Source Shortest paths: General weights - BellMan-Ford algo, 0/1 knapsack, The travelling salesperson Problem.

The General Method:

→ The word "programming" in the name of this technique stands for "planning".

→ Dynamic programming is an algorithm design method that can be used when the solution to a problem can be viewed as the results of a sequence of decisions.

Examples: knapsack problem, Optimal merge patterns, Shortest path, Principle of Optimality.

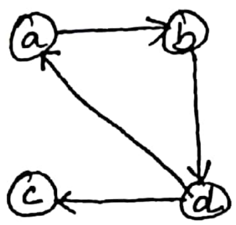
Warshall's algorithm

→ Warshall's algorithm constructs the transitive closure of a given digraph with n vertices through a series of n-by-n boolean Matrices:

$$R^0, R^1, \ldots R^{(k-1)}, R^{(k)}, \ldots R^{(n)}$$

→ Transitive Closure: The transitive closure of a directed graph with n vertices can be defined as the n-by-n boolean Matrix $T = \{t_{ij}\}$, in which the element in the i^{th} row $(1 \le i \le n)$ & the j^{th} column $(1 \le j \le n)$ is 1 if there exits a nontrivial directed path from the i^{th} vertex to the j^{th} vertex; otherwise t_{ij} is 0.

Example 1



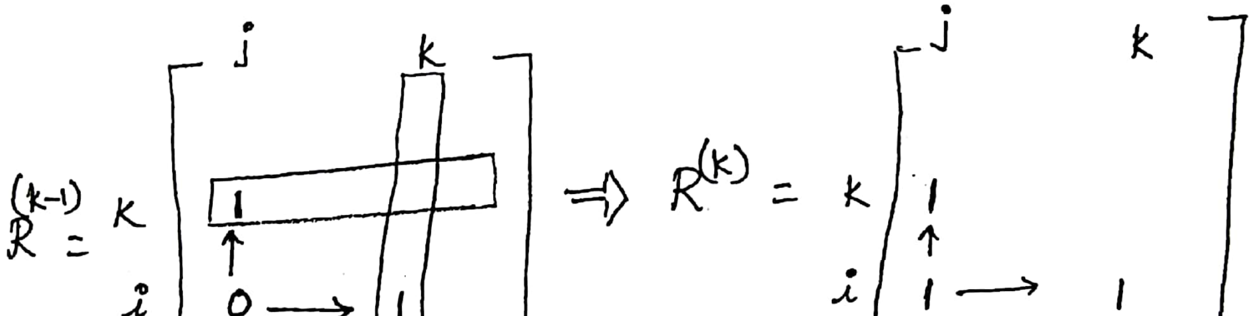
Digraph

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array} \qquad T = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Adjacency Matrix

Transitive closure

→ The Rule for changing zeros in warshall's algorithm.



→ The following formula ~~is~~ used for generating the elements of matrix $R^{(k)}$ from the elements of matrix $R^{(k-1)}$

$$r_{ij}^{(k)} = r_{ij}^{(k-1)} \text{ or } r_{ik}^{(k-1)} \text{ and } r_{kj}^{(k-1)} \quad \text{—(1)}$$

$$\therefore\ R^{0} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 0 & 1 & 0 \end{array} \qquad R^{1} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 0 \end{array}$$

$$R^{2} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array} \qquad R^{3} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array} \qquad R^{4} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 1 & 1 & 1 & 1 \\ b & 1 & 1 & 1 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

Algorithm Warshalls (A[1..n, 1..n])
// Implements Warshall's algo for computing the Transitive closure
// Input: The adjacency matrix A of a digraph with n vertices
// Output: The transitive closure of the digraph

$R^{(0)} \leftarrow A$
for $k \leftarrow 1$ to n do
  for $i \leftarrow 1$ to n do
    for $j \leftarrow 1$ to 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.

$$f(n) = \sum_{k=0}^{n-1} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1}$$

$$= n^2 \sum_{k=0}^{n-1} 1 = n^2(n-1-0+1) = n^2 * n = n^3$$

$\therefore$ it is $\theta(n^3)$

3

Example 2 Apply warshall algorithm to find the transitive closureness of a digraph defined by the adjacency matrix.

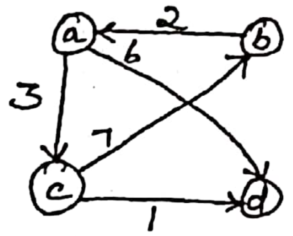
$$R^0 = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 1 & 0 \\ c & 0 & 0 & 0 & 1 \\ d & 0 & 0 & 0 & 0 \end{array} \qquad R^1 = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 1 & 0 \\ c & 0 & 0 & 0 & 1 \\ d & 0 & 0 & 0 & 0 \end{array} \qquad R^2 = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 1 & 0 \\ b & 0 & 0 & 1 & 0 \\ c & 0 & 0 & 0 & 1 \\ d & 0 & 0 & 0 & 0 \end{array}$$

$$R^3 = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 1 & 1 \\ b & 0 & 0 & 1 & 1 \\ c & 0 & 0 & 0 & 1 \\ d & 0 & 0 & 0 & 0 \end{array} \qquad R^4 = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 1 & 1 \\ b & 0 & 0 & 1 & 1 \\ c & 0 & 0 & 0 & 1 \\ d & 0 & 0 & 0 & 0 \end{array}$$

Floyd's Algorithm.

- This algorithm is used to calculate the all pair shortest path for a weighted graph
- offcourse the graph may be either directed or undirected.

Example 1



$$d[i,j] = \min\left[d[i,j],\ d[i,k] + d[k,j]\right]$$

```
void warshall (int d[10][10], int n)
{
    int i, j, k;
    for (k=0 ; k<n; k++)
        for(i=0; i<n; i++)
            for (j=0; j<n; j++)
                d[i][j] = max(d[i][j],
            d[i][k]&& d[k][j]);
}
```