模版·Floyd(求全对最短路径，权值可负但不可环)

```
class Edge{
    int v;
    int weight;

    Edge(int v,int weight){
        this.v=v;
        this.weight=weight;
    }
}
class Graph{
    int V;
    LinkedList<Edge>[] adjList;

    Graph(int V){
        this.V=V;
        adjList=new LinkedList[V];
        for(int i=0;i<V;i++){
            adjList[i]=new LinkedList<>();
        }
    }

    void addEdge(int u,int v,int weight){
        adjList[u].add(new Edge(v,weight));
    }

    static void initializeDistanceMatrix(Graph graph,int[][] dist){
        for(int i=0;i<graph.V;i++){
            Arrays.fill(dist[i],Integer.MAX_VALUE);
            dist[i][i]=0;
            for(Edge edge:graph.adjList[i]){
                dist[i][edge.v]=edge.weight;
            }
        }
    }

    static void floydWarshall(Graph graph){
        int V=graph.V;
        int[][] dist=new int[V][V];

        initializeDistanceMatrix(graph,dist);

        for(int k=0;k<V;k++){
            for(int i=0;i<V;i++){
                for(int j=0;j<V;j++){
                    if(dist[i][k]!=Integer.MAX_VALUE&&dist[k][j]!=Integer.MAX_VALUE&&dist[i][k]+dist[k][j]<dist[i][j]){
                        dist[i][j]=dist[i][k]+dist[k][j];
                    }
                }
            }
        }

        printSolution(dist);
    }

    static void printSolution(int[][] dist){
        int V=dist.length;
        for(int[] ints:dist){
            for(int j=0;j<V;j++){
                if(ints[j]==Integer.MAX_VALUE){
                    System.out.print("INF");
                }else{
                    System.out.print(ints[j]+"");
                }
            }
            System.out.println();
        }
    }
}
public static class Floyd{
    public static void main(String[] args){
        int V=4;
        Graph graph=new Graph(V);

        graph.addEdge(0,1,5);
        graph.addEdge(0,3,10);
        graph.addEdge(1,2,3);
        graph.addEdge(2,3,1);

        floydWarshall(graph);
        }
    }
}
```