(a,8), (b,14), (c,5), (d,7)}

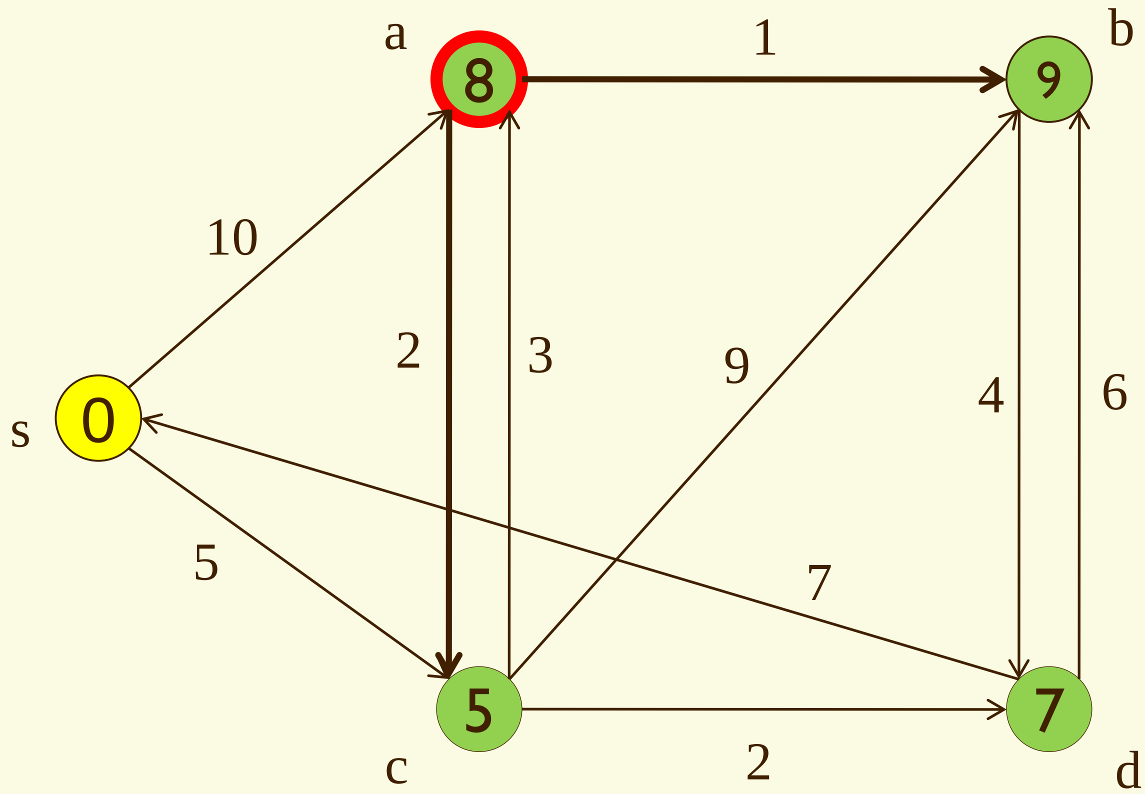
Example from CLR 2nd ed. p. 528

Example: Dijkstra's algorithm



Q = {a,b}
d: {(a,8), (b,13), (c,5), (d,7)}

Example: Dijkstra's algorithm

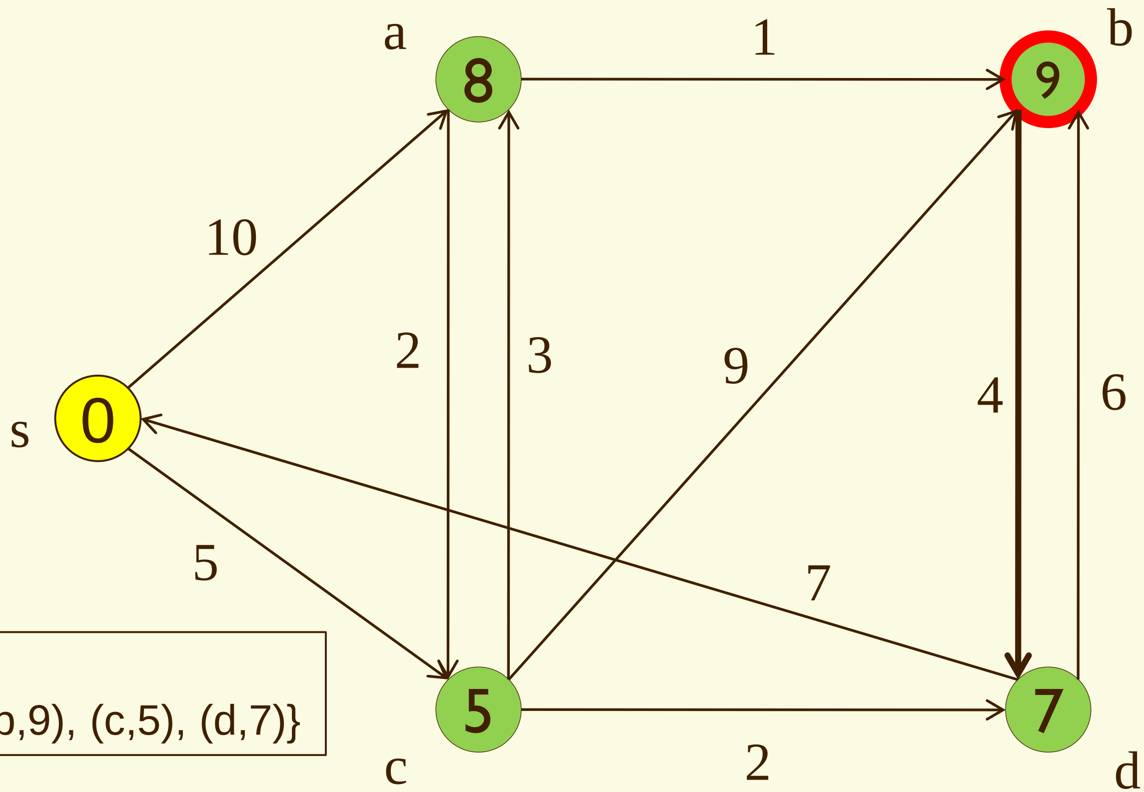


Q = {b}
d: {(a,8), (b,9), (c,5), (d,7)}

Example from CLR 2nd ed. p. 528

Example: Dijkstra’s algorithm

Example from CLR 2nd ed. p. 528



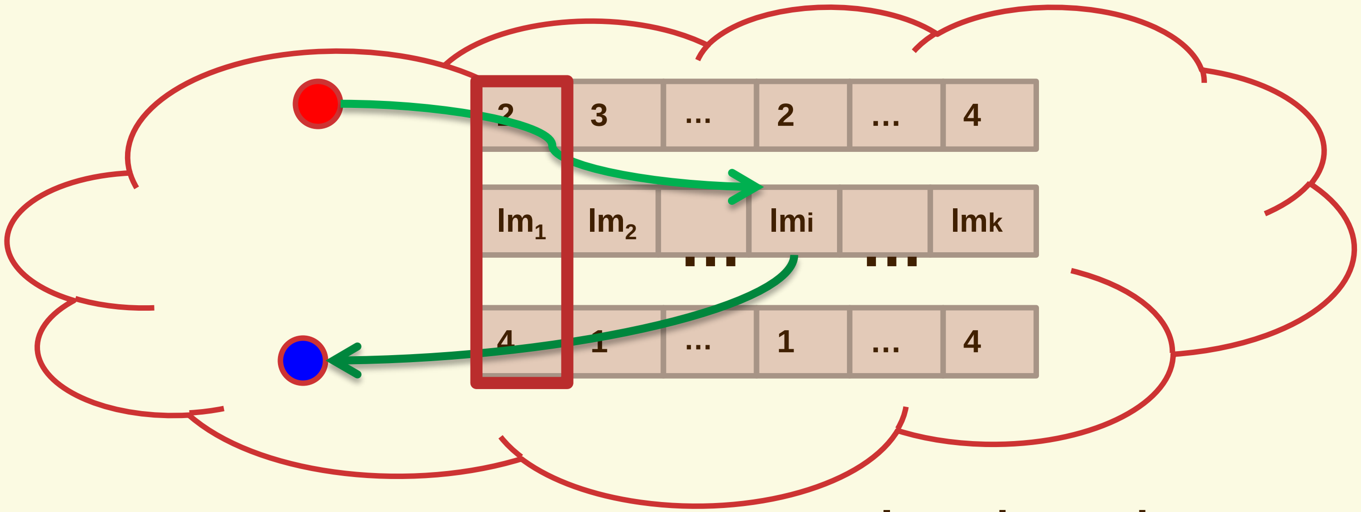
Q = {}
d: {(a,8), (b,9), (c,5), (d,7)}

How to speed it up?

O(|V| log|V| + |E|). A beaten-to-death topic ?

Landmarks

- Landmark vectors
  - A list of nodes L in a graph G, s.t for each pair (u,v) of nodes in G, there is an node in L on a shortest path from u to v
  - Answering distance query: linear time



A landmark vector LM

Regular path queries

- Regular simple path
  - Input: A node-labeled directed graph G, a pair of nodes s and t in G, and a regular expression R
  - Question: Does there exist a simple path p from s to t such that the labels of ad

A. Mendelzon, and P. T. Wood. *Finding Regular Simple Paths In Graph Databases. SICOMP 24(6), 1995*

- NP-complete, even when R is a fixed regular expression (00)* or 0*10*.
- In PTIME when G is a DAG (directed acyclic graph)

Patterns of social links

Why do we care about regular path queries?

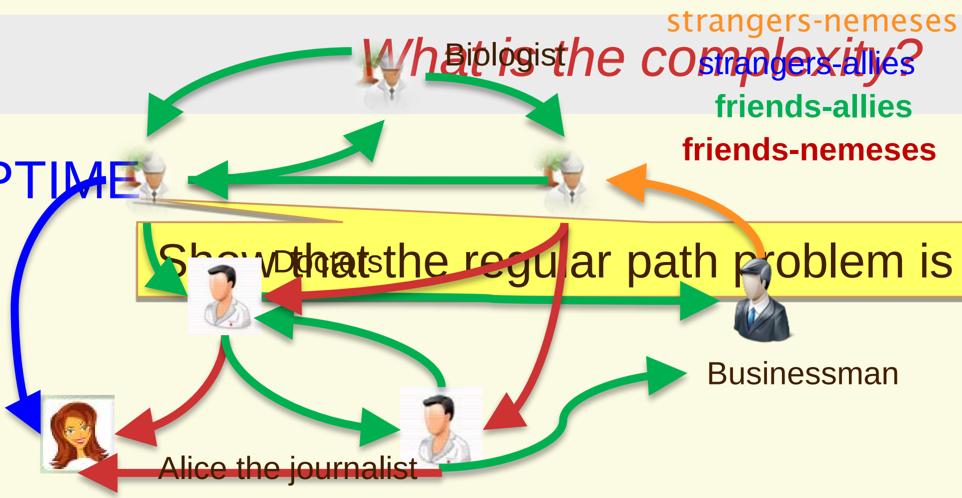
Regular path queries

- Regular path
  - Input: A node-labeled directed graph G, a pair of nodes s and t in G, and a regular expression R
  - Question: Does there exist a path p from s to t such that the labels of adjacent nodes on p form a string in R?

What is the complexity?

- PTIME

Show that the regular path problem is in O(|G| |R|) time



a social voting network

potential supporters?
$fa^{<=2}$ $sa^{<=2}$

Graph queries are nontrivial, even for path queries

Strongly connected components

- A strongly connected component in a direct graph G is a set V of nodes in G such that
  - for any pair (u, v) in V, u can reach v and v can reach u; and
  - V is maximal: adding any node to V makes it no longer strongly connected
- SCC
  - Input: A graph G
  - Question: all strongly connected components of G

Find social circles: how large? How many?

What is the complexity?

by extending search algorithms, e.g., BFS

$O(|V| + |E|)$

PageRank

Introduction to PageRank

- To measure the "quality" of a Web page
  - Input: A directed graph G modelling the Web, in which nodes represent Web pages, and edges indicate hyperlinks
  - Output: For each node v in G, P(v): the likelihood that a random walk over G will arrive at v.
- Intuition: how a random walk can reach v?
  - A random jump: α (1/|V|)

    The chances of hitting v among |V| pages
    - α: random jump factor (teleportation factor)
  - Following a hyperlink: $(1 - \alpha) \sum(u \in L(v))\, P(u)/C(u)$
    - $(1 - \alpha)$: damping factor
    - $(1 - \alpha) \sum(u \in L(v))\, P(u)/C(u)$: the chances for one to click a hyperlink at a page u and reach v

Intuition

- Following a hyperlink: $(1 - \alpha) \sum_{(u \in L(v))} P(u)/C(u)$
  - L(v): the set of pages that link to v;
  - C(u): the out-degree of node u (the number of links on u)
  - P(u)/C(u): The chances of reaching page v from page u
    - the probability of u being visited itself
    - the probability of clicking the link to v among C(u) many links on page u
- Intuition:
  - the more pages link to v, and
  - the more popular those pages that link to v,

v has a higher chance to be visited

One of the models

Putting together

The likelihood that page v is visited by a random walk:

$$\alpha\ (1/|V|)\ +\ (1 - \alpha)\ \Sigma\_(u \in L(v))\ P(u)/C(u)$$

random jump

following a link from other pages

- Recursive computation: for each page v in G,
  - compute P(v) by using P(u) for all $u \in L(v)$

until

too expensive; use an error factor

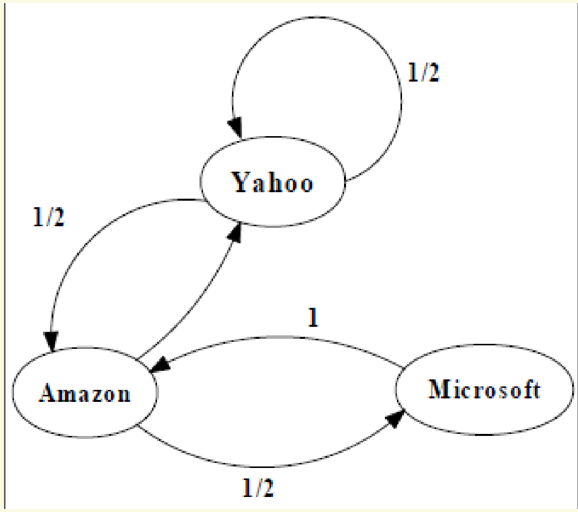
- converge: no changes to any P(v)
- after a fixed number of iterations

costly: trillions of pages

Parallel computation

How to speed it up?

An example of Simplified PageRank



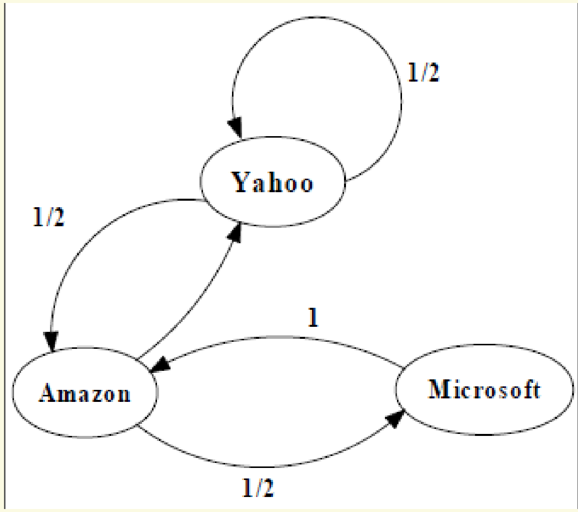
$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

PageRank Calculation: first iteration

An example of Simplified PageRank



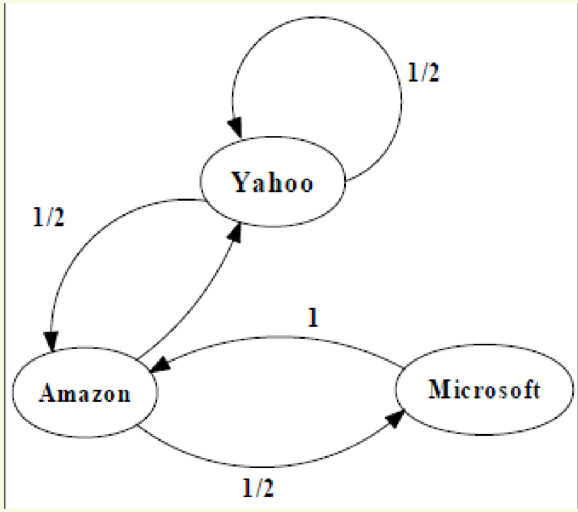
$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 5/12 \\ 1/3 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix}$$

PageRank Calculation: second iteration

An example of Simplified PageRank



$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 3/8 \\ 11/24 \\ 1/6 \end{bmatrix} \begin{bmatrix} 5/12 \\ 17/48 \\ 11/48 \end{bmatrix} \ldots \begin{bmatrix} 2/5 \\ 2/5 \\ 1/5 \end{bmatrix}$$

Convergence after some iterations

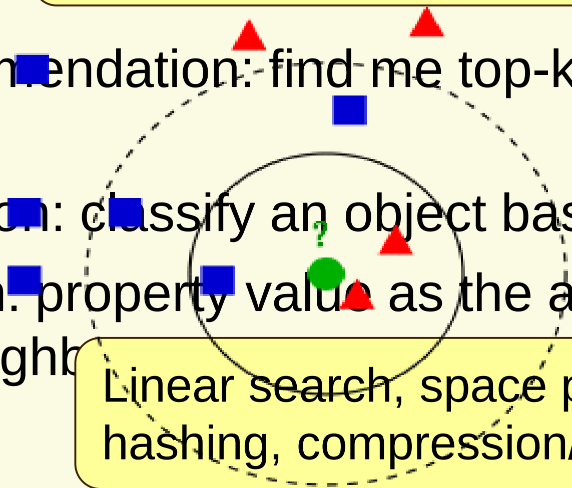
Nearest neighbors

Nearest neighbor

- ✓ Nearest neighbor (kNN)
  - Input: A set S of points in a space M, a query point p in M, a distance function dist(u, v), and a positive integer k
  - Output: Find top-k points in S that are closest to p based on dist(p, u)

Euclidean distance, Hamming distance, continuous variables, …

- ✓ Applications
  - POI recommendation: find me top-k restaurants close to where I am
  - Classification: classify an object based on its nearest neighbors
  - Regression: property value as the average of the values of its k nearest neighbors



Linear search, space partitioning, locality sensitive hashing, compression/clustering based search, …

A number of techniques

kNN join

- kNN join
  - Input: Two datasets R and S, a distance function dist(r, s), and a positive integer k
  - Output: pairs (r, s) for all r in R, where s is in S, and is one of the k-nearest neighbors of r
- A naive algorithm

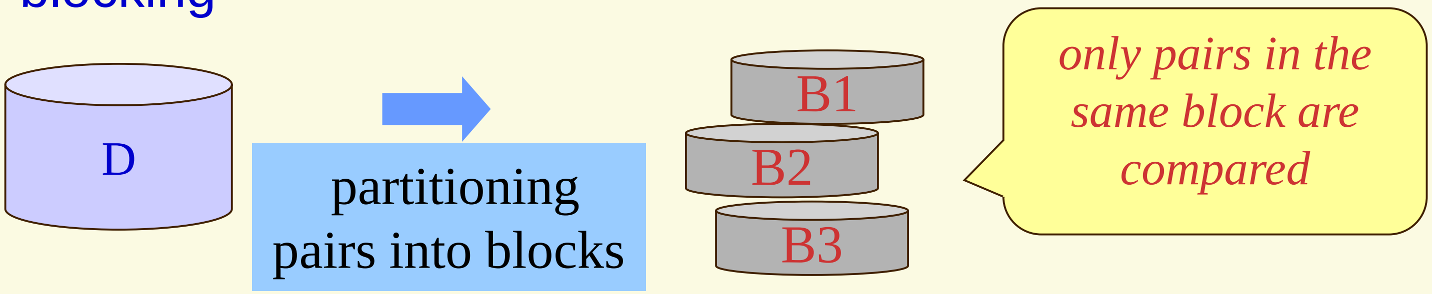
Pairwise comparison

  - Scanning S once for each object in R
  - O(|R| |S|): expensive when R or S is large

Can we do better?

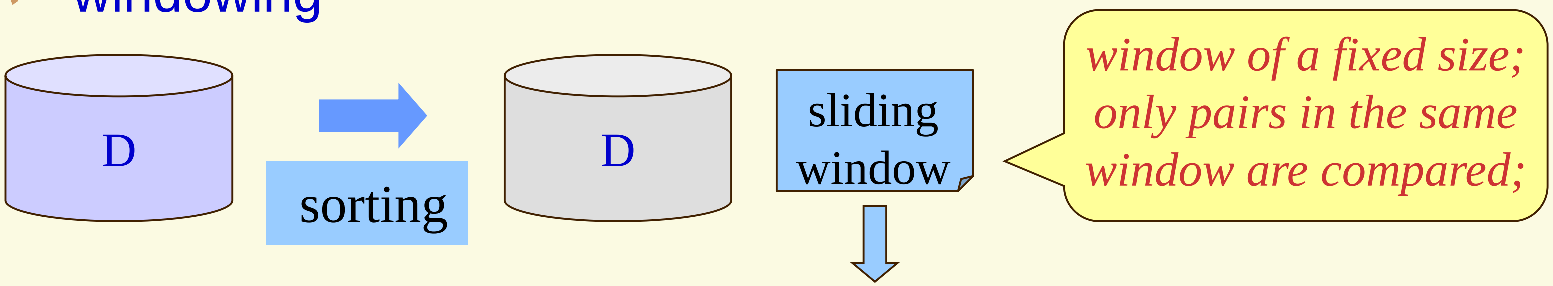
Blocking and windowing

- ✓ blocking



GORDER: An Efficient Method for KNN Join Processing. VLDB 2004.

- ✓ windowing



Several indexing and ordering techniques

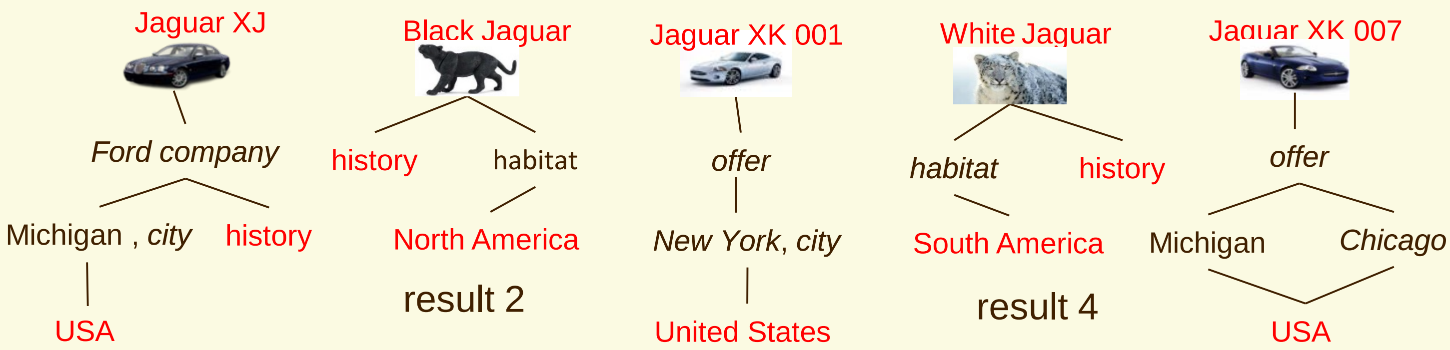
Keyword search

Keyword search

- Input: A list Q of keywords, a graph G, a positive integer k
- Output: top-k "matches" of Q in G

Information retrieval

Query Q: '[Jaguar', 'America', 'history']



- What makes a match?
- How to sort the matches?
- How to efficiently find top-k matches?

Questions to answer

Semantics: Steiner tree

- Input: A list Q of keywords, a graph G, a weight function w(e) on the edges on G, and a positive integer k
- Output: top-k Steiner trees that match Q

PageRank scores

- Match: a subtree T of G such that
  - each keyword in Q is contained in a leaf of T
- Ranking:
  - The total weight of T (the sum of w(e) for all edges e in T)

The cost to connect the keywords

- Complexity?

NP-complete

What can we do about it?

Semantics: distinct-root (tree)

- Input: A list Q of keywords, a graph G, and a positive integer k
- Output: top-k distinct trees that match Q
- Match: a subtree T of G such that
  - each keyword in Q is contained in a leaf of T
- Ranking:
  - dist(r, q): from the root of T to a leaf q
  - The sum of distances from the root to all leaves of T
- Diversification:
  - each match in the top-k answer has a distinct root

$O(|Q|\ (|V| \log |V| + |E|))$

Semantics: Steiner graphs

- Input: A list Q of keywords, an undirected (unweighted) graph G, a positive integer r, and a positive integer k
- Output: Find all r-radius Steiner graphs that match Q
- Match: a subgraph G' of G such that it is
  - r-radius: the shortest distance between any pair of nodes in G is at most r (at least one pair with the distance); and
  - each keyword is contained in either
    - a content node: containing the key word
    - a Steiner node: on a simple path between a pair of content nodes
- Computation: M^r, the r-th power of adjacency graph of G

Revision: minimum subgraphs

Answering keyword queries

- A host of techniques
  - Backward search
  - Bidirectional search
  - Bi-level indexing
  - …
- G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. ICDE 2002.
- V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. VLDB 2005.
- H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: ranked keyword searches on graphs. SIGMOD 2007.

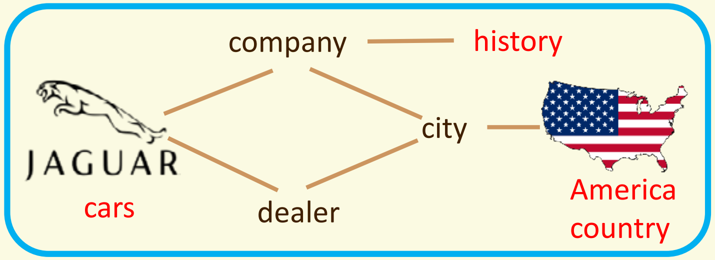
A well studied topic

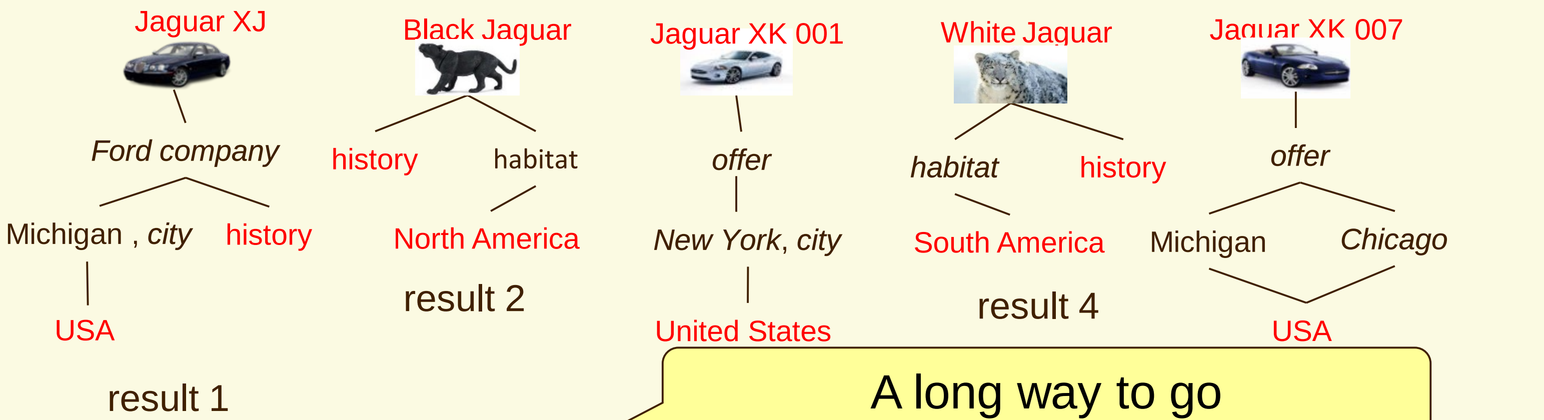
However, …

What does the user really want to find? Tree or graph? How to explain matches found?

✓ The semantics is rather "ad hoc"

Query Q: '[Jaguar', 'America', 'history']





A long way to go

Add semantics to keyword search

Summing up

Reading List 2: Graph query languages

There have been efforts to develo

Querying topological structures

- ✓ SoQL: an SQL-like language to retrieve paths
- ✓ CRPQ: extending conjunctive queries with regular path expressions
  - R. Ronen and O. Shmueli. SoQL: A language for querying and creating data in social networks. ICDE, 2009.
  - P. Barceló, C. A. Hurtado, L. Libkin, and P. T. Wood. Expressive languages for path queries over graph-structured data. In PODS, 2010
- ✓ SPARQL: for RDF data

Read this

- http://www.w3.org/TR/rdf-sparql-query/

Unfortunately, no "standard" query language for graphs, yet

Summary and review

- ✓ Review of searching RDBMS
  - Algorithms for selection/complex selection
  - Algorithms for joins: what are several common ways to compute joins? Design ideas?
- ✓ Why are reachability queries? Regular path queries? Connected components? Complexity? Algorithms?
- ✓ What are factors for PageRank? How does PageRank work?
- ✓ What are kNN queries? What is a kNN join? Complexity?
- ✓ What are keyword queries? What is its Steiner tree semantics? Distinct-root semantics? Steiner graph semantics? Complexity?
- ✓ Name a few applications of graph queries you have learned.
- ✓ Find graph queries that are not covered in the lecture.

Reading List 2 (graduate students)

- G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. ICDE 2002.
- V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. VLDB 2005.
- H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: ranked keyword searches on graphs. SIGMOD 2007.
- W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu. Adding Regular Expressions to Graph Reachability and Pattern Queries, ICDE 2011.
- M. R. Henzinger, T. Henzinger, and P. Kopke. Computing simulations on finite and infinite graphs. FOCS, 1995. http://infoscience.epfl.ch/record/99332/files/HenzingerHK95.pdf
- L. P. Cordella, P. Foggia, C. Sansone, M. Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs, IEEE Trans. Pattern Anal. Mach. Intell. 26, 2004 (search Google scholar)
- A. Fard, M. U. Nisar, J. A. Miller, L. Ramaswamy, Distributed and scalable graph pattern matching: models and algorithms. Int. J. Big Data. http://cobweb.cs.uga.edu/~ar/papers/IJBD_final.pdf
- W. Fan J. Li, S. Ma, and N. Tang, and Y. Wu. *Graph pattern matching: From intractable to polynomial time*, VLDB, 2010.
- W. Fan, F. Geerts, and F. Neven. *Making Queries Tractable on Big Data with Preprocessing*, VLDB 2013