UE19CS251
Design and Analysis of Algorithms
Unit 5: Limitations of Algorithmic Power and Coping with the Limitations

Transitive Closure (Warshall's Algorithm)

PES University

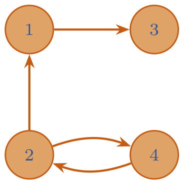
Outline

Concepts covered

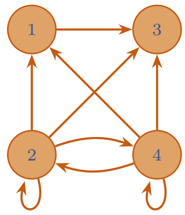
- Transitive Closure (Warshall's Algorithm)
  - Motivation
  - Algorithm
  - Example

1 Transitive Closure

- Computes the transitive closure of a relation
- Alternatively: existence of all nontrivial paths in a digraph (directed graph)
- Example of transitive closure:



$$\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$



$$\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & \mathbf{1} & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & \mathbf{1} & \mathbf{1} \end{bmatrix} \end{array}$$

2 Warshall's Algorithm

- Constructs transitive closure T as the last matrix in the sequence of $n \times n$ matrices $R^{(0)}, \ldots, R^{(k)}, \ldots, R^{(n)}$ where $R^{(k)}[i,j] = 1$ iff there is nontrivial path from i to j with only first k vertices allowed as intermediate vertices
  - $R^{(0)} = A$ (adjacency matrix), $R^{(n)} = T$ (transitive closure)
- On the k^{th} iteration, the algorithm computes $R^{(k)}$

$$R^{(k)}[i,j] = \begin{cases} 1 & \text{if path from } i \text{ to } k \text{ and } k \text{ to } j, \text{i.e., } R^{(k-1)}[i,k] = R^{(k-1)}[k,j] = 1 \\ R^{(k-1)}[i,j] & \text{otherwise} \end{cases}$$

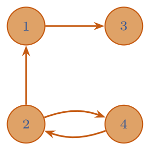
$$R^{(k)}[i,j] = R^{(k-1)}[i,j] \ \textbf{ or } \ (R^{(k-1)}[i,k] \textbf{ and } R^{(k-1)}[k,j])$$

3 Algorithm

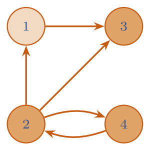
Transitive Closure (Warshall's Algorithm)

```
procedure WARSHALL(()A[1...n, 1...n])
    ▷ Input: The adjacency matrix A of a digraph with n vertices
    ▷ Output: The transitive closure of the digraph
    R^(0) ← A
    for k ← 1 to n do
        for i ← 1 to n do
            for j ← 1 to n do
                R^(k)[i, j] ← R^(k−1)[i, j] or (R^(k−1)[i, k] and R^(k−1)[k, j]);
    return R^(n)
```

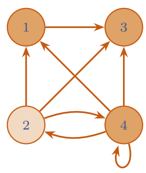
4 Warshall's Algorithm



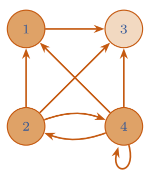
$$\begin{array}{c@{\;}c}
 & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\end{array}$$



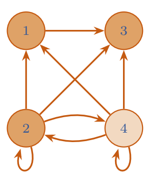
$$\begin{array}{c@{\;}c}
 & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\end{array}$$



$$\begin{array}{c@{\;}c}
 & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & \mathbf{1} & \mathbf{1} \end{bmatrix}
\end{array}$$



$$\begin{array}{c@{\;}c}
 & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}
\end{array}$$



$$
\begin{array}{c c}
 & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}
\end{array}
$$

5 Think About It

- Is Warshall's algorithm efficient for sparse graphs? Why / why not?
- Can Warshall's algorithm be used to determine if a graph is a DAG (Directed Acyclic Graph)? If so, how?