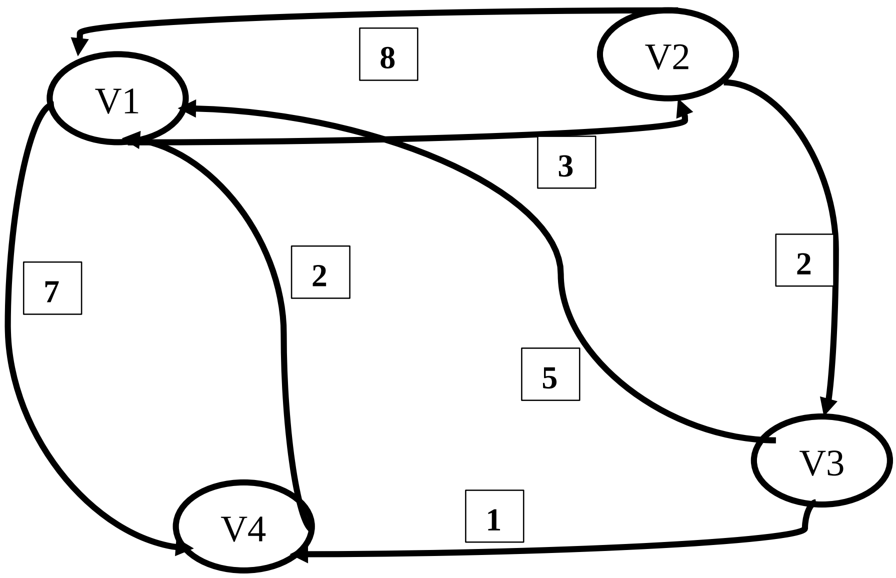
Flod-Warshall Algorithm

Swarup Kr Ghosh

Flod-Warshall Algorithm: Multiple source and multiple destination shortest path.

- It follows both DP and greedy approaches.
- Applicable for both positive and negative edges.



V1-----V2
V1------ V3
V1------V4
V2---V1
V2---V3
V2---V4

A0=	V1	V2	V3	V4
V1	0	3	INF	7
V2	8	0	2	INF
V3	5	INF	0	1
V4	2	INF	INF	0

A[V4,V1]=

min(A[V4,V1], A[V4,V2]+A[V2,V1])

Min (INF, 2+3)

TRIANGULAR INEQUALITY

A1=	V1	V2	V3	V4
V1	0	3	INF	7
V2	8	0	2	15
V3	5	8	0	1
V4	2	5	INF	0

A2=	V1	V2	V3	V4
V1	0	3	5	7
V2	8	0	2	15
V3	5	8	0	1
V4	2	5	7	0

A3=	V1	V2	V3	V4
V1	0	3	5	6
V2	7	0	2	3
V3	5	8	0	1
V4	2	5	7	0

A4=	V1	V2	V3	V4
V1	0	3	5	6
V2	5	0	2	3
V3	3	6	0	1
V4	2	5	7	0

A[V1,V2]= **min(A[V1,V2], A[V1,V3]+A[V3,V2])**

```
for(k=1; k<=n; k++){
for(i=1; i<= n; i++){
    for(j=1; j<=n; j++){
        A[i, j]= min( A[i, j], A[i, k]+A[k, j] )
        }
    }
}
```

Time complexity = O (n^3)

<u>Algorithm: Floyd-Warshall</u>

Procedure: floydWarshall (G, w)
for each edge (u, v | u,v ε V) do
    A[u][v] ← w(u, v)
    next [u][v] ← v
for each vertex v ε V do
    A[v][v] ← 0
    next [u][v] ← v
while (k< |V|) do
    while (i< |V|) do
        while (j< |V|) do
            A[i][j] = minimum (A[i][j], A[i][k]+A[k][j])

```
                        next[i][j] ← next[i][k]
                end do
        end do
end do
end procedure
```

How do you apply greedy approach of this problem?
How it differ from Dijkstra algorithm.