import java.util.\*;  
  
class Graph {  
 private int V; // 图中顶点的数量  
 private LinkedList<Edge>[] adj; // 邻接表，用于存储图的边  
 private int[] earliest; // 存储最早开始时间  
 private int[] latest; // 存储最晚开始时间  
  
 Graph(int v) {  
 V = v;  
 adj = new LinkedList[v];  
 earliest = new int[v];  
 latest = new int[v];  
 for (int i = 0; i < v; ++i) {  
 adj[i] = new LinkedList<>(); // 初始化邻接表  
 }  
 }  
  
 void addEdge(int u, int v, int weight) {  
 Edge edge = new Edge(u, v, weight); // 创建一条边  
 adj[u].add(edge); // 将边加入邻接表中  
 }  
  
 void topologicalSortUtil(int v, boolean visited[], Stack<Integer> stack) {  
 visited[v] = true;  
 for (Edge edge : adj[v]) {  
 if (!visited[edge.v])  
 topologicalSortUtil(edge.v, visited, stack);  
 }  
 stack.push(v);  
 }  
  
 void criticalPath() {  
 // 计算最早开始时间 earliest[]  
 Arrays.fill(earliest, 0);  
 for (int i = 0; i < V; i++) {  
 for (Edge edge : adj[i]) {  
 int v = edge.v;  
 int weight = edge.weight;  
 earliest[v] = Math.max(earliest[v], earliest[i] + weight);  
 }  
 }  
  
 // 计算最晚开始时间 latest[]  
 Arrays.fill(latest, earliest[V - 1]);  
 for (int i = V - 1; i >= 0; i--) {  
 for (Edge edge : adj[i]) {  
 int v = edge.v;  
 int weight = edge.weight;  
 latest[i] = Math.min(latest[i], latest[v] - weight);  
 }  
 }  
  
 // 计算关键路径  
 for (int i = 0; i < V; i++) {  
 for (Edge edge : adj[i]) {  
 int v = edge.v;  
 int weight = edge.weight;  
 int slack = latest[v] - earliest[i] - weight;  
 if (slack == 0)  
 System.out.println("边 " + i + " -> " + v + " 在关键路径上");  
 }  
 }  
 }  
  
 class Edge {  
 int u, v, weight;  
  
 Edge(int u, int v, int weight) {  
 this.u = u;  
 this.v = v;  
 this.weight = weight;  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("请输入图的节点个数：");  
 int n = scanner.nextInt();  
 Graph graph = new Graph(n);  
  
 // Adding edges  
 System.out.println("请输入边的数量：");  
 int m = scanner.nextInt();  
 System.out.println("请输入权重（起点 终点 权重）：");  
 for (int i = 0; i < m; i++) {  
 int u = scanner.nextInt();  
 int v = scanner.nextInt();  
 int weight = scanner.nextInt();  
 graph.addEdge(u, v, weight);  
 }  
  
 System.out.println("关键路径：");  
 graph.criticalPath();  
 }  
}