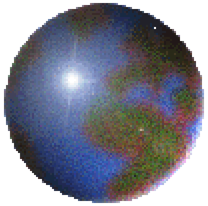
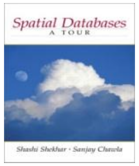


Chap 6: Spatial Networks

6.1 Example Network Databases
6.2 Conceptual, Logical and Physical Data Models
6.3 Query Language for Graphs
6.4 Graph Algorithms
6.5 Trends: Access Methods for Spatial Networks



Learning Objectives

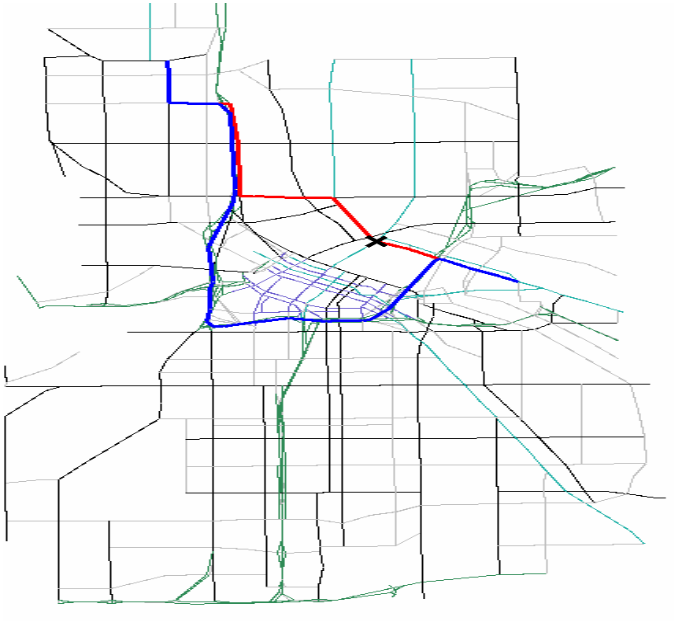
- Learning Objectives (LO)
  - LO1: Understand the concept of spatial network (SN)
    - What is a spatial network?
    - Why learn about spatial network?
  - LO2 : Learn about data models for SN
  - LO3: Learn about query languages and query processing
  - LO4: Learn about trends
- Focus on concepts not procedures!
- Mapping Sections to learning objectives
  - LO1 - 6.1
  - LO2 - 6.2
  - LO3 - 6.3, 6.4
  - LO4 - 6.5

Transistion

From **proximity** to **connectivity**

6.1 Example Spatial Networks

Road networks



1. Find shortest path from my current location to a destination.

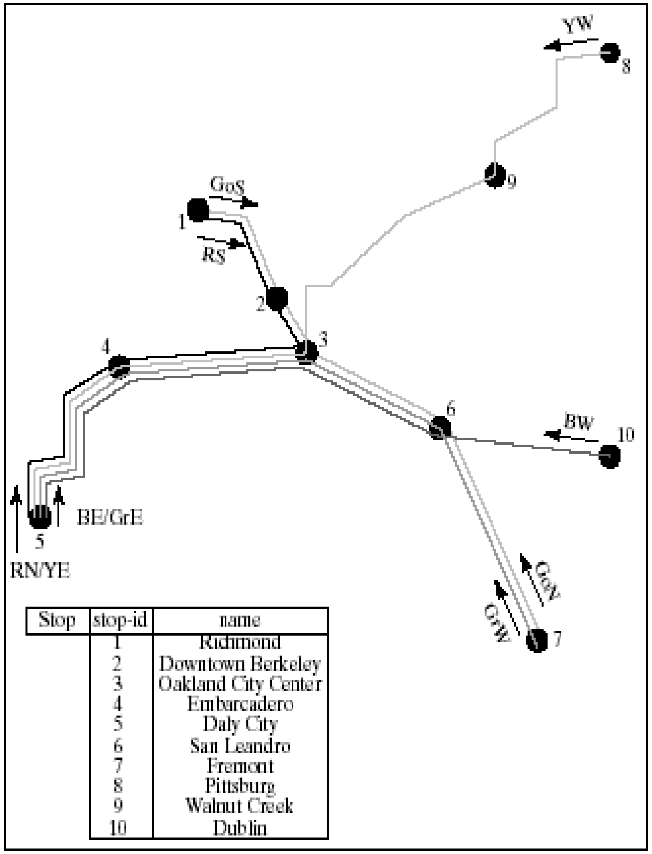
2. Find nearest hospital by distance along road networks.

3. Find shortest route to deliver goods to a list of retail stores.

4. Allocate customers to nearest service center using distance along roads

6.1 Example Spatial Networks

Railway networks



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.

3. List the route numbers that connect Downtown Berkeley and Daly City.

4. Find the last stop on the Blue West (BW) route.

6.1 Example Spatial Networks

River networks

1. List the names of all direct and indirect tributaries of the Mississippi river

2. List the direct tributaries of the Colorado.

3. Which rivers could be affected if there is a spill in river P1?

Learning Objectives

- Learning Objectives (LO)
  - LO1: Understand the concept of spatial network (SN)
  - LO2 : Learn about data models of SN
    - Representative data types and operations for SN
    - Representative data-structures
  - LO3: Learn about query languages and query processing
  - LO4: Learn about trends
- Focus on concepts not procedures!
- Mapping Sections to learning objectives
  - LO1 - 6.1
  - LO2 - 6.2
  - LO3 - 6.3, 6.4
  - LO4 - 6.5

6.2 Spatial Network Data Models

- Recall 3 level Database Design
  - Conceptual Data Model
    - Graphs
  - Logical Data Model -
    - Data types - Graph, Vertex, Edge, Path, …
    - Operations - Connected(), Shortest_Path(), ...
  - Physical Data Model
    - Record and file representation - adjacency list
    - File-structures and access methods - CCAM

6.2 Conceptual Data Models

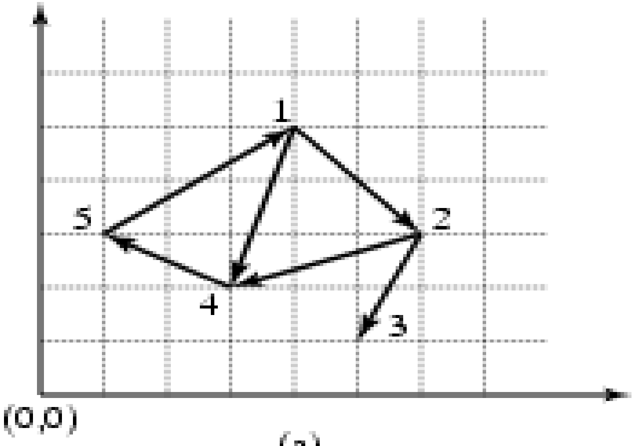
- Conceptual Data Model for Spatial Networks
  - A *graph*, G = (V,E)
  - V = a finite set of *vertices*
  - E = a set of edges E , between vertices in V
- Classifying graph models
  - Do nodes represent spatial points? - spatial vs. abstract graphs
  - Are vertex-pair in an edge order? - directed vs. undirected
- Examples
  - Road network is a spatial graph, River network is an abstract graph
  - River network is directed, Road network can be directed or undirected

6.2 Physical Data Models

- Categories of record/file representations
    - Main memory based
    - Disk based
- Main memory representations of graphs
    - Adjacency matrix M[A, B] = 1 if and only if edge(vertex A, vertex B) exists
    - Adjacency list : maps vertex A to a list of successors of A
    - Example: See Figure 6.2(a), (b) and (c) on next slide
- Disk based
    - normalized - tables, one for vertices, other for edges
    - denormalized - one table for nodes with adjacency lists
    - Example: See Figure 6.2(a), (d) and (e) on next slide

6.2.2 Physical Data Models - Figure 6.2

Fig 6.2

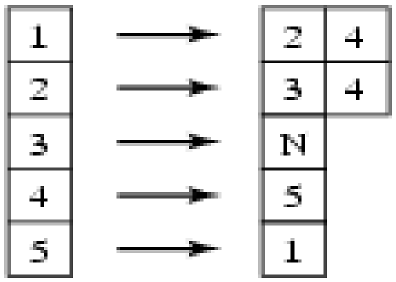


(a)

Destination

	1	2	3	4	5
source 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



(c) Adjacency-List

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)

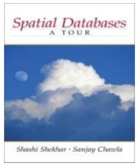
(e) Denormalized Node Table

Learning Objectives

- Learning Objectives (LO)
    - LO1: Understand the concept of spatial network (SN)
    - LO2 : Learn data models for SN
    - LO3: Learn about query languages and query processing
        - Query building blocks
        - Processing strategies
    - LO4: Learn about trends
- Focus on concepts not procedures!
- Mapping Sections to learning objectives
    - LO1 - 6.1
    - LO2 - 6.2
    - LO3 - 6.3, 6.4
    - LO4 - 6.5

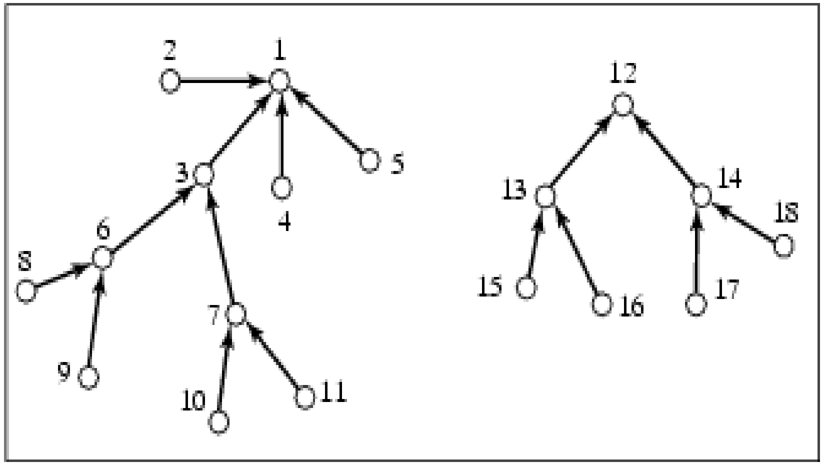
6.3 Query Languages For Graphs

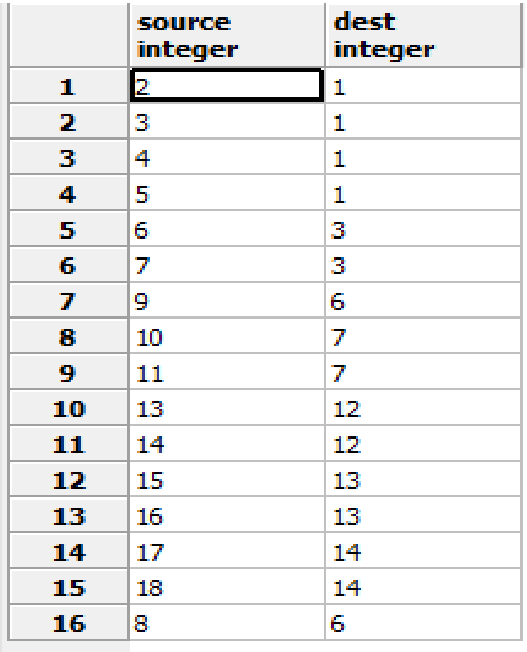
- Recall Relation algebra (RA) based languages
  - Can not compute paths with arbitrary length
  - SQL support for graph queries
  - SQL2 - **CONNECT** clause in SELECT statement
    - For directed acyclic graphs, e.g. hierarchies
  - SQL 3 - **WITH RECURSIVE** statement
    - Transitive closure on general graphs
  - SQL 3 -user defined data types
    - Can include shortest path operation!



6.3.2 SQL2 Connect Clause

- Syntax details
  - FROM clause a table for directed edges of an acyclic graph
  - PRIOR identifies direction of traversal for the edge
  - START WITH specifies first vertex for path computations
- Semantics
  - List all nodes reachable from first vertex using directed edge in specified table
  - Assumption - no cycle in the graph!
  - Not suitable for train networks, road networks





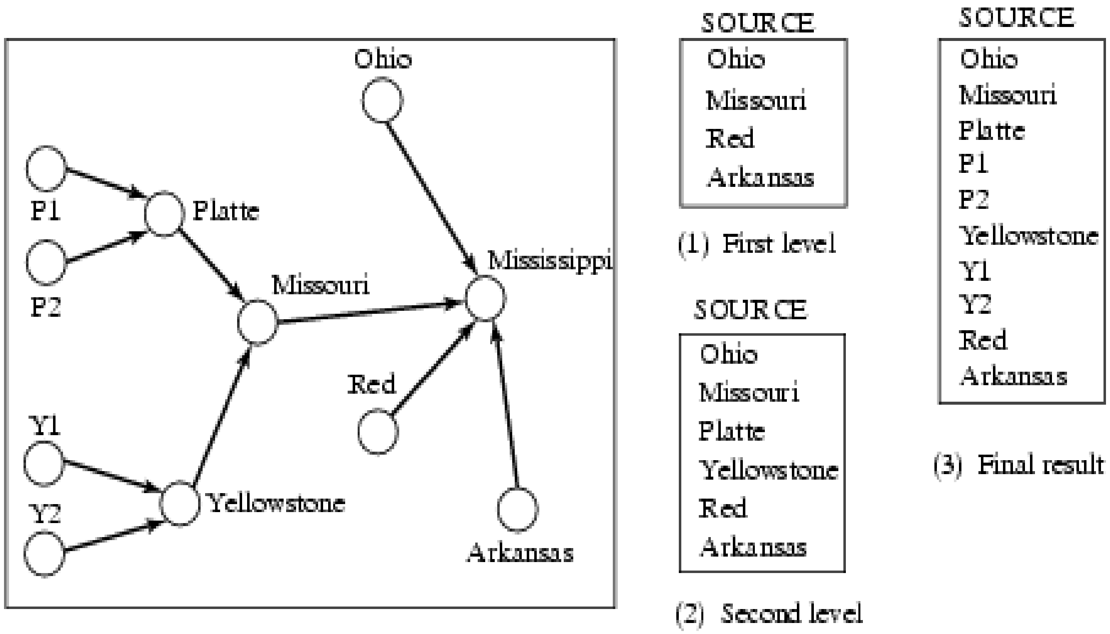
	source integer	dest integer
1	2	1
2	3	1
3	4	1
4	5	1
5	6	3
6	7	3
7	9	6
8	10	7
9	11	7
10	13	12
11	14	12
12	15	13
13	16	13
14	17	14
15	18	14
16	8	6

Table FallsInto

SQL Connect Clause - Example

- SQL experssion on right
- Execution trace of paths
  - starts at vertex 1 (Mississippi)
  - adds children of 1
  - adds children of Missouri
  - adds children of Platte
  - adds children of Yellostone
- Result has edges
  - from descendents
  - to Mississippi

```
SELECT source
FROM FallsInto
CONNECT BY PRIOR source = dest
START WITH dest =1
```



(a) Missisippi network (Y1 = Bighorn river, Y2 = Powder river, P1 = Sweet water river, P2 = Big Thompson river).

(b) The sequence of the CONNECT clause

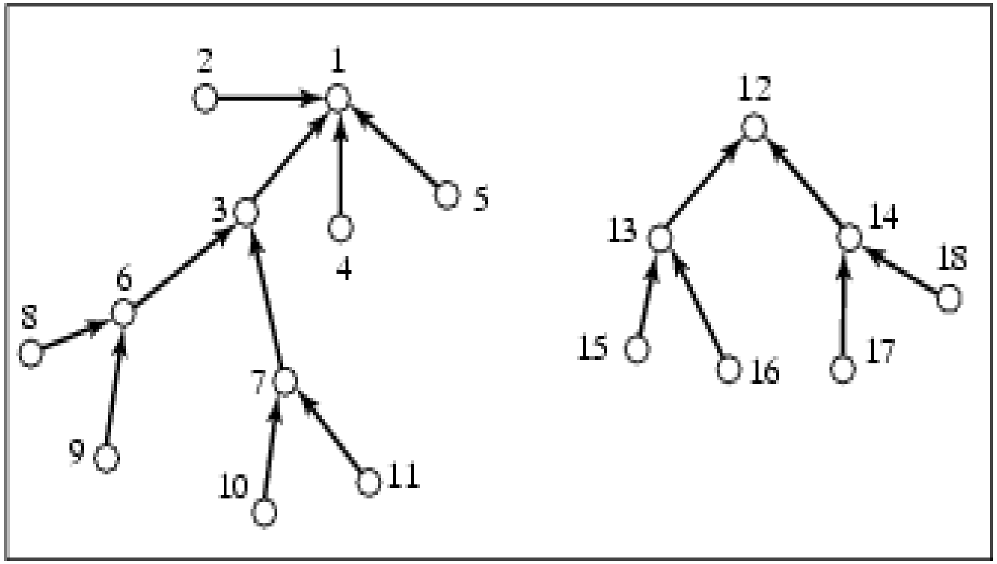
SQL Connect Clause - Exercise

- Study 2 SQL queries on right
    - Note different use of PRIOR keyword

```
SELECT source FROM FallsInto
CONNECT BY PRIOR source = dest
START WITH dest =3
```

- Compute results of each query
- Which one returns ancestors of 3?
- Which returns descendents of 3?
- Which query lists river affected by
    - oil spill in Missouri (id = 3)?

```
SELECT source FROM FallsInto
CONNECT BY  source = PRIOR  dest
START WITH dest =3
```



SQL Connect Clause - Exercise

- Stop to a specified level

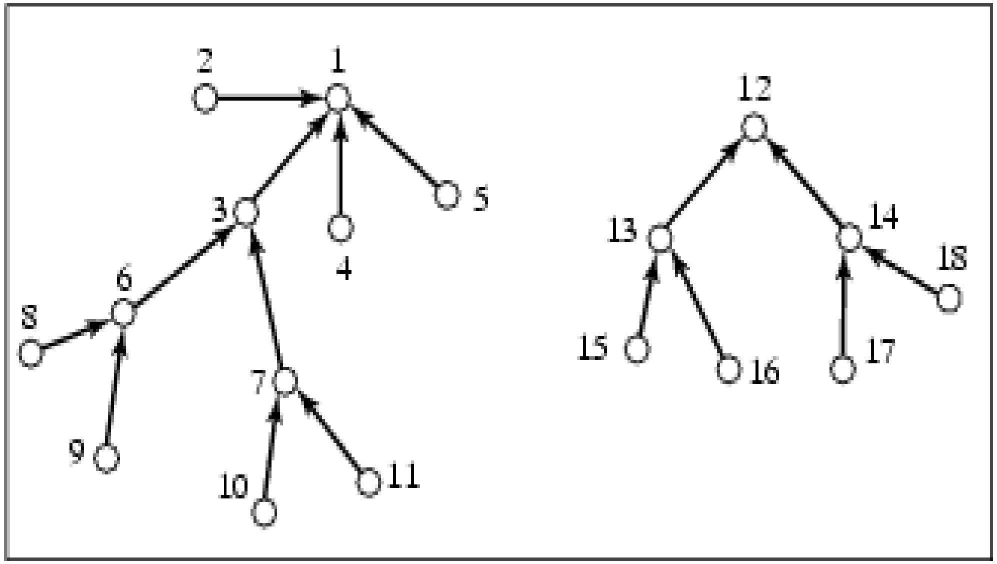
•Visit 2, 3, 4, 5, 6, 7

SELECT source **FROM** FallsInto

CONNECT BY PRIOR source = dest

WHERE Level <= 2

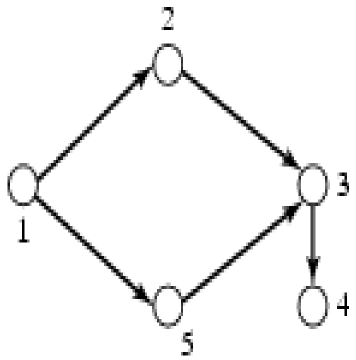
START WITH dest =1



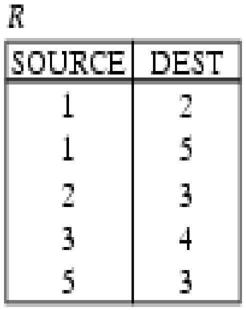
Concept of Transitive Closure

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

- 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.



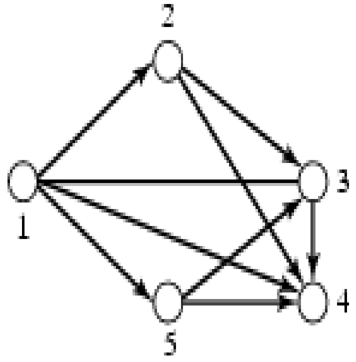
(a) Graph G



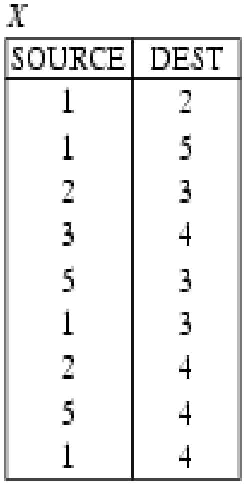
R

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 relational form

6.3.3 SQL3 Recursion

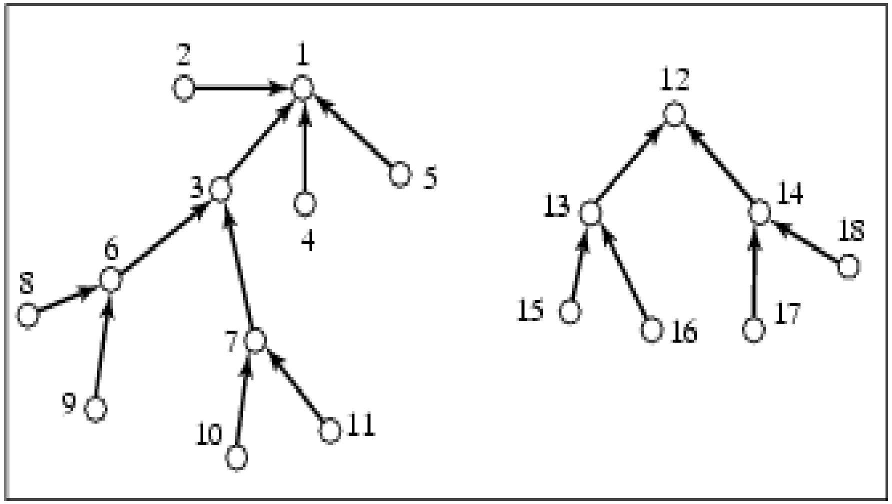
- Computing table X from table R (figure in previous slide)

```
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);
```

- Meaning
  - Initialize X by copying directed edges in relation R
  - Infer new edge(a,c) if edges (a,b) and (b,c) are in X
  - Declarative query does not specify algorithm needed to implement it
- The graph can contain cycles!

Recursion Example

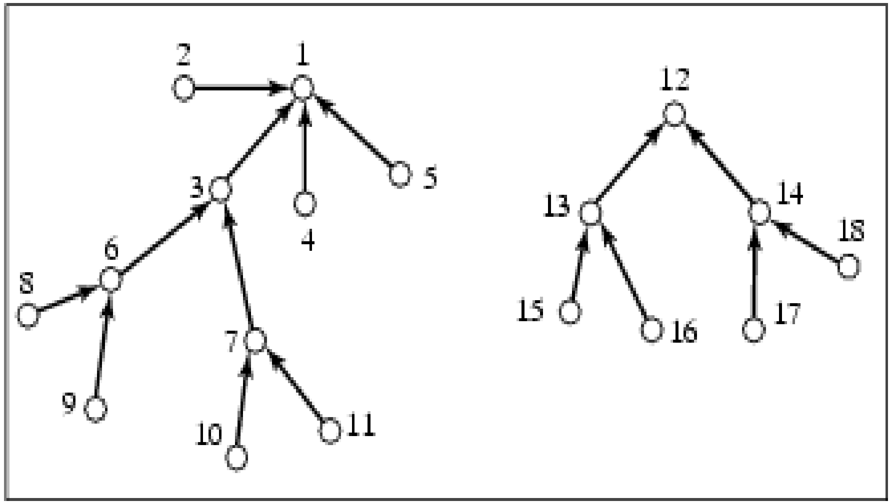
```
With Recursive C(source, dest) as
(Select source, dest From FallsInto
 Union
 Select FallsInto.source, C.dest
 From FallsInto, C
 Where FallsInto.dest = C.source)
 Select * From C
 Where dest = 1;
```



- Visit 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Recursion Example

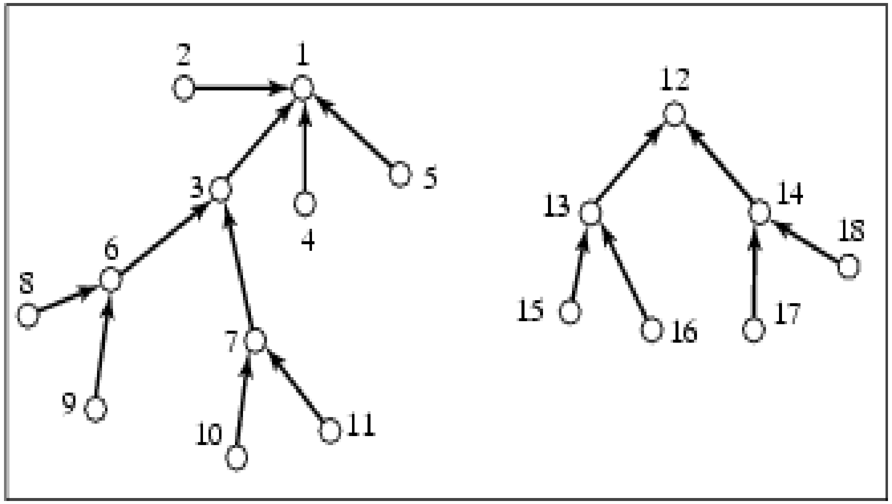
```
With Recursive C(source, dest) as
(Select source, dest From FallsInto
 Union
 Select FallsInto.source, C.dest
 From FallsInto, C
 Where FallsInto.dest = C.source)
 Select * From C
 Where source = 3;
```



- Visit 6, 3, 1

Recursion Example

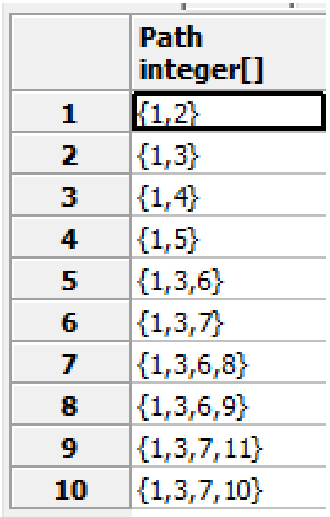
```
With Recursive C(source, dest, depth) as
(Select source, dest, 1 From FallsInto
 Union
 Select FallsInto.source, C.dest, C.depth+1
 From FallsInto, C
 Where FallsInto.dest = C.source)
 Select * From C
 Where dest = 1 And depth <= 2;
```



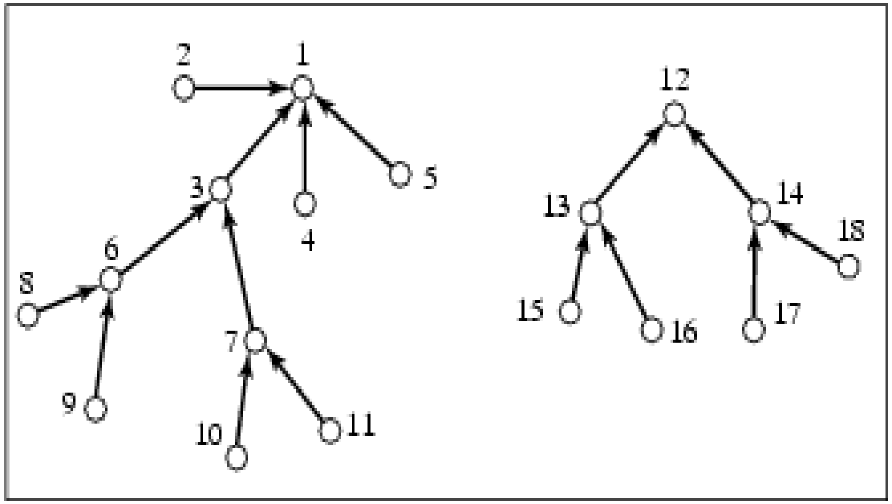
- Visit 2, 3, 4, 5, 6, 7

Recursion Example

```
With Recursive C(source, dest, path) as
(Select source, dest, ARRAY[FallsInto.dest] From FallsInto
 Union All
 Select FallsInto.source, C.dest, path || FallsInto.dest
 From FallsInto, C
 Where FallsInto.dest = C.source)
 Select path || source as "Path"
 From C
 Where dest = 1;
```

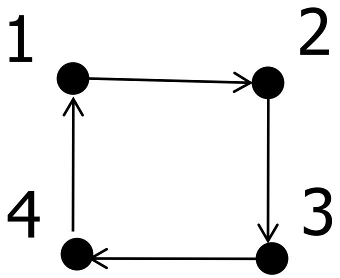


	Path integer[]
1	{1,2}
2	{1,3}
3	{1,4}
4	{1,5}
5	{1,3,6}
6	{1,3,7}
7	{1,3,6,8}
8	{1,3,6,9}
9	{1,3,7,11}
10	{1,3,7,10}



Cycles

```
With Recursive C(source, dest) as
(Select source, dest From g
 Union
 Select g.source, C.dest
 From g, C
 Where g.dest = C.source)
 Select * From C
 Where source = 1;
```



- Visit 2, 3, 4, 1

But be careful with Union All