Spatial Networks

UNIVERSITY OF MINNESOTA
Driven to Discover℠

Spatial Computing
Research Group

Outline

1. Motivation, and use cases
2. Example spatial networks
3. Conceptual model
4. Need for SQL extensions
5. CONNECT statement
6. RECURSIVE statement
7. Storage and data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

Navigation Systems

- Historical
  - Navigation is a core human activity for ages!
  - Trade-routes, Routes for Armed-Forces
- Recent Consumer Platforms
  - Devices: Phone Apps, In-vehicle, “GPS”, …
  - WWW: Google Maps, MapQuest, …
- Services
  - Display map around current location
  - Compute the shortest route to a destination
  - Help drivers follow selected route

Location Based Services

- Location: Where am I ?
  - Geo-code: Place Name (or Street Address) → <latitude, longitude>
  - Reverse Geo-code: <latitude, longitude> → Place Name
- Directory: What is around me?
  - Where is the nearest Clinic? Restaurant? Taxi?
  - List all Banks within 1 mile.
- Routes: How do I get there?
  - What is the shortest path to get there?
  - …

Limitations of Spatial Querying

- OGIS Simple Feature Types
  - Supports Geometry (e.g., Points, LineStrings, Polygons, …)
  - However, lack Graphs data type, shortest_path operator

- Traditional SQL
  - Supports select, project, join, statistics
  - Lacked transitive closure, e.g., network analysis (next slide)
  - SQL3 added recursion & transitive closure

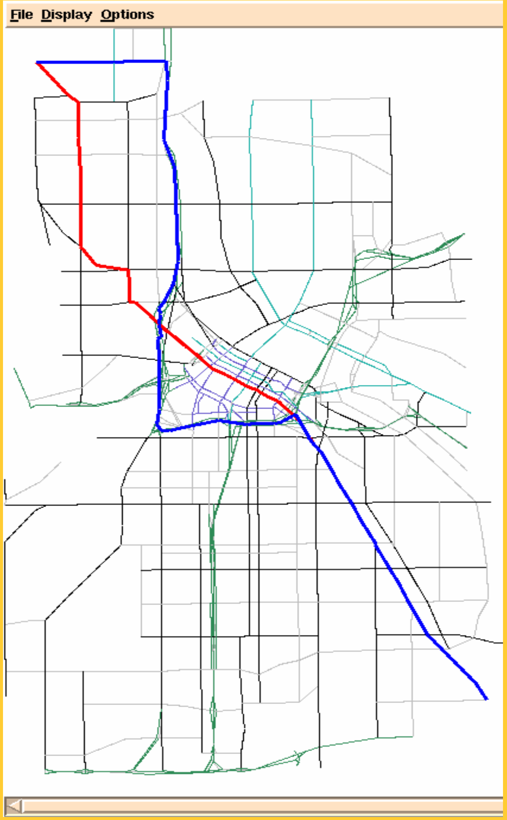
Spatial Network Analysis

- Route (A start-point, Destination(s))
  - What is the shortest path to get there?
  - What is the shortest path to cover a set of destinations?
- Allocation (A set of service centers, A set of customers)
  - Assign customers to nearest service centers
  - Map service area for each service center
- Site Selection (A set of customers, Number of new service centers)
  - What are best locations for new service centers ?

Outline

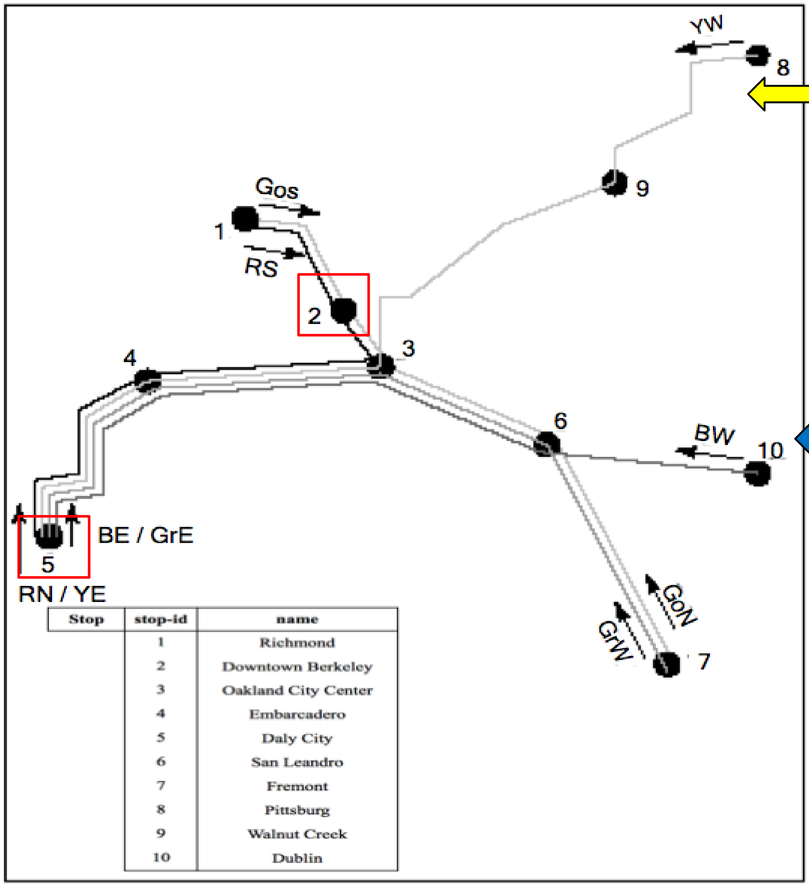
1. Motivation, and use cases
2. Example spatial networks
3. Conceptual model
4. Need for SQL extensions
5. CONNECT statement
6. RECURSIVE statement
7. Storage and data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

Spatial Network Query Example



1. Find shortest path from a start-point to a destination
2. Find nearest hospital by driving distance
3. Find shortest route to deliver packages to a set of homes
4. Allocate customers to nearest service center

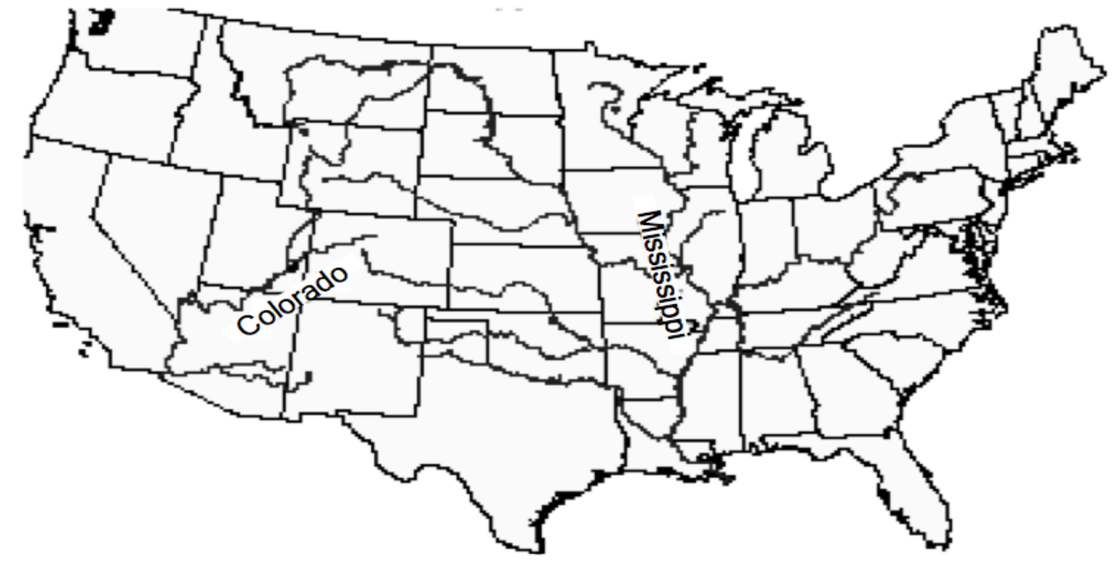
Railway Network & Queries



Stop	stop-id	name
	1	Richmond
	2	Downtown Berkeley
	3	Oakland City Center
	4	Embarcadero
	5	Daly City
	6	San Leandro
	7	Fremont
	8	Pittsburg
	9	Walnut Creek
	10	Dublin

1. Find the number of stops on the Yellow West (YW) route.
2. List all stops which can be reached from Downtown Berkeley (2)
3. List the routes numbers that connect Downtown Berkeley (2) & Daly City (5)
4. Find the last stop on the Blue West (BW) route

River Network & Queries

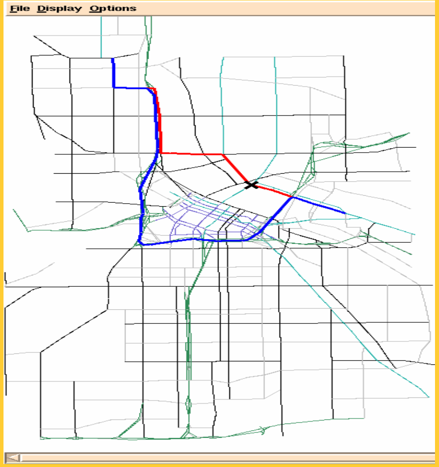


(b) River Network

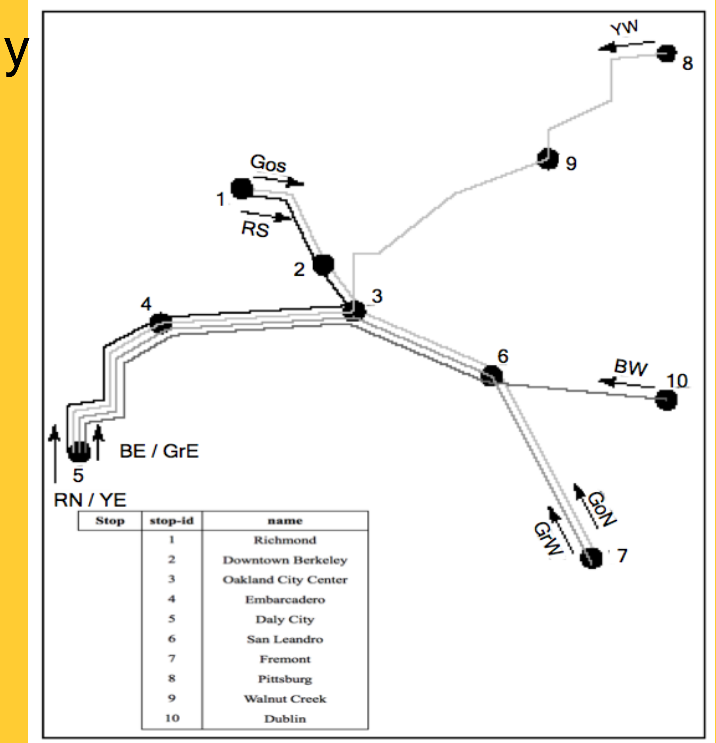
1. List the names of all direct and indirect tributaries of Mississippi river
2. List the direct tributaries of Colorado
3. Which rivers could be affected if there is a spill in North Platte river

Spatial Networks: Three Examples

A Road Network



A Railway Network



Stop	stop-id	name
	1	Richmond
	2	Downtown Berkeley
	3	Oakland City Center
	4	Embarcadero
	5	Daly City
	6	San Leandro
	7	Fremont
	8	Pittsburg
	9	Walnut Creek
	10	Dublin

A River Network

Colorado
Mississippi

(b) River Network

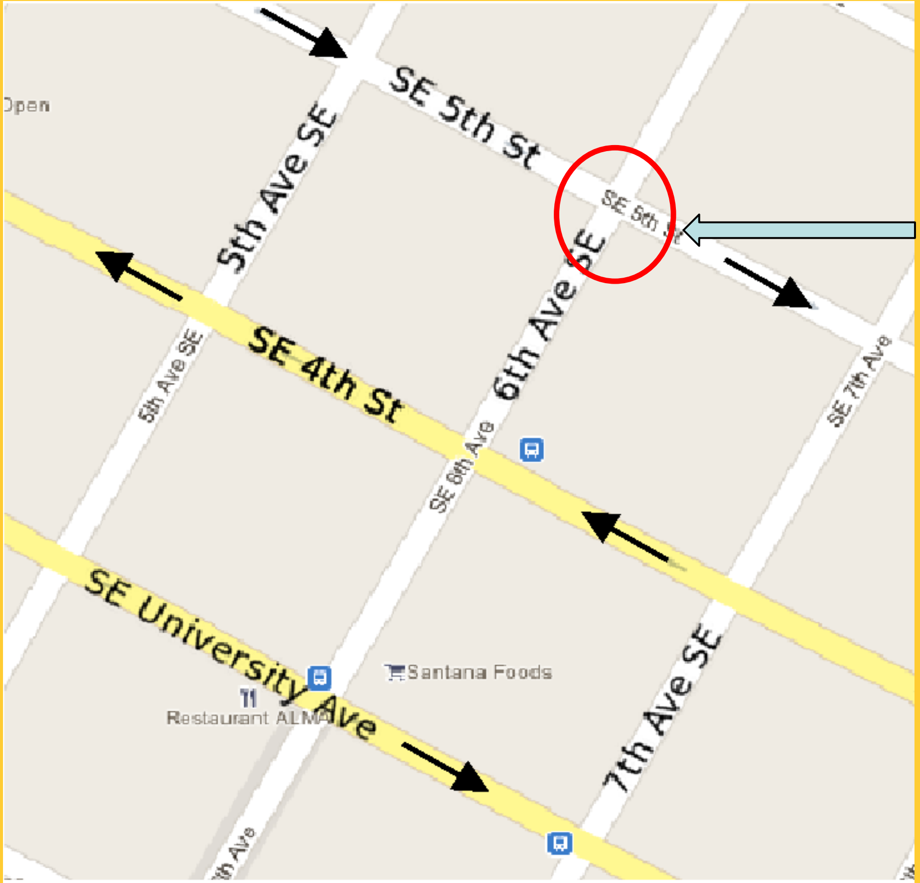
Outline

1. Motivation, and use cases
2. Example spatial networks
3. Conceptual model
4. Need for SQL extensions
5. CONNECT statement
6. RECURSIVE statement
7. Storage and data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

Data Models of Spatial Networks

1. Conceptual Model
   - Information Model: Entity Relationship Diagrams
   - Mathematical Model: Graphs
2. Logical Data Model
   - Abstract Data types
   - Custom Statements in SQL
3. Physical Data Model
   - Storage-Structures
   - Algorithms for common operations

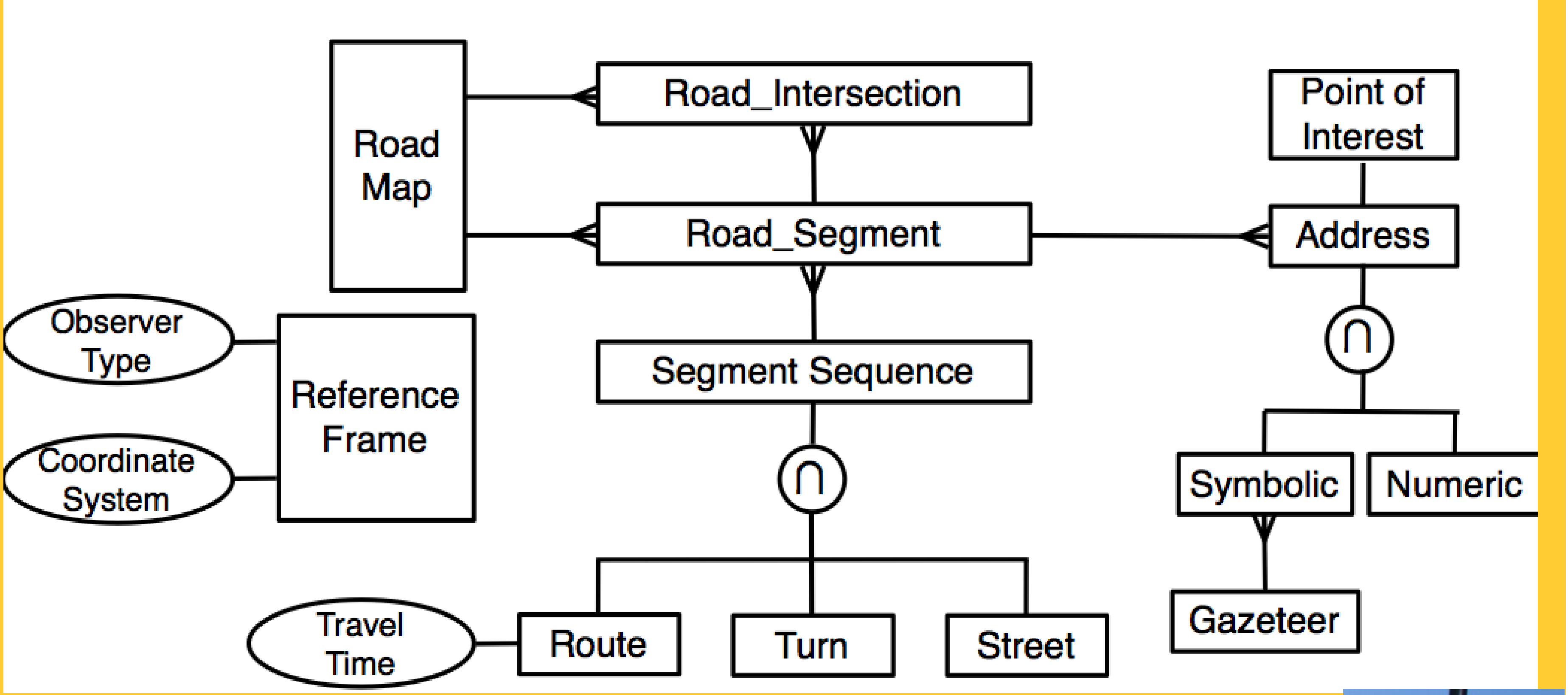
Modeling Roadmaps



Many Concepts, e.g.

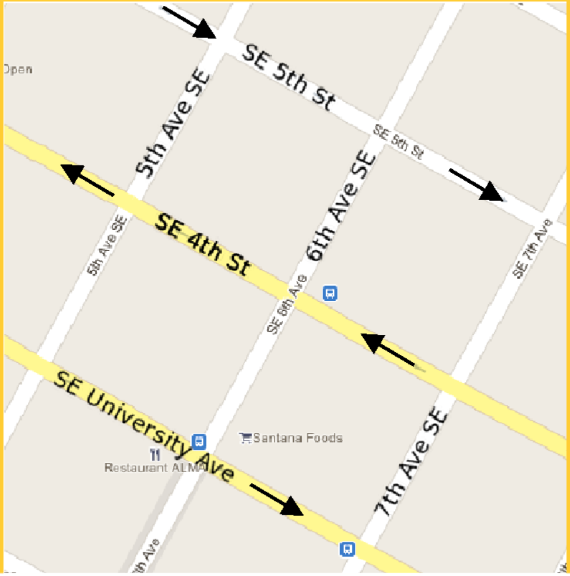
- Roads (or streets, avenues)
- Road-Intersections
- Road-Segments
- Turns
- …

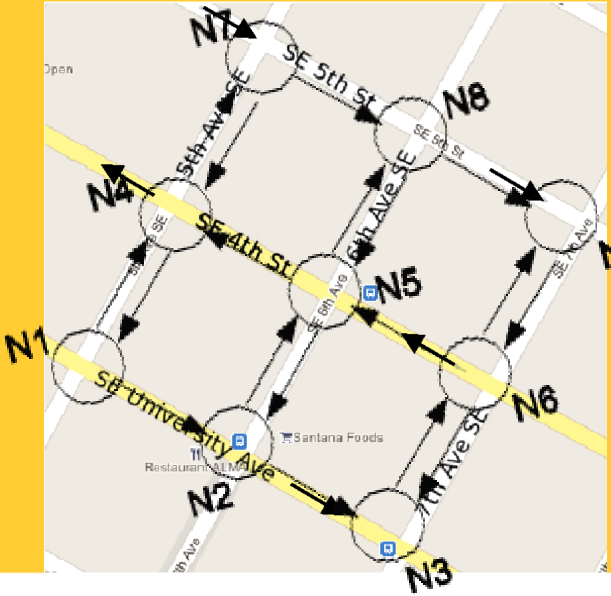
An Entity Relationship Diagram

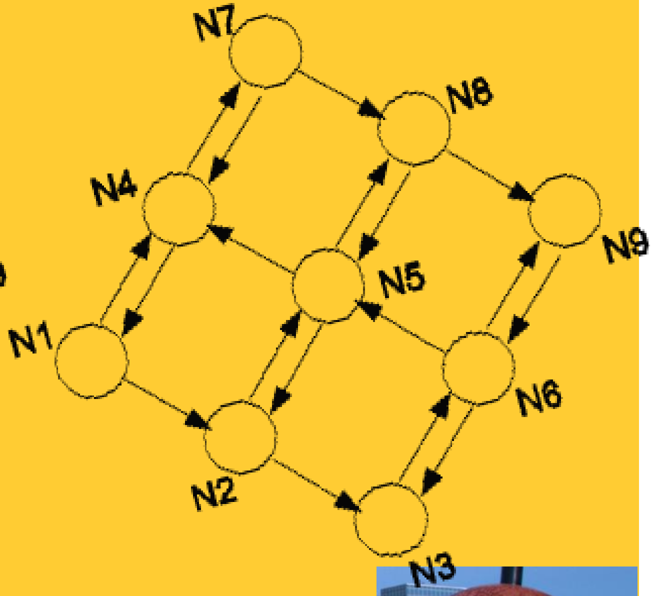


Graph Models

- A Simple Mathematical Model
  - A graph G = (V,E)
  - V = a finite set of vertices
  - E = a set of edges model a binary relationship between vertices
- Example

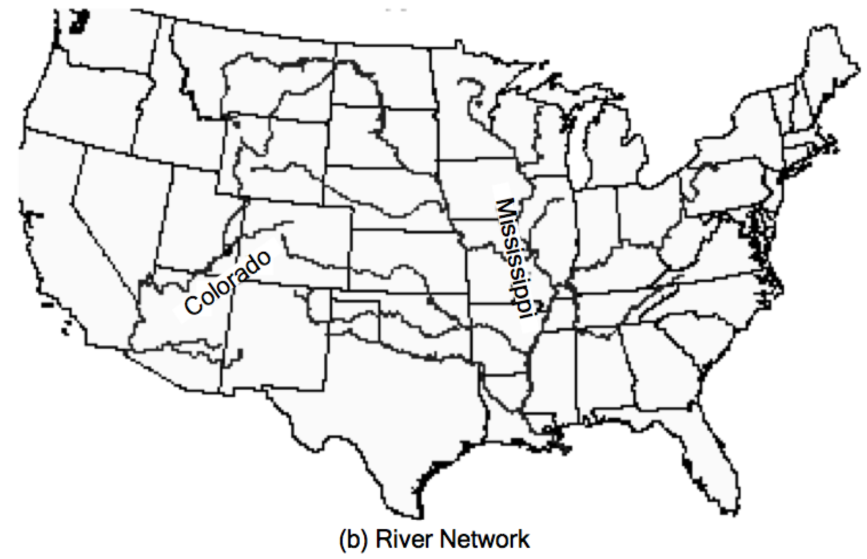




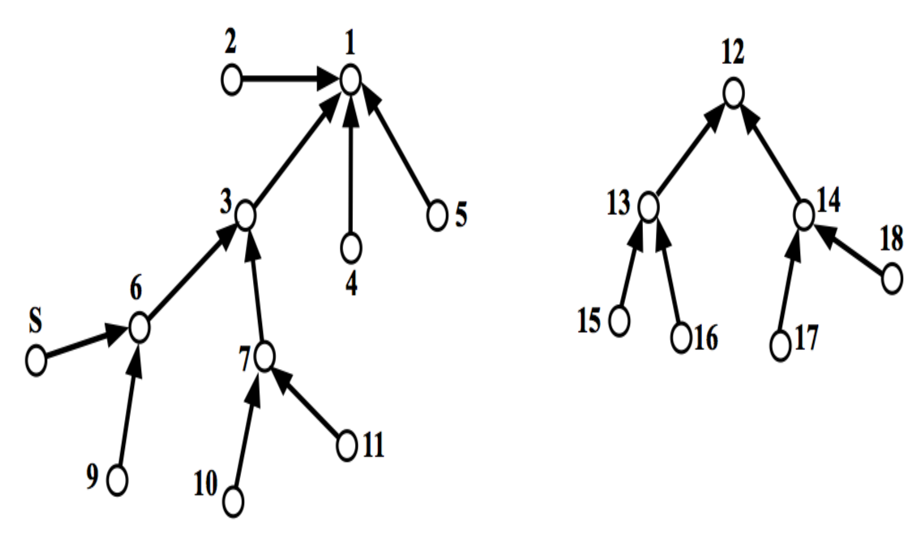


A Graph Model of River Network

- Nodes = rivers
- Edges = A river falls into another river



(b) River Network



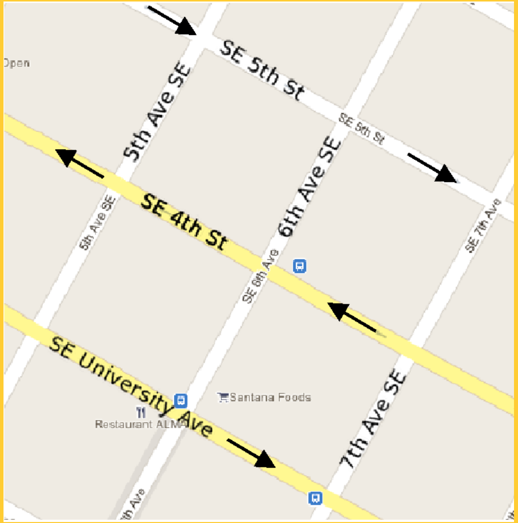
Pros and Cons of Graph Models

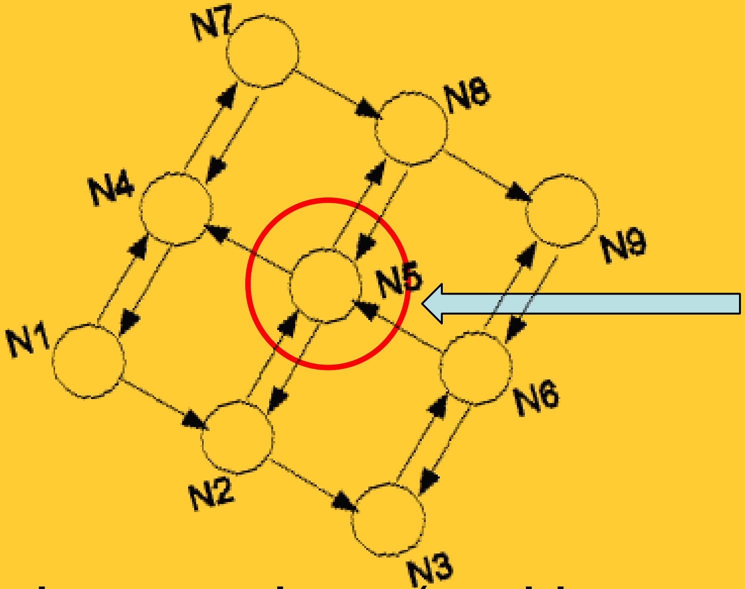
- Strength
  - Well developed mathematics for reasoning
  - Rich set of computational algorithms and data-structures
- Weakness
  - Models only one binary relationship

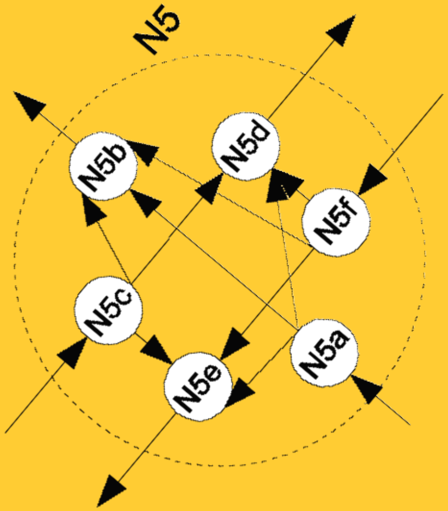
- Implications
  - A. Difficult to model multiple relationships, e.g., connect, turn
  - B. Multiple graph models possible for a spatial network

Modeling Turns in Roadmaps

- Approach 1: Model turns as a set of connects







- Approach 2: Use hyper-edges (and hyper-graphs)
- Approach 3: Annotate graph node with turn information

Alternative Graph Models for Roadmaps

- Choice 1:
  - Nodes = road-intersections
  - Edge (A, B) = road-segment connects adjacent road-intersections A, B
- Choice 2:
  - Nodes = (directed) road-segments
  - Edge (A,B) = turn from road-segment A to road-segment B
- Choice 3:
  - Nodes = roads
  - Edge(A,B) = road A intersects_with road B

Quiz

Which of the following are usually not captured in common graph models of roadmaps?

a) Turn restrictions (e.g., no U turn)

b) Road intersections

c) Road segments

d) All of the above

Outline

1. Motivation, and use cases
2. Example spatial networks
3. Conceptual model
4. Need for SQL extensions
5. CONNECT statement
6. RECURSIVE statement
7. Storage and data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

Data Models of Spatial Networks

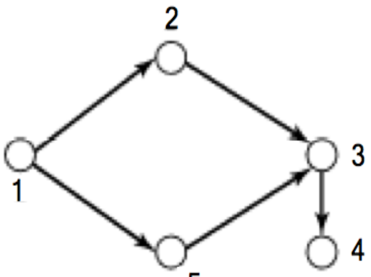
1. Conceptual Model: Entity Relationship Diagrams, Graphs
2. Logical Data Model & Query Languages
   - Abstract Data types
   - Custom Statements in SQL
3. Physical Data Model: Storage-Structures, Algorithms

Transitive Closure

- Consider a graph G = (V, E)
- Transitive closure(G) = G* = (V*, E*), where
  - V* = V
  - (A, B) in E* if and only if there is a path from A to B in G.

Transitive Closure - Example

- Example
  - G has 5 nodes and 5 edges
  - G* has 5 nodes and 9 edges
  - Note edge (1,4) in G* for
    - path (1, 2, 3, 4) in G.



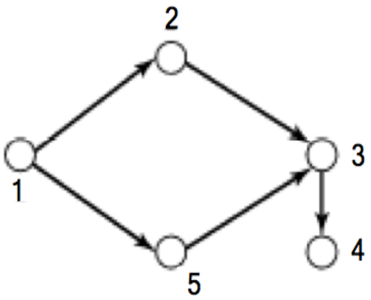
(a) Graph G

(c) Transitive closure (G) = Graph G

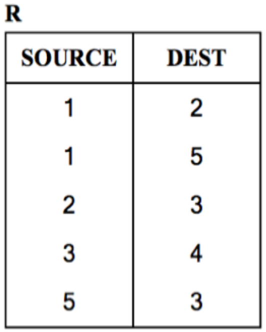
(d) Transitive closure in relation form

Transitive Closure - Example

- Example
  - G has 5 nodes and 5 edges
  - G* has 5 nodes and 9 edges
  - Note edge (1,4) in G* for
    - path (1, 2, 3, 4) in G.



(a) Graph G


R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

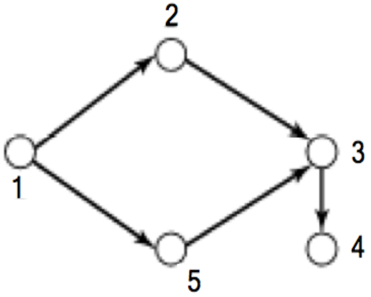
(b) Relation form

(c) Transitive closure (G) = Graph G

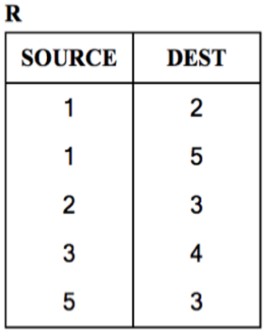
(d) Transitive closure in relation form

Transitive Closure - Example

- Example
  - G has 5 nodes and 5 edges
  - G* has 5 nodes and 9 edges
  - Note edge (1,4) in G* for
    - path (1, 2, 3, 4) in G.



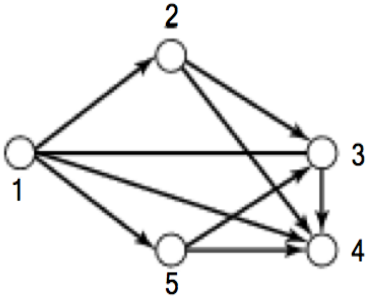
(a) Graph G



R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form

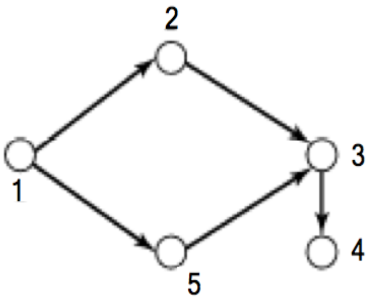


(c) Transitive closure (G) = Graph G

(d) Transitive closure in relation form

Transitive Closure - Example

- Example
  - G has 5 nodes and 5 edges
  - G* has 5 nodes and 9 edges
  - Note edge (1,4) in G* for
    - path (1, 2, 3, 4) in G.

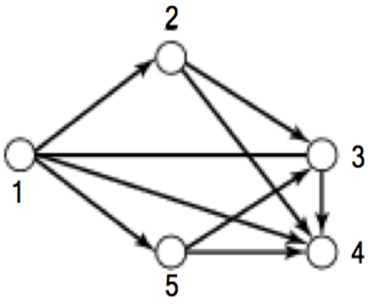


(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form



(c) Transitive closure (G) = Graph G

X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4
1	4

(d) Transitive closure in relation form

Limitations of Original SQL

- Recall Relation algebra based languages
  - Ex. Original SQL
  - Can not compute transitive closure, e.g., shortest path!

Supporting Graphs in SQL

- Abstract Data Type (user defined)
  - SQL3
  - May include shortest path operation!

- Custom Statements
  - SQL2 - CONNECT clause in SELECT statement
    - For directed acyclic graphs, e.g. hierarchies
  - SQL3 - WITH RECURSIVE statement
    - Transitive closure on general graphs

Outline

1. Motivation, and use cases
2. Example spatial networks
3. Conceptual model
4. Need for SQL extensions
5. CONNECT statement
6. RECURSIVE statement
7. Storage and data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

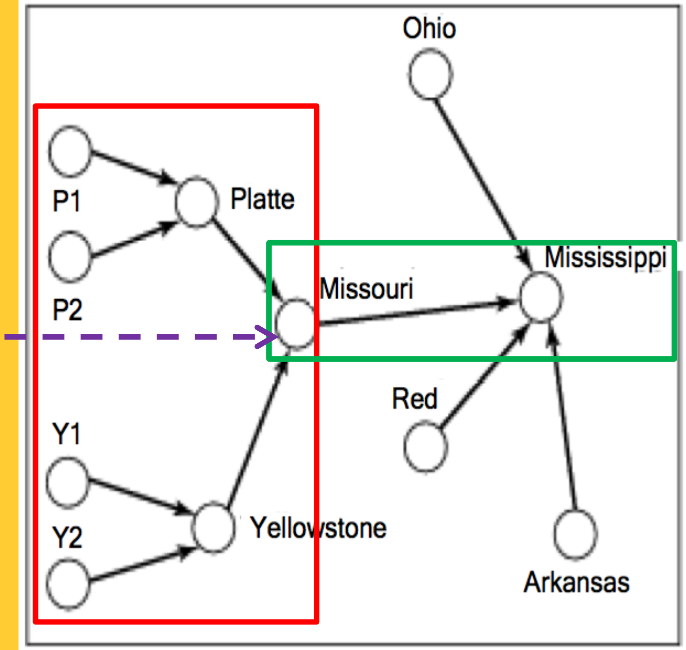
Querying Graphs: Overview

- Relational Algebra
  - Can not express transitive closure queries

- Two ways to extend SQL to support graphs
  1. Abstract Data Types
  2. Custom Statements
     - SQL2 - CONNECT BY clause(s) in SELECT statement
     - SQL3 - WITH RECURSIVE statement

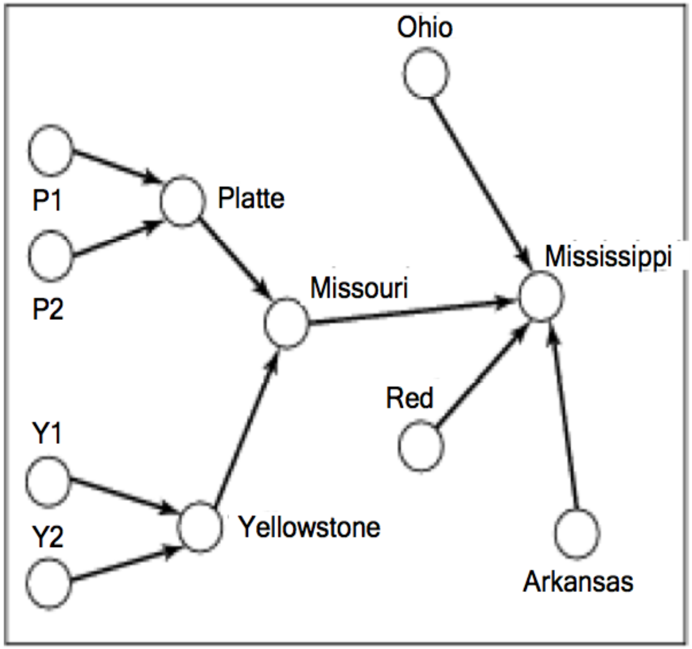
CONNECT BY : Input, Output

- Input: (a) Edges of a directed acyclic graph G
  - (b) Start Node S, e.g., Missouri
  - (c) Travel Direction
- Output: Transitive closure of G
  - Ex. Predecessors of S = Missouri
  - Ex. Successors of S = Missouri



(a) Mississippi network (Y1 = Bighorn river, Y2 = Power river, P1 = Sweet water River, P2 = Big Thompson river)

Directed Edges: Tabular Representation



(a) Mississippi network (Y1 = Bighorn river,
Y2 = Power river, P1 = Sweet water River,
P2 = Big Thompson river)

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

CONNECT BY- PRIOR - START WITH

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest =“Missouri”
```

Table: Falls_Into

Source	**Dest**
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

CONNECT BY- PRIOR - START WITH

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest =“Missouri”
```

Q? What does CONNECT BY … PRIOR specify?

- Direction of travel
- Example: From Dest to Source
- Alternative: From Source to Dest

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

CONNECT BY- PRIOR - START WITH

Choice 1: Travel from Dest to Source

Ex. List direct & indirect tributaries of Missouri.

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest =“Missouri”
```

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

CONNECT BY- PRIOR - START WITH

Choice 1: Travel from Dest to Source

Ex. List direct & indirect tributaries of Missouri.

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest =“Missouri”
```

Choice 2: Travel from Source to Dest

Ex. Which rivers are affected by spill in Missouri?

```
SELECT  dest
FROM      Falls_Into
CONNECT BY source = PRIOR dest
START WITH  source =“Missouri”
```

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

Execution Trace - Step 1

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest = Missouri
```

1. Root Result = SELECT * FROM Falls_Into
   WHERE (dest = Missouri)

Table: "Root Result "

Source	Dest
Platte	Missouri
Yellowstone	Missouri

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

Execution Trace - Step 2.

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest = Missouri
```

2. Add rows from Falls_Into where (Root_Result.source = Falls_Into.dest) to create table <Root + 1 level children>

Root Result

Source	Dest
Platte	Missouri
Yellowstone	Missouri

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

Execution Trace - Step 2.

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest = Missouri
```

2. Add rows from Falls_Into where (Root_Result.source = Falls_Into.dest) to create table <Root + 1 level children>

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

Root Result

Source	Dest
Platte	Missouri
Yellowstone	Missouri

Execution Trace - Step 2.

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest = Missouri
```

2. Add rows from Falls_Into where (Root_Result.source = Falls_Into.dest) to create table <Root + 1 level children>

Root + 1 level children

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri

Seed Result

Source	Dest
Platte	Missouri
Yellowstone	Missouri

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

Execution Trace - Step 2.

```
SELECT  source
FROM      Falls_Into
CONNECT BY PRIOR  source = dest
START WITH  dest = Missouri
```

2. Add rows from Falls_Into where (Root+1levchild.source = Falls_Into.dest) to create table <Root + 2 level children>

No new rows can be added into the result!

Root + 1 level children

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

Execution Trace - Step 2.

```
SELECT  source
FROM      Falls  Into
CONNECT BY PRIOR  source = dest
START WITH  dest = Missouri
```

2. Add rows from Falls_Into where (Root+1lchild.source = Falls_Into.dest) to create table <Root + 2 level children>

Table: Falls_Into

Source	Dest
P1	Platte
P2	Platte
Y1	Yellowstone
Y2	Yellowstone
Platte	Missouri
Yellowstone	Missouri
Missouri	Mississippi
Ohio	Mississippi
Red	Mississippi
Arkansas	Mississippi

Final answer

Source
P1
P2
Y1
Y2
Platte
Yellowstone

The query returned all predecessors of Missouri!

Quiz

Which of the following is false about CONNECT BY clause?

a) It is only able to output predecessors, but not successors, of the start node

b) It is able to output transitive closure of a directed graph

c) It usually works with PRIOR and START WITH keywords

d) None of the above

Outline

1. Motivation, and use cases
2. Example spatial networks
3. Conceptual model
4. Need for SQL extensions
5. CONNECT statement
6. RECURSIVE statement
7. Storage and data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

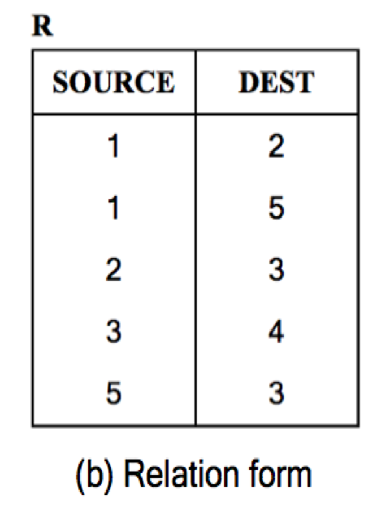
Querying Graphs: Overview

- Relational Algebra
  - Can not express transitive closure queries

- Two ways to extend SQL to support graphs
  1. Abstract Data Types
  2. Custom Statements
     - SQL2 - CONNECT clause(s) in SELECT statement
     - SQL3 - WITH RECURSIVE statement

WITH RECURSIVE: Input, Output

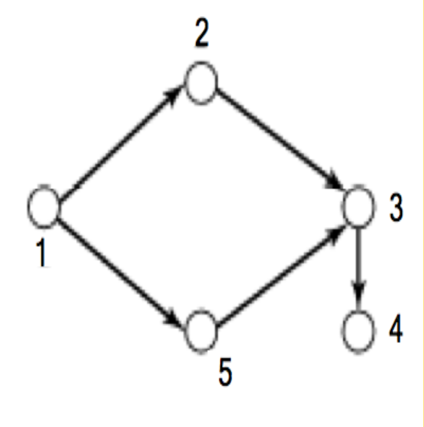
- Input:
  - (a) Edges of a directed graph G
  - (b) Sub-queries to
    - Initialize results
    - Recursively grow results
    - Additional constraints



R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form



(a) Graph G

- Output: Transitive closure of G
  - Ex. Predecessors of a node
  - Ex. Successors of a node

Syntax of WITH RECURSIVE Statement

```
WITH RECURSIVE  X(source,dest)
AS  (SELECT source,dest FROM R )
          UNION
    (SELECT R.source,  X.dest
          FROM R,  X
          WHERE R.dest=X.source )
```

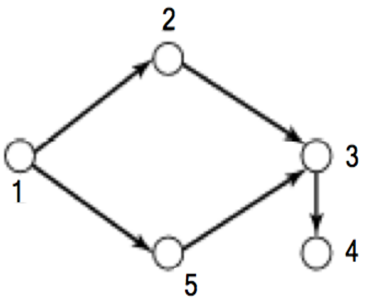
Description of Result Table

Initialization Query

Recursive Query to grow result

Example Input and Output

```
WITH RECURSIVE  X(source,dest)
AS  (SELECT source,dest FROM R )
            UNION
      (SELECT R.source,  X.dest
            FROM R,  X
            WHERE R.dest=X.source )
```

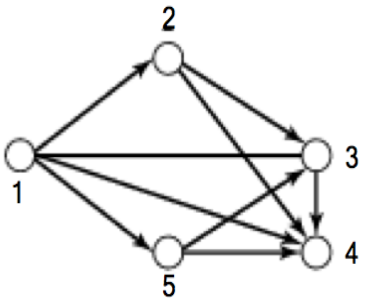


(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form



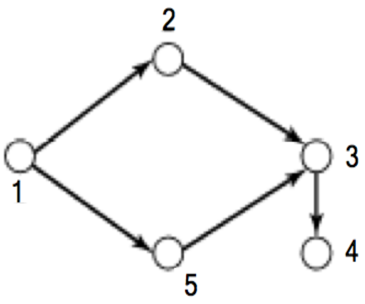
(c) Transitive closure (G) = Graph G

X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4
1	4

(d) Transitive closure in relation form

SQL3 Recursion Example - Meaning



(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form

X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(c) Transitive closure (G) = Graph G

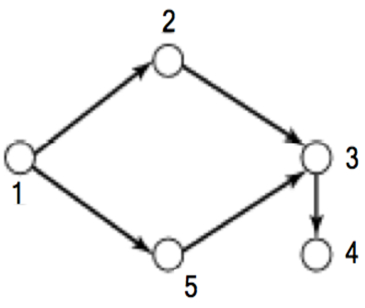
(d) Transitive closure in relation form

- Initialize X by
  (**SELECT** source,dest **FROM** R)

- Recursively grow X by
  (**SELECT** R.source, X.dest
  FROM R, X
  WHERE R.dest=X.source)

- Infer X(a,c) from R(a,b),X(b,c)

SQL3 Recursion Example - Meaning



(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form

X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3

(c) Transitive closure (G) = Graph G

(d) Transitive closure in relation form

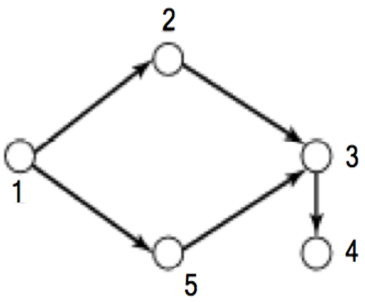
- Initialize X by
  (**SELECT** source,dest **FROM** R)

- Recursively grow X by
  (**SELECT** R.source, X.dest
  FROM R, X
  WHERE R.dest=X.source)

- Infer X(a,c) from R(a,b),X(b,c)

- Infer X(1,3) from R(1,2),X(2,3)

SQL3 Recursion Example - Meaning



(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form

X

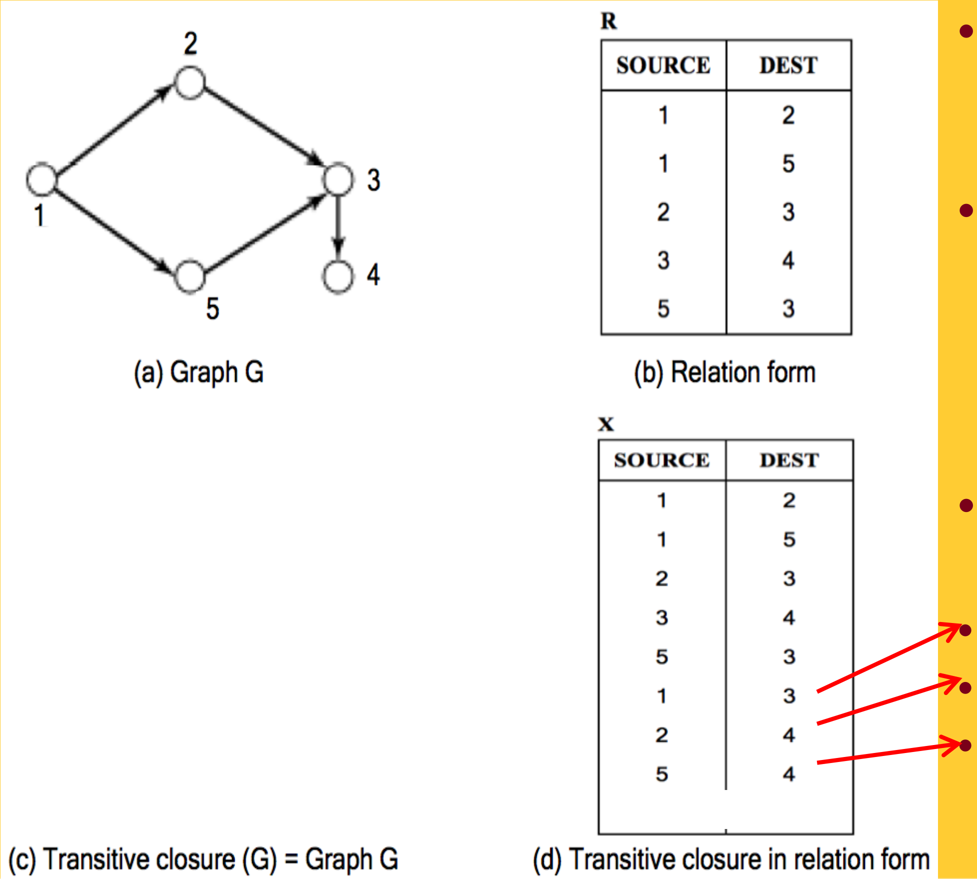
SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4

(c) Transitive closure (G) = Graph G

(d) Transitive closure in relation form

- Initialize X by
  (**SELECT** source,dest **FROM** R)
- Recursively grow X by
  (**SELECT** R.source, X.dest
  FROM R, X
  WHERE R.dest=X.source)
- Infer X(a,c) from R(a,b),X(b,c)
- Infer X(1,3) from R(1,2),X(2,3)
- Infer X(2,4) from R(2,3),X(3,4)

SQL3 Recursion Example - Meaning



(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form

X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4

(c) Transitive closure (G) = Graph G

(d) Transitive closure in relation form

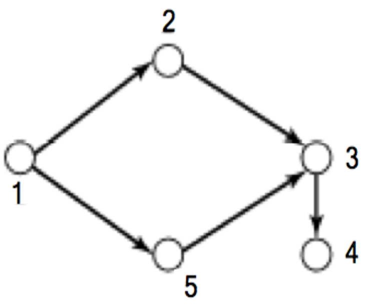
- Initialize X by
  (**SELECT** source,dest **FROM** R)

- Recursively grow X by
  (**SELECT** R.source, X.dest
  FROM R, X
  WHERE R.dest=X.source)

- Infer X(a,c) from R(a,b),X(b,c)

- Infer X(1,3) from R(1,2),X(2,3)
- Infer X(2,4) from R(2,3),X(3,4)
- Infer X(5,4) from R(5,3),X(3,4)

SQL3 Recursion Example - Meaning



(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form

X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4
1	4

(c) Transitive closure (G) = Graph G

(d) Transitive closure in relation form

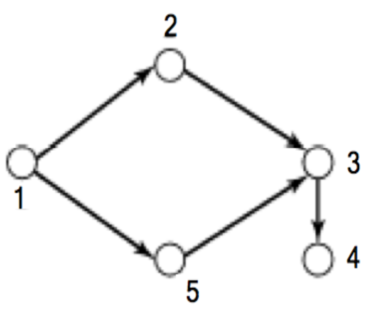
- Initialize X by
  (**SELECT** source,dest **FROM** R)

- Recursively grow X by
  (**SELECT** R.source, X.dest
  FROM R, X
  WHERE R.dest=X.source)

- Infer X(a,c) from R(a,b),X(b,c)

- Infer X(1,3) from R(1,2),X(2,3)
- Infer X(2,4) from R(2,3),X(3,4)
- Infer X(5,4) from R(5,3),X(3,4)
- Infer X(1,4) from R(1,5),X(5,4)

SQL3 Recursion Example - Meaning

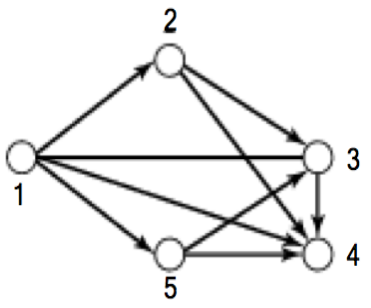


(a) Graph G

R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

(b) Relation form



(c) Transitive closure (G) = Graph G

X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4
1	4

(d) Transitive closure in relation form

- Initialize X by
  (**SELECT** source,dest **FROM** R)

- Recursively grow X by

```
(SELECT R.source,  X.dest
        FROM R,  X
        WHERE  R.dest=X.source )
```

- Infer X(a,c) from R(a,b),X(b,c)

- Infer X(1,3) from R(1,2),X(2,3)
- Infer X(2,4) from R(2,3),X(3,4)
- Infer X(5,4) from R(5,3),X(3,4)
- Infer X(1,4) from R(1,5),X(5,4)

Quiz

Which of the following are true about WITH RECURSIVE clause?

a) It is able to output transitive closure of a directed graph

b) It usually works with an edge table

c) It includes two SELECT statements

d) All of the above

Outline

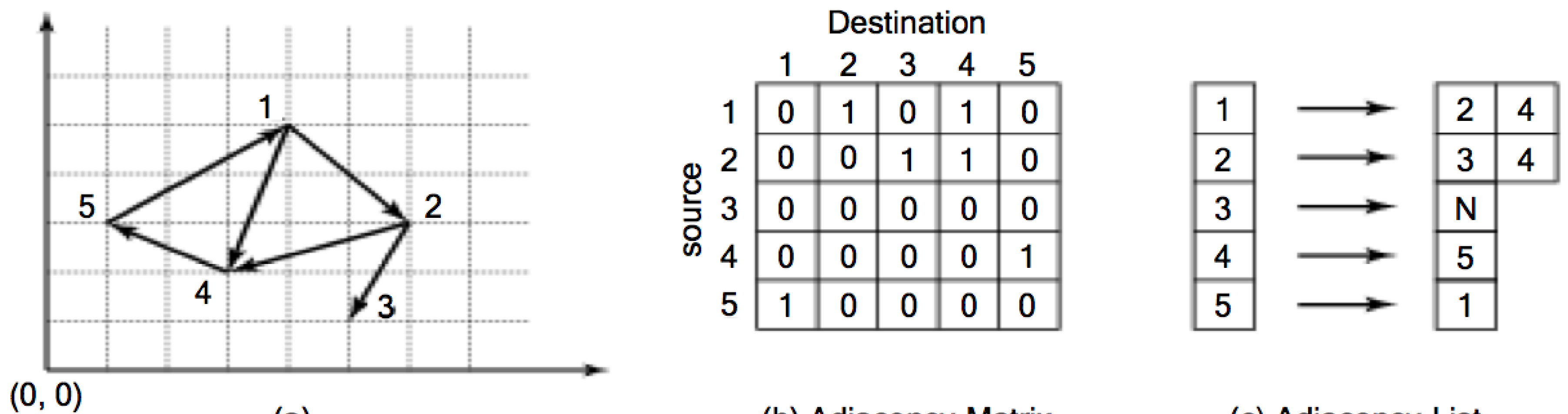
1. Where do we use spatial networks?
2. Example spatial networks
3. Conceptual model of spatial networks
4. Why do we need SQL extensions
5. CONNECT clause
6. RECURSIVE statement
7. Data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

Data Models of Spatial Networks

1. Conceptual Model : Entity Relationship Diagrams, Graphs
2. Logical Data Model : Abstract Data types , Custom Statements in SQL
3. Physical Data Model
   - Storage: Data-Structures, File-Structures
   - Algorithms for common operations

Main Memory Data-Structures

- Adjacency matrix
  - M[A, B] = 1 if and only if edge(vertex A, vertex B) exists
- Adjacency list :
  - maps a vertex to a list of its successors



(a)

Destination

source	1	2	3	4	5
1	0	1	0	1	0
2	0	0	1	1	0
3	0	0	0	0	0
4	0	0	0	0	1
5	1	0	0	0	0

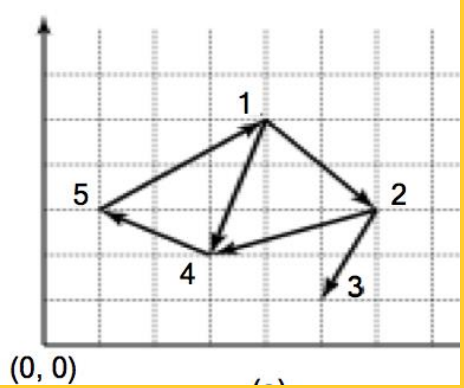
(b) Adjacency-Matrix

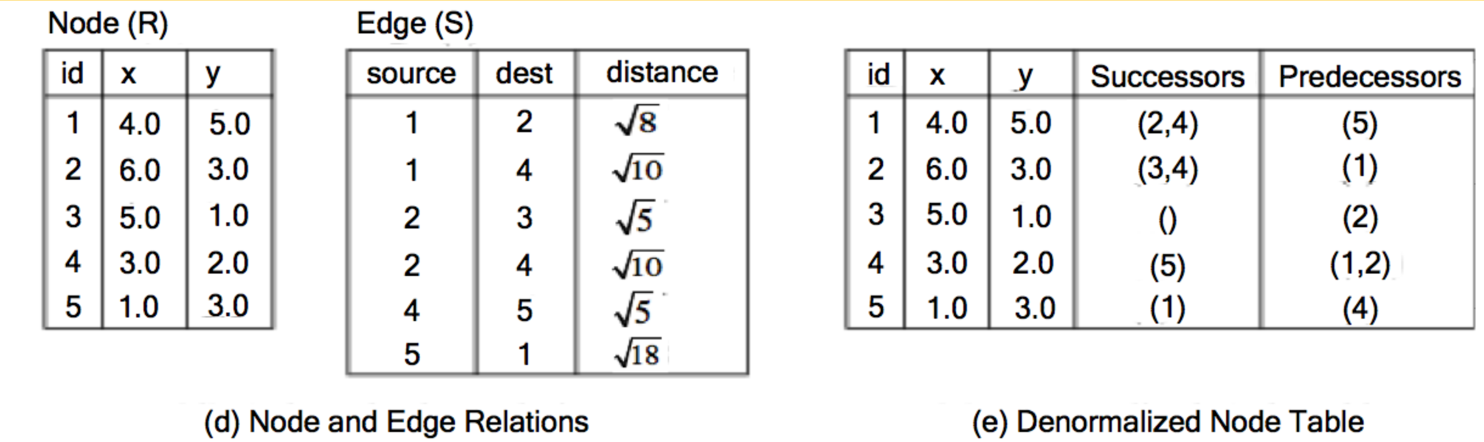
1	→	2	4
2	→	3	4
3	→	N	
4	→	5	
5	→	1	

(c) Adjacency-List

Disk-based Tables

- Normalized tables
  - one for vertices, other for edges
- Denormalized
  - one table for nodes with adjacency lists





Node (R)

id	x	y
1	4.0	5.0
2	6.0	3.0
3	5.0	1.0
4	3.0	2.0
5	1.0	3.0

Edge (S)

source	dest	distance
1	2	$\sqrt{8}$
1	4	$\sqrt{10}$
2	3	$\sqrt{5}$
2	4	$\sqrt{10}$
4	5	$\sqrt{5}$
5	1	$\sqrt{18}$

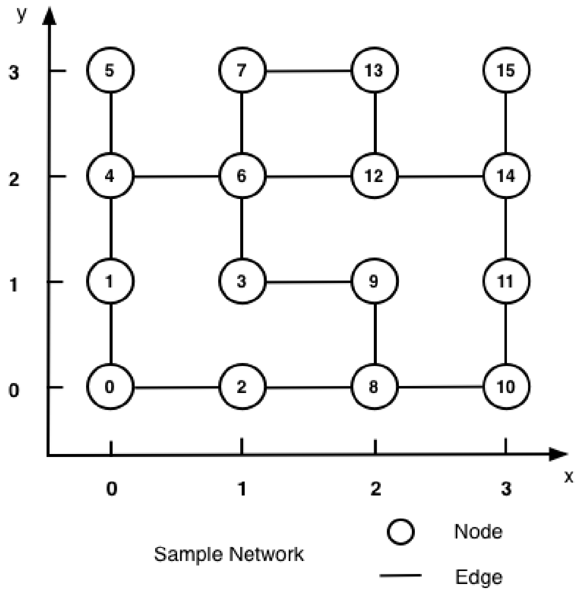
(d) Node and Edge Relations

id	x	y	Successors	Predecessors
1	4.0	5.0	(2,4)	(5)
2	6.0	3.0	(3,4)	(1)
3	5.0	1.0	()	(2)
4	3.0	2.0	(5)	(1,2)
5	1.0	3.0	(1)	(4)

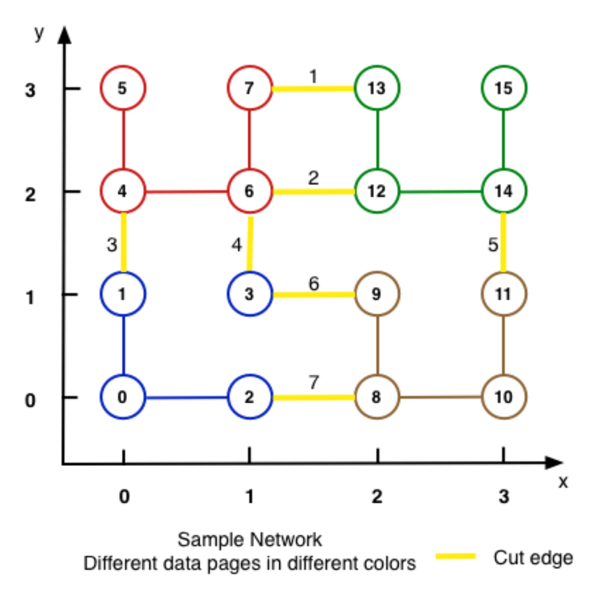
(e) Denormalized Node Table

File-Structures:
Partition Graph into Disk Blocks

- Which partitioning reduces disk I/O for graph operations?
    - Choice 1: Geometric partition



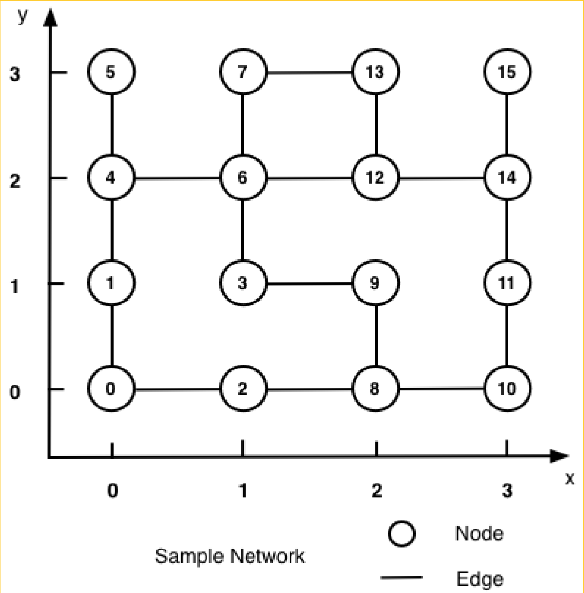
Sample Network



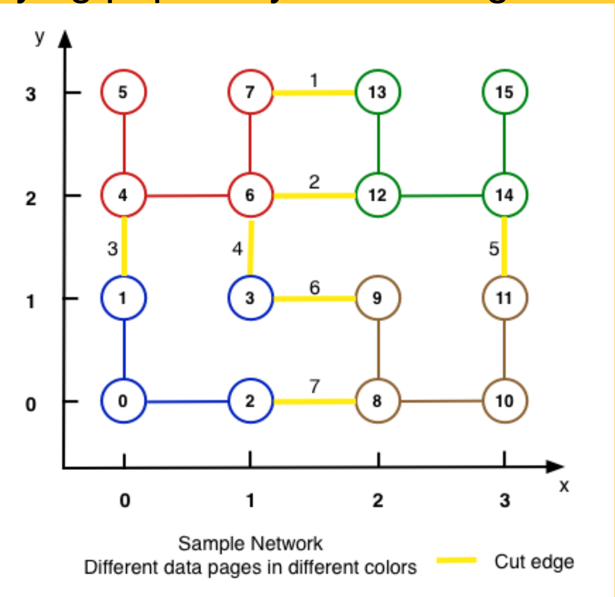
Sample Network
Different data pages in different colors

File-Structures:
Partition Graph into Disk Blocks

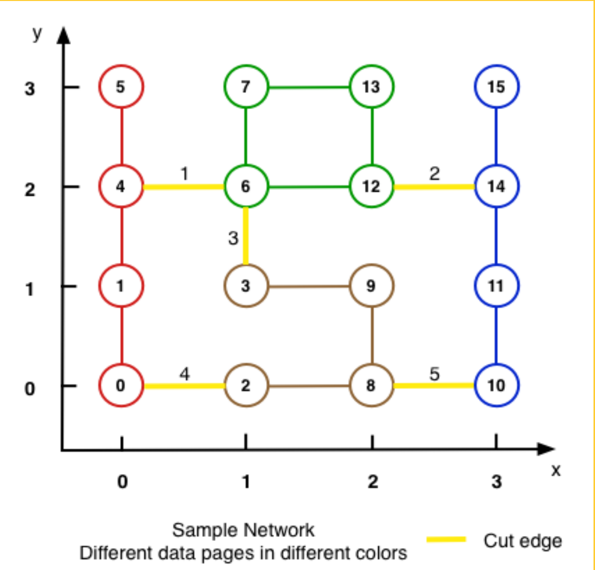
- Which partitioning reduces disk I/O for graph operations?
  - Choice 1: Geometric partition
  - Choice 2: min-cut Graph Partition
  - Choice 2 cuts fewer edges and is preferred
  - Assuming uniform querying popularity across edges



Sample Network



Sample Network
Different data pages in different colors



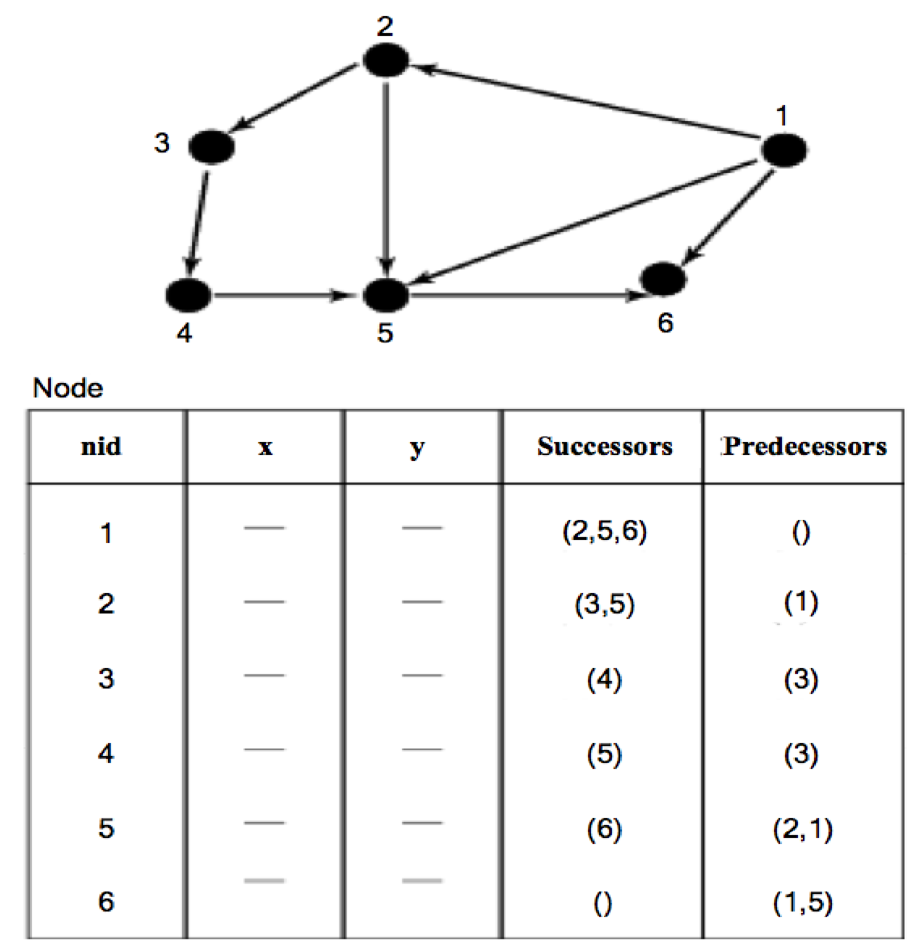
Sample Network
Different data pages in different colors

Spatial Computing
Research Group

Exercise: Graph Based Storage Methods

- Consider spatial network on right
- If a disk page holds 3 nodes, which partitioning will has fewest cut-edges?

  (a) (1, 2, 3), (4,5,6)
  (b) (2, 3, 4), (1, 5, 6)
  (c) (1, 2, 6), (3, 4, 5)
  (d) (1, 3, 5), (2, 4, 6)



Node

nid	x	y	Successors	Predecessors
1	—	—	(2,5,6)	()
2	—	—	(3,5)	(1)
3	—	—	(4)	(3)
4	—	—	(5)	(3)
5	—	—	(6)	(2,1)
6	—	—	()	(1,5)

Outline

1. Motivation, and use cases
2. Example spatial networks
3. Conceptual model
4. Need for SQL extensions
5. CONNECT statement
6. RECURSIVE statement
7. Storage and data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

Data Models of Spatial Networks

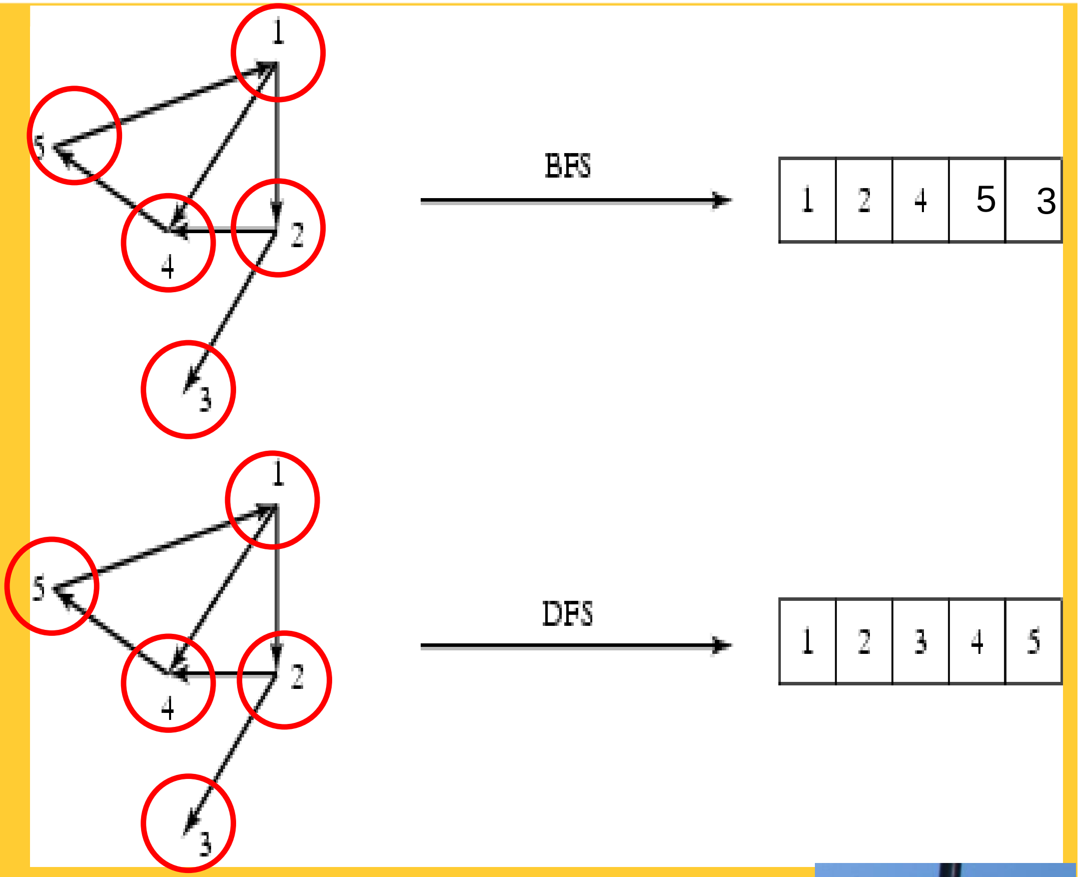
1. Conceptual Model : Entity Relationship Diagrams, Graphs
2. Logical Data Model : Abstract Data types , Custom Statements in SQL
3. Physical Data Model
    - Storage-Structures
    - Algorithms for common operations

Algorithms

- Main memory
  - Connectivity: Breadth first search, depth first search
  - Shortest path: Dijkstra's algorithm, A*
- Disk-based
  - Shortest path - Hierarchical routing algorithm

Algorithms for Connectivity Query

- Breadth first search
  - Visit descendent by generation
  - Children before grandchildren
  - Example: 1 - (2,4) - (3, 5)
- Depth first search
  - Try a path till dead-end
  - Backtrack to try different paths
  - Like a maze game
  - Example: 1-2-3-2-4-5
  - Note backtrack from 3 to 2



Quiz

Which of the following is false?

a) Breadth first search visits nodes layer (i.e. generation) by layer

b) Depth first search try a path till dead-end, then backtrack to try different paths

c) Depth first search always performs better than breadth first search

d) None of the above

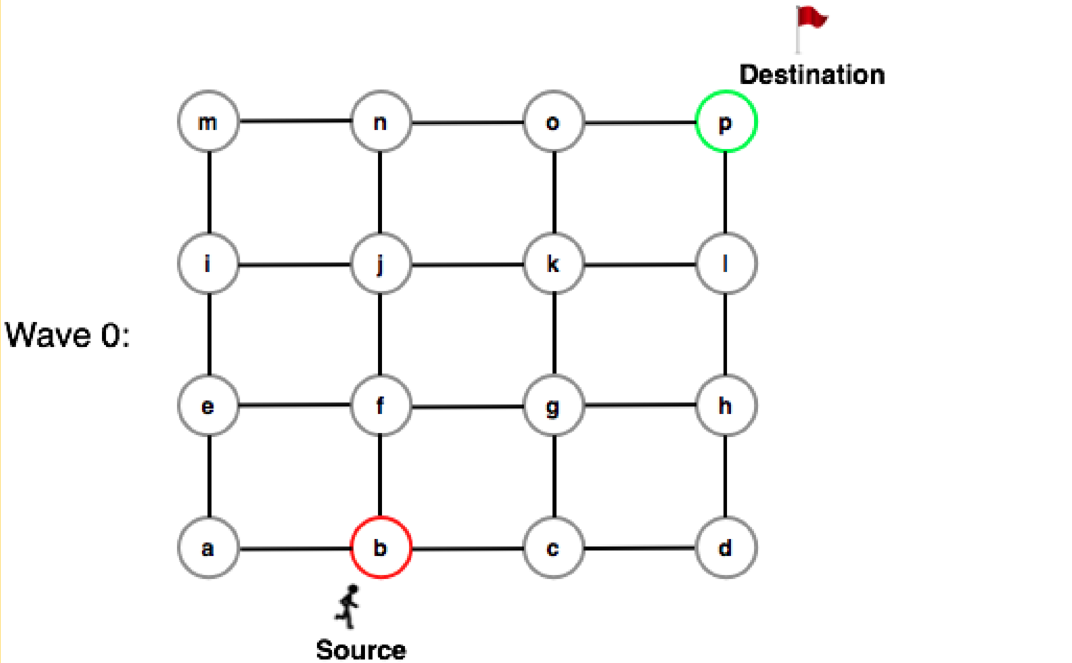
Outline

1. Motivation, and use cases
2. Example spatial networks
3. Conceptual model
4. Need for SQL extensions
5. CONNECT statement
6. RECURSIVE statement
7. Storage and data structures
8. Algorithms for connectivity query
9. Algorithms for shortest path

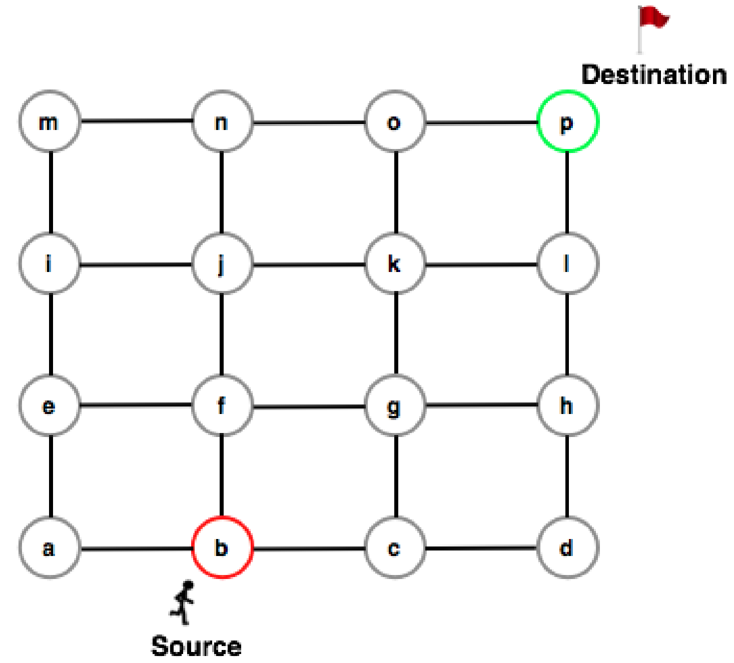
Shortest Path Algorithms

- Iterate
  - Expand most promising descent node
    - Dijkstra’s: try closest descendent to self
    - A* : try closest descendent to both destination and self
  - Update current best path to each node, if a better path is found
- Till destination node is expanded

Dijkstra’s vs. A*

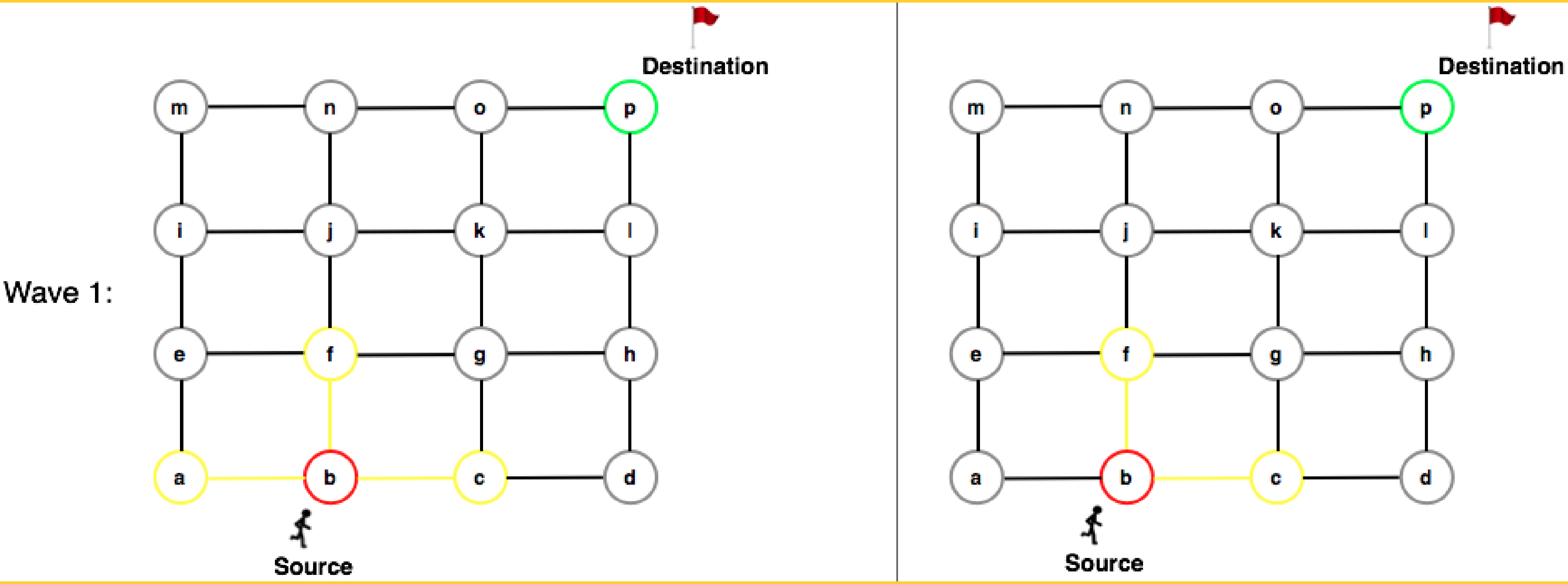


Dijkstra’s Algorithm



A* Algorithm

Dijkstra’s vs. A*

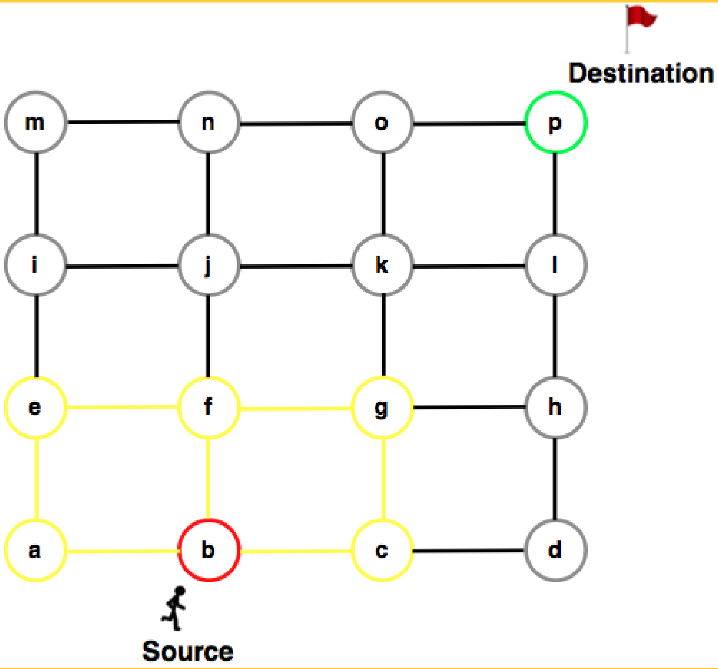


Dijkstra’s Algorithm

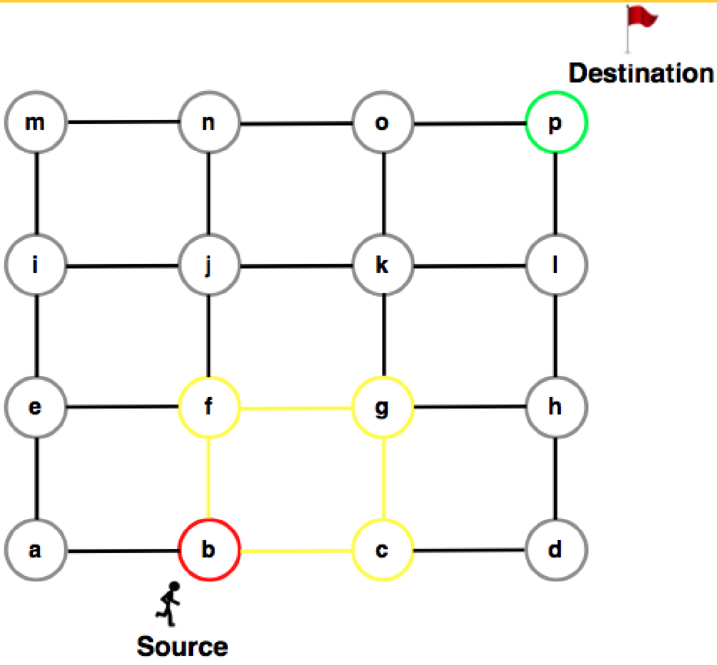
A* Algorithm

Dijkstra’s vs. A*

Wave 2:



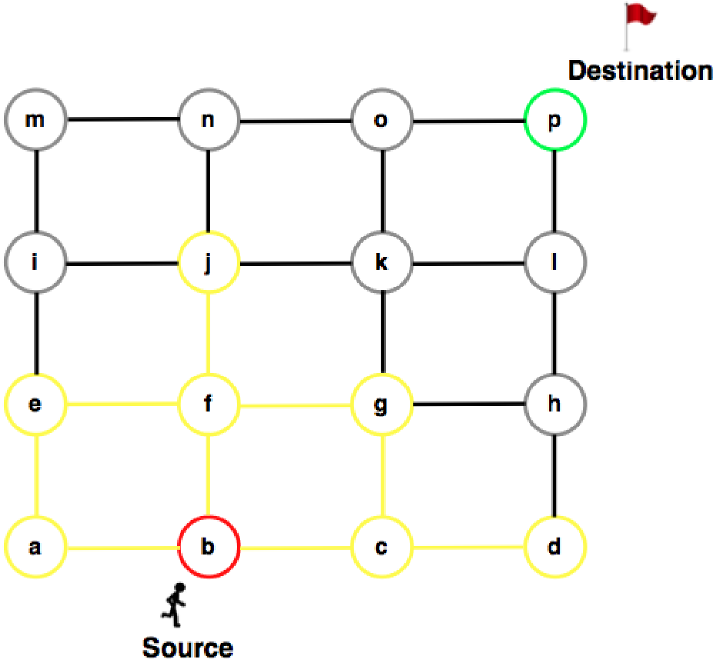
Dijkstra’s Algorithm



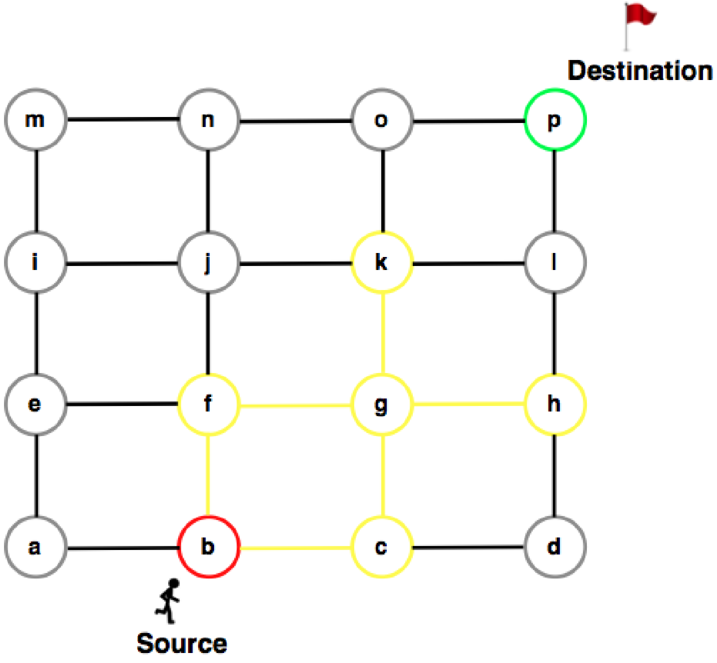
A* Algorithm

Dijkstra's vs. A*

Wave 3:



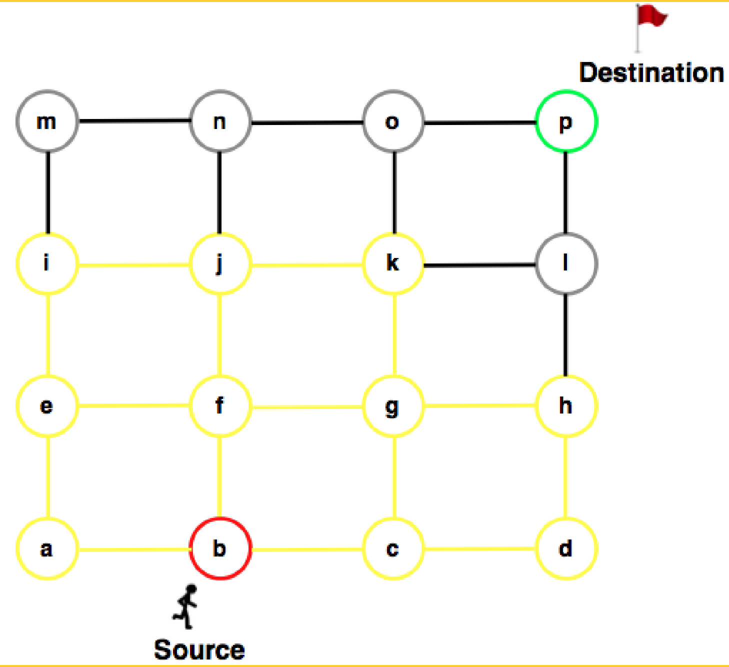
Dijkstra's Algorithm



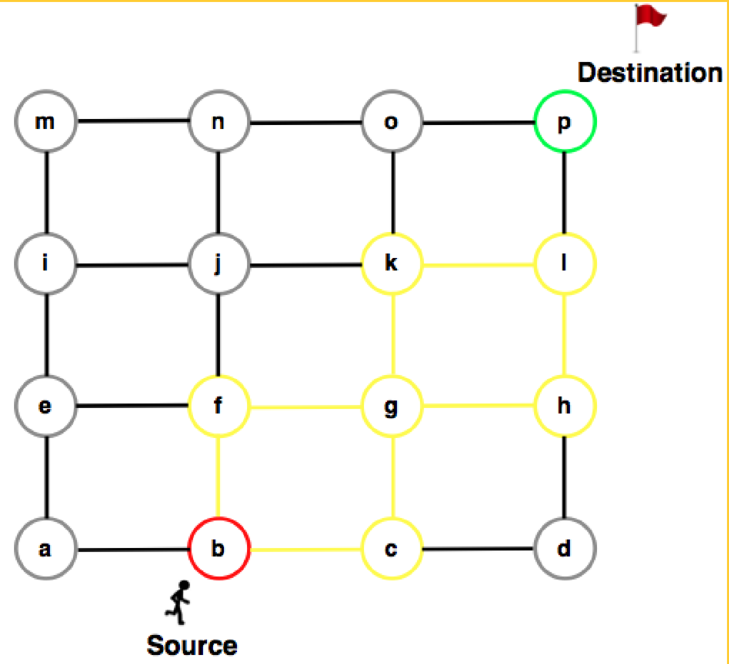
A* Algorithm

Dijkstra’s vs. A*

Wave 4:

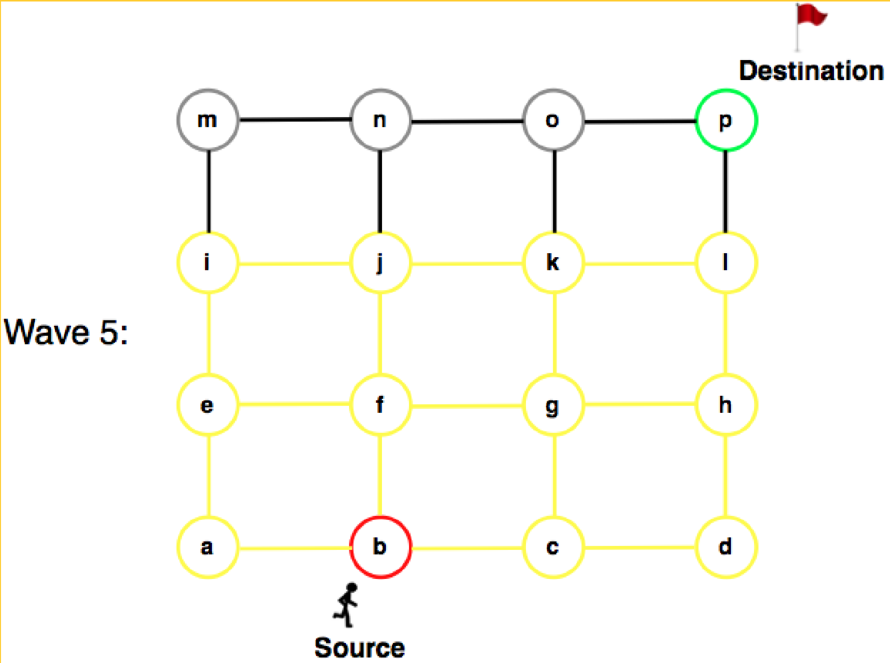


Dijkstra’s Algorithm

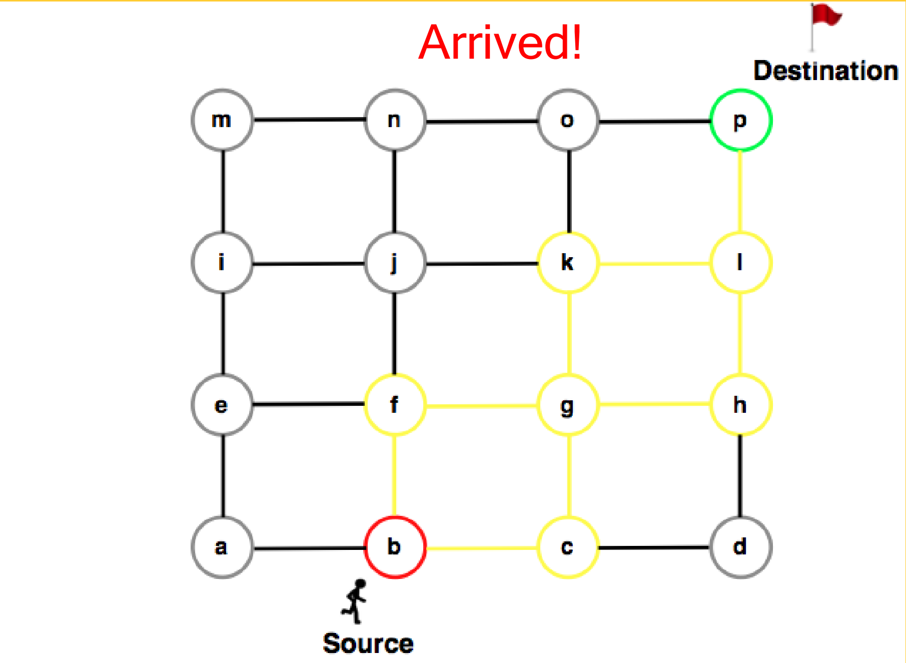


A* Algorithm

Dijkstra’s vs. A*



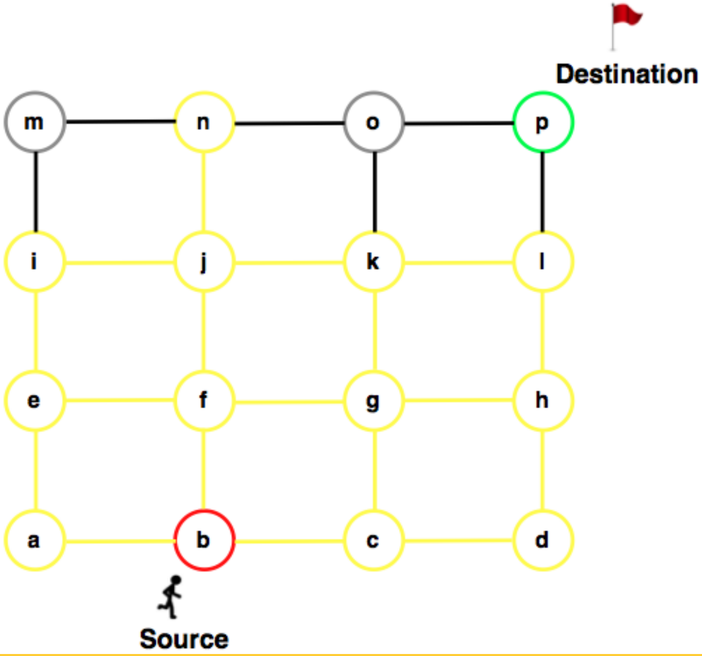
Dijkstra’s Algorithm



A* Algorithm

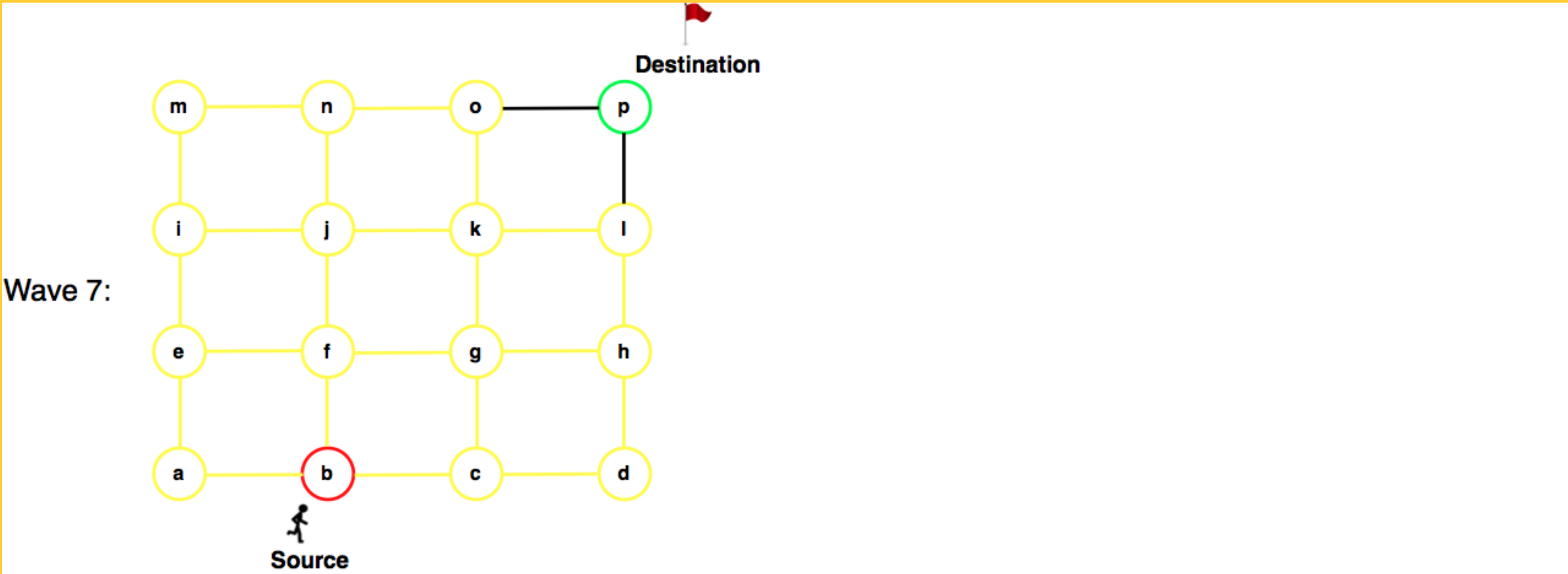
Dijkstra's vs. A*

Wave 6:



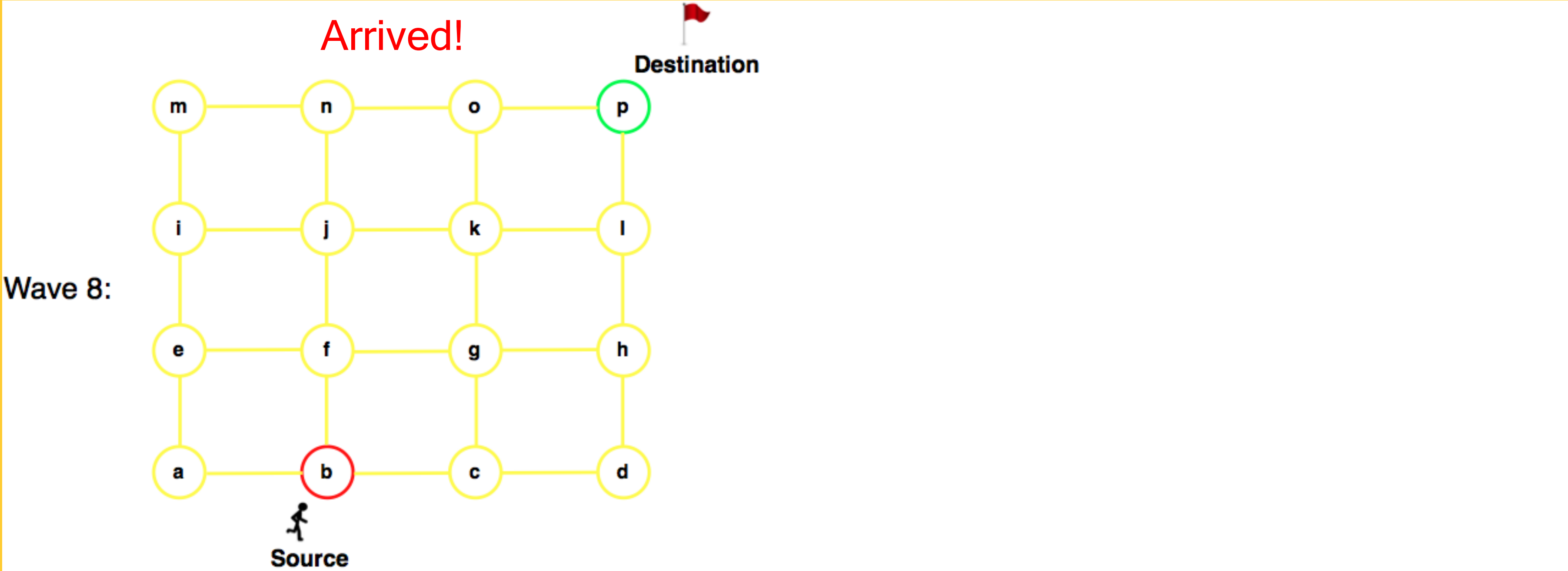
Dijkstra's Algorithm

Dijkstra's vs. A*



Dijkstra's Algorithm

Dijkstra’s vs. A*



Dijkstra’s Algorithm

Shortest Path Algorithms

- Iterate
  - Expand most promising node
    - Dijkstra’s: try closest descendent to self
    - A* : try closest descendent to both destination and self
  - Update current best path to each node, if a better path is found
- Till destination node is expanded

- Correct assuming
  - Sub-path optimality
  - Fixed, positive and additive edge costs
  - A* heuristic function h(x) (estimated distance to destination from x) has two properties
    - It is an underestimate to the actual distance
    - It is consistent i.e., for every edge (x,y), h(x) <= d(x,y) + h(y) (d is the length of edge (x,y)

Shortest Path Strategies for Sec Memory

- Dijkstra’s and Best first algorithms
  - Work well when entire graph is loaded in main memory
  - Otherwise their performance degrades substantially
- Hierarchical Routing Algorithms
  - Works with graphs on secondary storage
  - Loads small pieces of the graph in main memories
  - Can compute least cost routes

Shortest Path Strategies for Sec Memory

- Key ideas behind Hierarchical Routing Algorithm
  - **Fragment graphs** - pieces of original graph obtained via node partitioning
  - **Boundary nodes** - nodes of with edges to two fragments
  - **Boundary graph** - a summary of original graph
    - Contains Boundary nodes
    - Boundary edges: edges across fragments or paths within a fragment

Shortest Path Strategies for Sec Memory

- A Summary of Optimal path in original graph can be computed
  - Using Boundary graph and 2 fragments
- The summary can be expanded into optimal path in original graph
  - Examining a fragments overlapping with the path
  - Loading one fragment in memory at a time

Shortest Path Strategies - (Illustration of the Algorithm)

- Figure 6.7(a) - fragments of source and destination nodes
- Figure 6.7(b) - computing summary of optimal path using
  - Boundary graph and 2 fragments
  - Note use of boundary edges only in the path computation
- Figure 6.8(a) - The summary of optimal path using boundary edges
- Figure 6.8(b) Expansion back to optimal path in original graph

Hierarchical Routing Algorithm-Step 1

- Step 1: Choose Boundary Node Pair
  - Minimize COST(S,Ba)+COST(Ba,Bd)+COST(Bd,D)
  - Determining Cost May Be Non-Trivial

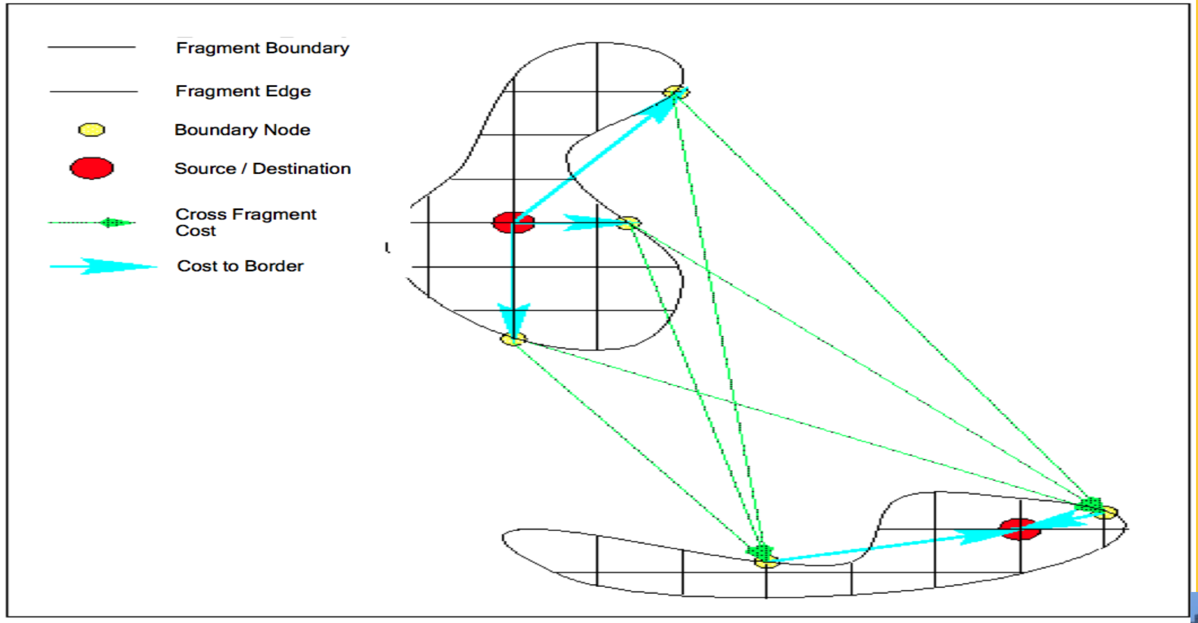


Fig 6.7(a)

Hierarchical Routing- Step 2

- Step 2: Shortest Boundary Path

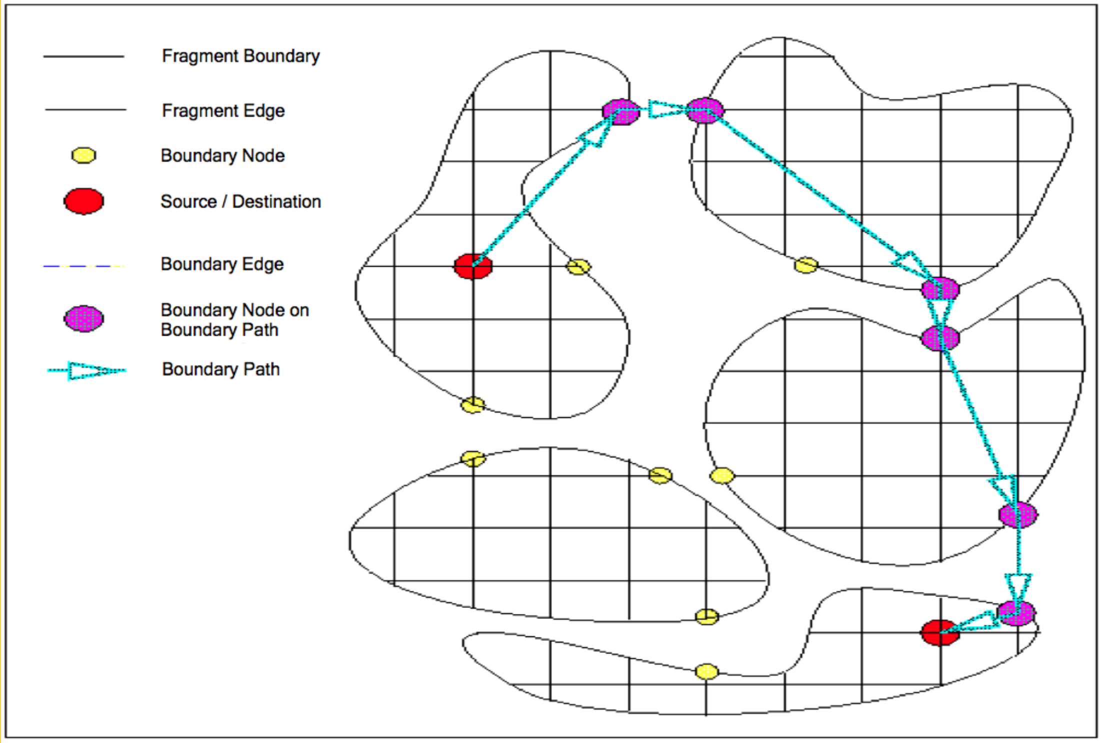


Fig 6.8(a)

Hierarchical Routing- Step 3

- Step 3: Expand Boundary Path: (B_{a1},B_d) -> B_{a1} Bd_{a2} B_{a3} B_{da4}...B_d
  - Boundary Edge (B_{ij},B_j) ->fragment path (B_{i1},$N_1N_2N_3$.......N_k,B_j)

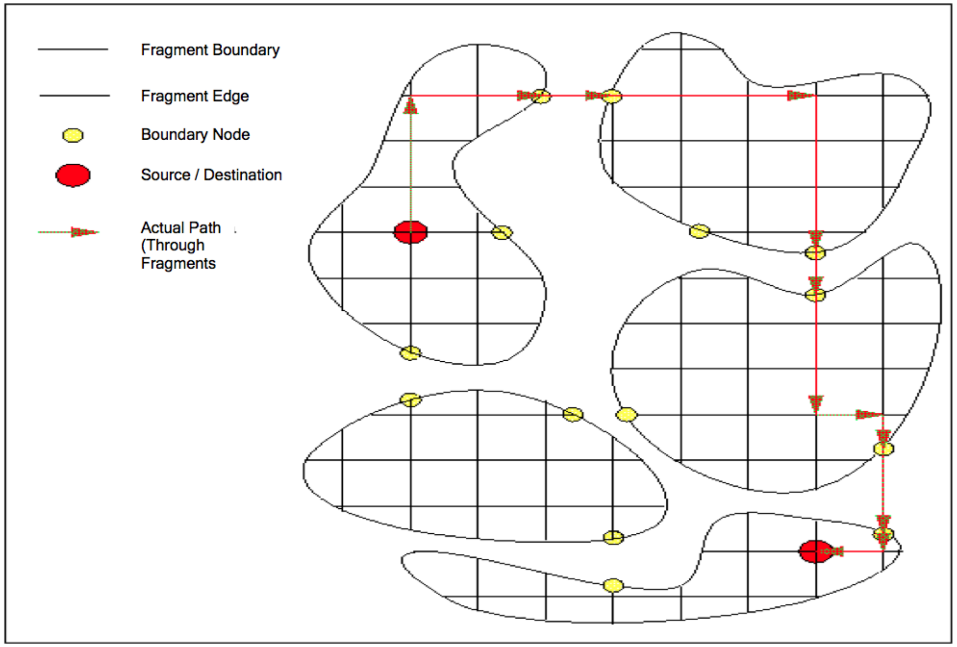


Fig 6.8(a)

Quiz

Which of the following is false?

a) Hierarchical routing algorithms are Disk-based shortest path algorithms

b) Breadth first search and depth first search are both connectivity query algorithms

c) Best first algorithm is always faster than Dijkstra’s algorithm

d) None of the above