Transitive Closure

Given a digraph $G$, the transitive closure of $G$ is the digraph $G^*$ such that
- $G^*$ has the same vertices as $G$
- If $G$ has a directed path from $u$ to $v$ ($u \neq v$), $G^*$ has a directed edge from $u$ to $v$

The transitive closure provides reachability information about a digraph.

Computing the Transitive Closure

We can perform DFS starting at each vertex
- DFS($G, v$) finds nodes reachable from $v$
- for each reachable node $w$, add edge $(v, w)$ to $G^*$
- $O(n(n+m))$

Alternatively ... Use dynamic programming: The Floyd-Warshall Algorithm

Floyd-Warshall Transitive Closure

Idea #1: Number the vertices 1, 2, ..., $n$. Call them $v_1, v_2, ..., v_n$.

Idea #2: Consider paths that use only vertices $v_1, v_2, ..., v_k$ as intermediate vertices

On path $P_2$, intermediate vertices are $X, W, Y$.

Subproblem definition: $G_k$ is a graph where
- directed edge $(v_i, v_j)$ if $G$ has a directed path from $v_i$ to $v_j$ with intermediate vertices in the set $\{v_1, v_2, ..., v_k\}$.
**Floyd-Warshall Transitive Closure**

- Constructing $G_k$ from $G_{k-1}$:
  - For each pair of vertices $(v_i, v_j)$ in $G_{k-1}$:
    - If $(v_i, v_j)$ is in $G_{k-1}$, then it is also in $G_k$.
    - If $(v_i, v_k)$ and $(v_k, v_j)$ are in $G_{k-1}$, then $(v_i, v_j)$ is in $G_k$.

  Uses only vertices numbered 1,...,k.

- (add this edge if it's not already in)

**Floyd-Warshall’s Algorithm**

- Floyd-Warshall’s algorithm numbers the vertices of $G$ as $v_1, ..., v_n$ and computes a series of digraphs $G_0, ..., G_n$.
  - $G_0 = G$.
  - In phase $k$, digraph $G_k$ is computed from $G_{k-1}$.

We have that $G_n = G*$.

Running time: $O(n^3)$, assuming areAdjacent is $O(1)$ (e.g., adjacency matrix).

**Algorithm**

```
Algorithm FloydWarshall(G)
Input digraph G
Output transitive closure G* of G
i ← 0
for all v ∈ G.vertices()
disjoint v as v_i
i ← i + 1
G_0 ← G
for k ← 1 to n do
  G_k ← G_{k-1}
  for i ← 1 to n (i ≠ k) do
    for j ← 1 to n (j ≠ i, k) do
      if G_{k-1}.areAdjacent(v_i, v_k) ∧ G_{k-1}.areAdjacent(v_k, v_j)
        G_k.insertDirectedEdge(v_i, v_j, k)
return G_n
```

**Floyd-Warshall Example**

- Diagram showing vertices and edges in a graph with vertices SFO, LAX, JFK, BOS, ORD, MIA, LAX, SFO, ORD.
Floyd-Warshall, Iteration 1

Floyd-Warshall, Iteration 2

Floyd-Warshall, Iteration 3
Floyd-Warshall, Conclusion

[Diagram of a network with labeled nodes and arrows indicating connections.]