# Warshall's and Floyd's Algorithms

---

## Warshall's Algorithm

Computes the *transitive closure* of a digraph.

Can find the transitive closure by building a spanning tree from each vertex (with something like BFT or DFT). If it costs up to $\Theta(|V|^2)$ to build each spanning tree we end up with a cost of

Warshall's approach:

**ALGORITHM** WARSHALL($A$)
  //Input: $A[1..n][1..n]$ an adjacency matrix representation of a graph
  // where $A[i][j]$ is **true** iff a edge exists from $i$ to $j$
  // Each $R^{(i)}[1..n][1..n]$ is an iteration toward the closure
  $R^{(0)} \leftarrow A$
  **for** $k \leftarrow 1..n$ **do**
    **for** $i \leftarrow 1..n$ **do**
      **for** $j \leftarrow 1..n$ **do**
        $R^{(k)}[i][j] \leftarrow R^{(k-1)}[i][j]$ **or** $(R^{(k-1)}[i][k]$ **and** $R^{(k-1)}[k][j])$
  **return** $R^{(n)}$

Efficiency:

Example of Warshall's Algorithm:

# Floyd's Algorithm

A pretty straightforward extension of Warshall's Algorithm can be used to solve the *all sources shortest path* problem.

**ALGORITHM** FLOYD($W$)
//Input: $W[1..n][1..n]$ an adjacency matrix representation of graph edge weights
// nonexistent edges have a weight of $\infty$
$D \leftarrow W$ // a matrix copy
**for** $k \leftarrow 1..n$ **do**
  **for** $i \leftarrow 1..n$ **do**
    **for** $j \leftarrow 1..n$ **do**
      $D[i][j] \leftarrow min(D[i][j], D[i][k] + D[k][j])$
**return** $D$

Exercise 8.4.7

$$\begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$