# 模版·Dijkstra(求单源最短路径，权值非负)

```
class Node{
    int val;
    int distance;

    public Node(int val,int distance){
        this.val=val;
        this.distance=distance;
    }
}

class Graph{
    private Map<Integer,List<Node>> adjacencyList;

    public Graph(int vertices){
        this.adjacencyList=new HashMap<>(vertices);
        for(int i=0;i<vertices;i++){
            adjacencyList.put(i,new ArrayList<>());
        }
    }

    public void addEdge(int source,int destination,int weight){
        adjacencyList.get(source).add(new Node(destination,weight));
    }

    public void findShortestPaths(int source){
        int vertices=adjacencyList.size();
        boolean[] visited=new boolean[vertices];
        int[] distances=new int[vertices];
        Arrays.fill(distances,Integer.MAX_VALUE);
        distances[source]=0;

        PriorityQueue<Node> priorityQueue=new PriorityQueue<>((a,b)->Integer.compare(a.distance,b.distance));
        priorityQueue.add(new Node(source,0));

        while(!priorityQueue.isEmpty()){
            Node current=priorityQueue.poll();
            int val=current.val;
            int distance=current.distance;

            if(visited[val]){
                continue;
            }

            visited[val]=true;
```

```
                for(Node neighbor:adjacencyList.get(val)){
                    int neighborVertex=neighbor.val;
                    int newDistance=distance+neighbor.distance;

                    if(newDistance<distances[neighborVertex]){
                        distances[neighborVertex]=newDistance;
                        priorityQueue.add(new Node(neighborVertex,newDistance));
                    }
                }
            }

            for(int i=0;i<vertices;i++){
                System.out.println("Source:"+source+",Destination:"+i+",Distance:"+distances[i]);
            }
        }
    }

    public class Dijkstra{

        public static void main(String[] args){
            Graph graph=new Graph(5);

            graph.addEdge(0,1,4);
            graph.addEdge(0,2,3);
            graph.addEdge(1,2,1);
            graph.addEdge(1,3,2);
            graph.addEdge(2,3,4);
            graph.addEdge(3,4,2);

            graph.findShortestPaths(0);
        }
    }
```