Floyd Warshall Algorithm(All pair shortest path)

The Floyd-Warshall algorithm is an example of dynamic programming. It breaks the problem down into smaller subproblems, then combines the answers to those subproblems to solve the big, initial problem.

Given a weighted graph g = (V, E), where V = {1,2,.., n}. The Floyd Warshall algorithm gives the shortest path between any pair of nodes in the graph. Assume that weights are represented in a matrix C[V][V], where C[i][j] indicates the weight (or cost) between the nodes i and j. Also, C[i][j] = ∞, if there is no direct path from node i to node j.

Floyd's algorithm for all pair shortest path problems uses matrix A[1...n][1...n] to compute the lengths of the shortest path. Initially

$$A[i,j] = C[i][j] \text{ if } i \neq j$$
$$= 0 \text{ if } i = j$$

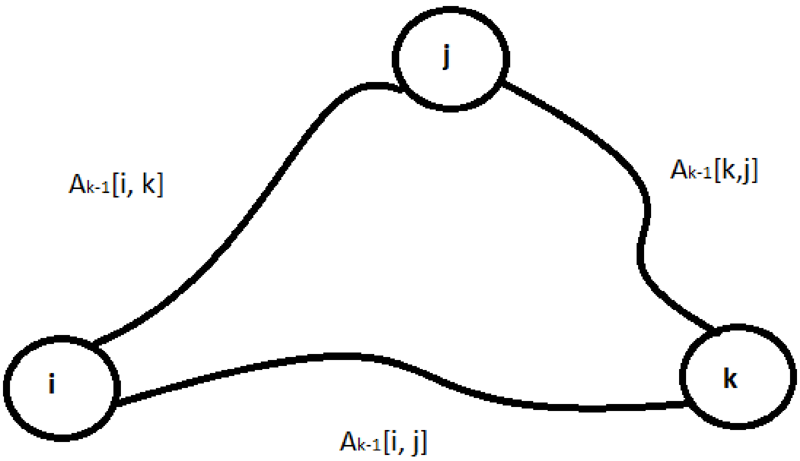
From the definition, C[i, j] = ∞, if there is no path from i to j. The algorithm makes n passes over A. Let A_0, A_1, ..., A_n be the values of A on the n passes, with A0 being the initial value.

Just after the k-1th iteration, $A_{k-1}[i, j]$ = smallest length of any path from vertex i to vertex j that does not pass through the vertices {k+1, k+2, ...n}. That means, it passes through the vertices possibly through {1,2,3,..., k-1}.

In each iteration, the value A[i][j] is updated with the minimum of $A_{k-1}[i, j]$ and $A_{k-1}[i, k]+A_{k-1}[k, j]$.

$$\mathbf{A[i][j] = minimum\ of\ (A_{k-1}[i,j],\ A_{k-1}[i, k]+ A_{k-1}[k, j])}$$

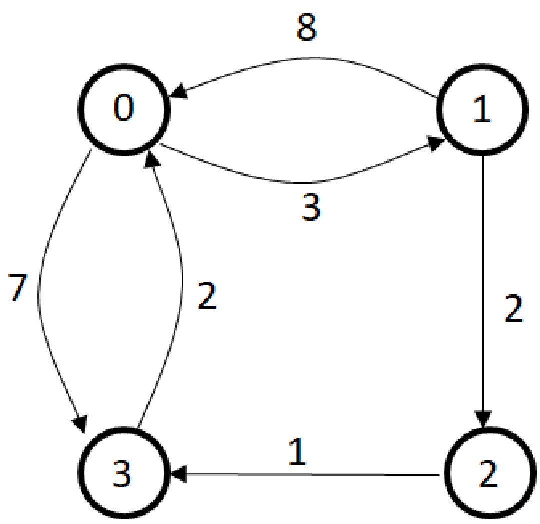
The kth pass explores whether the vertex k lies on an optimal path from i to j, for all i, j. The same is shown in diagram below



```
function floyd(C, A, n){
        //  initialize 3 variables i, j and k
        i = 0

        while(i <= n-1)
            j = 0
            while(j <= n-1)
                A[i][j] = C[i][j]
                j++
            i++

        i = 0
        while(i <= n-1)
            A[i][i] = 0
            i++

        k = 0
        while(k <= n-1)
            i = 0
            while(i <= n-1)
                j=0
                while(j <= n-1)
                    if(A[i][k]+A[k][j] < A[i][j])
                        A[i][j] = A[i][k]+A[k][j]
}
```

Time Complexity = $O(n^3)$

Example: Consider the directed graph below



The adjacency matrix A for the above graph will be

	0	1	2	3
0	0	3	∞	7
1	8	0	2	∞
2	∞	∞	0	1
3	2	∞	∞	0

Now initially the matrix C will be the same as matrix A.

How is the update of vertices done?

Consider k=0, i=1 and j=3

A[i][k] =8, A[k][j] = 7 and A[i][j] = INF,

Therefore A[i][k]+A[k][j] < A[i][j], so update A[i][j] = 15 which A[i][k]+A[k][j]

Now for different values of k, the C matrix will look like as shown below:

	0	1	2	3
0	0	3	∞	7
1	8	0	2	15
2	∞	∞	0	1
3	2	5	∞	0

k = 0

	0	1	2	3
0	0	3	5	7
1	8	0	2	15
2	∞	∞	0	1
3	2	5	7	0

k = 1

	0	1	2	3
0	0	3	5	6
1	8	0	2	3
2	∞	∞	0	1
3	2	5	7	0

k = 2

	0	1	2	3
0	0	3	5	6
1	5	0	2	3
2	3	6	0	1
3	2	5	7	0

k = 3

Note: Both Floyd warshall and Bellman Ford algorithm are an example of dynamic programming where as Dijikstra's Algorithm is based on greedy technique