Floyds

```
import java.util.*;
class Floyds{
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the number of vertices: ");
    int n =sc.nextInt();
    System.out.println("Enter the adj matrix:(enter 999 for infinity) ");
    int adj[][] = new int[10][10];
    for(int i=1;i<=n;i++)
      for(int j=1;j<=n;j++)
        adj[i][j] = sc.nextInt();
    flyod(adj,n);
    System.out.println("the all pair shoretst path is: ");
    for(int i=1;i<=n;i++){
      for(int j=1;j<=n;j++){
        System.out.print(adj[i][j]+" ");
      }
      System.out.println();
    }
  }
  static void flyod(int arr[][],int n){
    for(int k=1;k<=n;k++)
      for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
          arr[i][j] = min(arr[i][j],(arr[i][k]+arr[k][j]));
  }
  static int min(int a,int b){
    if(a<b)
      return a;
    return b;
  }
}
```

Bellman Ford

```
class Graph {
  static class Edge {
    int src, dest, weight;
    Edge(int s, int d, int w) {
      src = s;
      dest = d;
      weight = w;
    }
  };
  int V, E;
  Edge edge[];
  Graph(int v, int e) {
    V = v;
    E = e;
    edge = new Edge[e];
  }
  void BellmanFord(Graph graph, int src) {
    int V = graph.V, E = graph.E;
    int dist[] = new int[V];
    for (int i = 0; i < V; ++i)
      dist[i] = Integer.MAX_VALUE;
    dist[src] = 0;
    for (int i = 1; i < V; ++i) {
      for (int j = 0; j < E; ++j) {
        int u = graph.edge[j].src;
        int v = graph.edge[j].dest;
        int weight = graph.edge[j].weight;
        if (dist[u] != Integer.MAX_VALUE && dist[u] + weight < dist[v])
          dist[v] = dist[u] + weight;
      }
    }
    for (int j = 0; j < E; ++j) {
      int u = graph.edge[j].src;
      int v = graph.edge[j].dest;
      int weight = graph.edge[j].weight;
      if (dist[u] != Integer.MAX_VALUE && dist[u] + weight < dist[v]) {
        System.out.println("Graph contains negative weight cycle");
        return;
      }
    }
    printArr(dist, V);
  }
  void printArr(int dist[], int V) {
    System.out.println("Vertex Distance from Source");
    for (int i = 0; i < V; ++i)
      System.out.println(i + "\t" + dist[i]);
  }
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.print("Enter no. of vertices: ");
    int V = in.nextInt();
    System.out.print("Enter no. of edges: ");
    int E = in.nextInt();
    Graph graph = new Graph(V, E);
    for (int i = 0; i < E; i++) {
      System.out.print("Enter src, dest and weight for edge " + (i + 1) + " : ");
      int src = in.nextInt();
      int dest = in.nextInt();
      int weight = in.nextInt();
      graph.edge[i] = new Edge(src, dest, weight);
    }
    graph.BellmanFord(graph, 0);
  }
}
```