Warshall's Algorithm

A more efficient way of computing the transitive closure of a relation with digraph on vertices $\{v_1,v_2,\ldots,v_n\}$:

Theorem (p. 606). Let $W_k=(w_{ij}^{[k]})$ be the 0,1 matrix $w_{ij}^{[k]} = 1$ iff there is a path from v_i to v_j with any interior vertices in the set $\{v_1,v_2,\ldots v_k\}$. Then

$w_{ij}^{[k]} = w_{ij}^{[k-1]} \vee (w_{ik}^{[k-1]} \wedge w_{kj}^{[k-1]})$

$W_0 = W_R$, $W_n = W_{R^*}$.

Proof: We'll use induction.

Base case: k=0. $W_0 = W_R$ because there can be no interior vertices, so just a single edge.

Induction step: If true for k-1, show $w_{ij}^{[k]} = w_{ij}^{[k-1]} \vee (w_{ik}^{[k-1]} \wedge w_{kj}^{[k-1]})$

because there is a path from v_i to v_j using interior vertices from $\{v_1, v_2,\ldots,v_k\}$ iff

- There is a path without v_k as an interior vertex (so $w_{ij}^{[k-1]} = 1$) or
- There is path with v_k as an interior vertex, in which case both $w_{ik}^{[k-1]}$ and $w_{kj}^{[k-1]}$ are 1. (there must be a k-1path from v_i to v_k and from v_k to v_j)

Using Warshall's Algorithm

As shown in the book, the formula giving Warshall's Algorithm easily translates to computer code.

If you do it by hand, just note that in $w_{ij}^{[k]} = w_{ij}^{[k-1]} \vee (w_{ik}^{[k-1]} \wedge w_{kj}^{[k-1]})$ you go from W_{k-1} to W_k by looking at the matrix for W_{k-1}. If you can go from v_i to v_k in W_{k-1} then in W_k you can add an entry ij if v_k goes to v_j in W_{k-1}. (this is easier than it sounds)

Transitive Closure via Warshall's Algorithm

$$W_0 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad W_1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad W_3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_{R^*} = W_3 = W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Arithmetic in Z_n

- If a ≡ b (mod n) and c ≡ d (mod n) then a*c ≡ b*d (mod n).
- This shows we can define [u] * [v] = [u*v] and we'll get the same answer no matter what representatives u and v we choose for the equivalence classes.

Multiplication in Z_7

*	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1