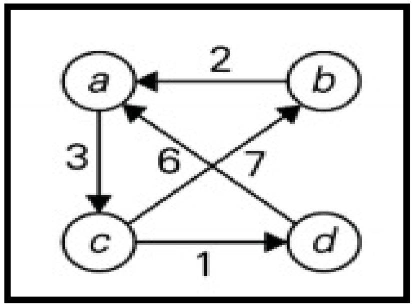
All pair shortest path & Single Source Shortest Path

Solutions for unsolved problems will be checked on 9th Feb 2015

All pair shortest path algorithm: (Flyodd-Warshall)

- Principle: To find shortest path from ONE source to ALL possible destinations in the GRAPH.
- The path can be DIRECT PATH or INDIRECT PATH
- Method:
- Initially the direct path matrix is generated.
- First vertex is used as intermediate and paths are generated, if existing path is greater then it is replaced by new path.
- The process is continued for all the vertices as intermediate vertices present in the graph.

Example of APSP



$$W = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \infty & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \infty & 0 \end{array}$$

$$D^{(1)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \mathbf{5} & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \mathbf{9} & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 1, i.e. just *a* (note two new shortest paths from *b* to *c* and from *d* to *c*).

$$D^{(2)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & 5 & \infty \\ c & \mathbf{9} & 7 & 0 & 1 \\ d & 6 & \infty & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 2, i.e. a and *b* (note a new shortest path from *c* to *a*).

Example: Cont..

$$D^{(3)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & \mathbf{10} & 3 & \mathbf{4} \\ 2 & 0 & 5 & \mathbf{6} \\ 9 & 7 & 0 & 1 \\ 6 & \mathbf{16} & 9 & 0 \end{bmatrix} \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 3, i.e. a, *b*, and *c* (note four new shortest paths from *a* to *b*, from a to *d*, from *b* to *d*, and from *d* to *b*).

$$D^{(4)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ \mathbf{7} & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 4, i.e. a, *b*, *c*, and *d* (note a new shortest path from *c* to *a*).

Algorithm: APSP

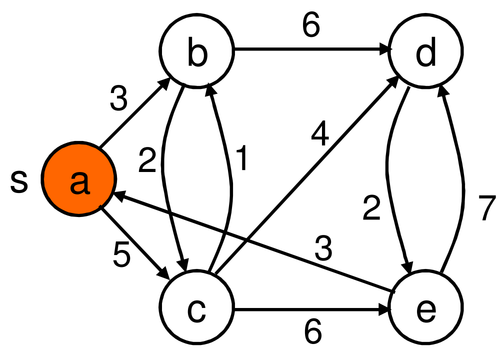
<u>Algorithm: All pair shortest path Algorithm</u>

Assumptions

1) The graph is represented using cost matrix of size nxn
2) The algorithm will generate output matrix in the form of matrix A of size nxn

```
Algorithm allpaths(G,cost,n: A)
{
  Step-1: Direct Path Matrix
         for i  = 1 to n do
           for j = 1 to n do
             A[i, j] = cost[i, j];
 Step – 2: Using intermediate vertices one by one
         for k = 1 to n do
           for i  = 1 to n do
             for j = 1 to n do
             A[i, j] = min(A[i,j], A[i,k] + A[k,j])
}//end of algorithm
```

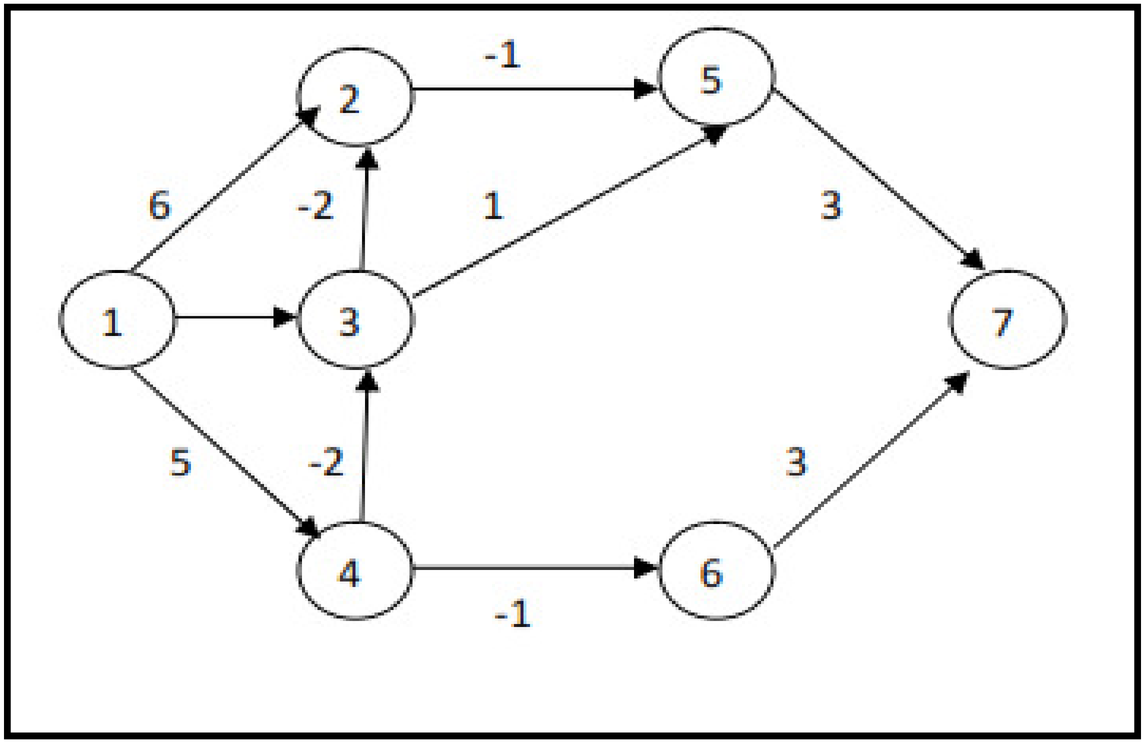
Example 2: All pair shortest path



Single Source Shortest Path: Bellman Ford

- **Algorithm permits NEGATIVE edges in the graph.**
- **Generally Graph has no cycles**
- **Objective: To find shortest path from one source to all possible destination.**
- **Path can be either DIRECT or INDIRECT**
- **For a graph of “n” vertices: the length of shortest path can be in the range of “1 to n-1”.**
- **The algorithm generates distance matrix by increasing the length value by 1, in each iteration.**
- **The results of previous iteration are used in computation for next iteration.**

Example: Graph



Formulation for Shortest Path

- $dist^{k}[\mathrm{u}] = \min\left\{dist^{k-1}[u], min\{dist^{k-1}[i] + cost[i,u]\}\right\}$
- “u” is destination vertex
- “i“ represents all possible intermediate vertices except “u”
- The vertex “i“ should have connection to “u”

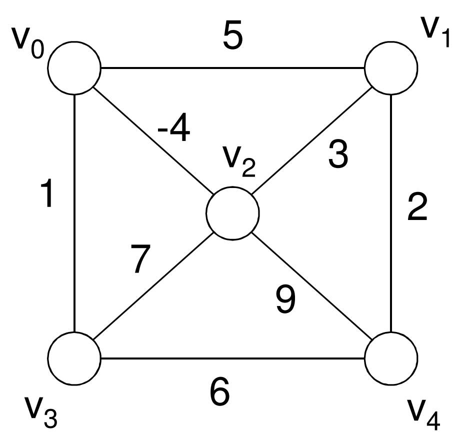
Execution

K	1	2	3	4	5	6	7
1	0	6	5	5	∞	∞	∞
2	0	3	3	5	5	4	∞
3	0	1	3	5	2	4	7
4	0	1	3	5	0	4	5
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

Algorithm: Same as Greedy

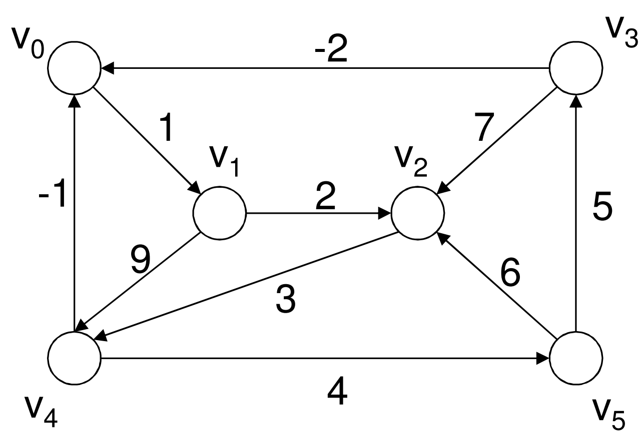
```
Algorithm:
Algorithm Bellman Ford Algo(v, cost, dist, n)
{
 Step 1
          For i= 1 to n do
                    Dist[i] = cost[v,i]
                    Parent[i]=v
          If (i==v) then
                    Parent[i]= v
          Else
                    If cost[v,i] != infinity
                      Parent[i]=v

Step 2
          For k = 2 to n-1 do
               For (each vertex "u" such that u≠v and "u" has at least one incoming edge) do
           // Let "i" represent the incoming edge
                    For (each <i,u> in the graph) do
                              If (dist[u] > dist[i] + cost[i,u]) then
                                        dist[u] = dist[i] + cost[i,u]
                                        parent[u] = i
} // End of Algorithm
```

Single source shortest path
Example 2: Weighted Graph and Weight Matrix



$$\begin{array}{c|c} & \begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 0 & 5 & -4 & 1 & 0 \\ 5 & 0 & 3 & 0 & 2 \\ -4 & 3 & 0 & 7 & 9 \\ 1 & 0 & 7 & 0 & 6 \\ 0 & 2 & 9 & 6 & 0 \end{pmatrix} \end{array}$$

Single source shortest path
Example 3: Directed Weighted Graph and Weight Matrix



$$
\begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{pmatrix}
0 & 1 & \infty & \infty & \infty & \infty \\
\infty & 0 & 2 & \infty & 9 & \infty \\
\infty & \infty & 0 & \infty & 3 & \infty \\
-2 & \infty & 7 & 0 & \infty & \infty \\
-1 & \infty & \infty & \infty & 0 & 4 \\
\infty & \infty & 6 & 5 & \infty & 0
\end{pmatrix}
\end{array}
$$