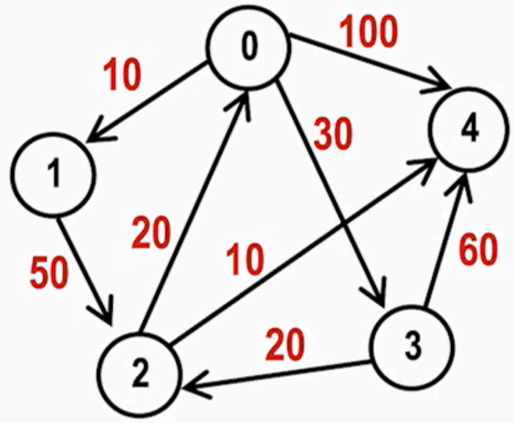
The All-Pairs Shortest Paths Problem (APSP)

APSP

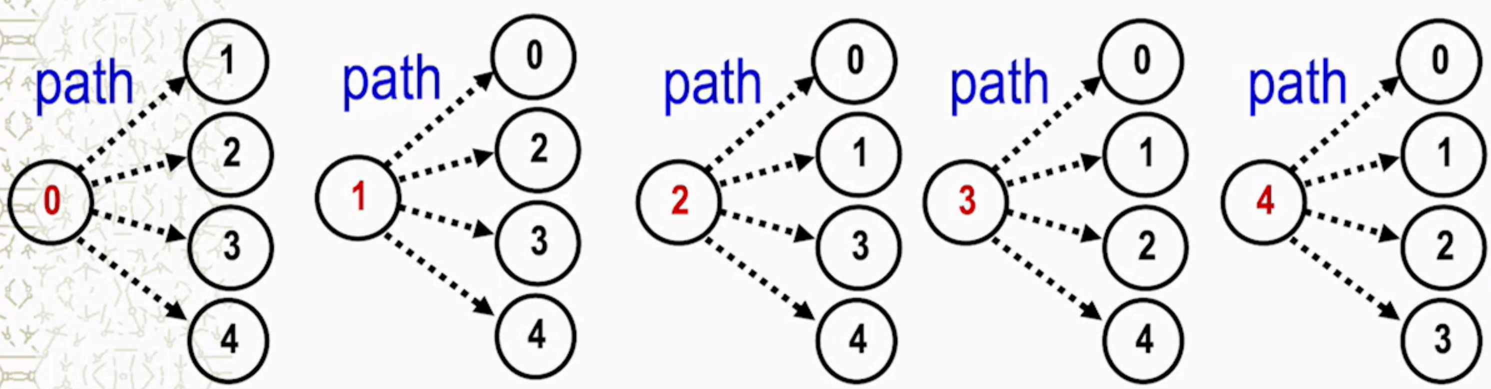
- **Given:**
  - Directed graph G = (V, E) in which each arc has a nonnegative label



- **Problem:**
  - Find the cost of the shortest path from any given SOURCE vertex to any DESTINATION vertex
  - Cost of the path may represent something different like time.

APSP

- The APSP problem is to find for each ordered pair of vertices (v,w) the smallest length of any path from v to w.
- This problem could be solve by using **Dijkstra's** algorithm with each vertex in turn as source. V = { 0, 1, 2, 3, 4}



APSP

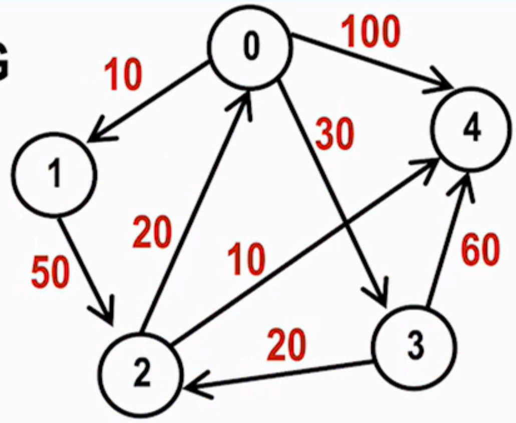
- A more direct way of solving the problem is to use the following algorithm due to **R. W. Floyd**.
- This algorithm will compute the shortest path from any given vertex to any other vertex given the **arc cost matrix C**. The result is stored in an **nxn matrix A**.

Floyd's Algorithm

```
function Floyd (int A[][], int C[][])
{
    int i, j, k;
    for (i=0; i < n; i++)
      for (j = 0; j < n ; j++)
          A[i][j] = C[i][j];
    for (i=0; i< n; i++)
        A[i][i] = 0;
    for( k = 0; k < n; k++)
      for (i=0; i < n; i++)
        for (j = 0; j < n ; j++)
            if ( A[i][k] + A[k][j] < A[i][j])
                 A[i][j] = A[i][k] + A[k][j];
} /* Floyd */
```

Simulation: Floyd's Algorithm

Graph G



Adj. matrix C

	0	1	2	3	4
0	∞	10	∞	30	100
1	∞	∞	50	∞	∞
2	20	∞	∞	∞	10
3	∞	∞	20	∞	60
4	∞	∞	∞	∞	∞

APSP matrix

	0	1	2	3	4
0	0	10	50	30	60
1	70	0	50	100	60
2	20	30	0	50	10
3	40	50	20	0	30
4	∞	∞	∞	∞	0