```cpp
#include <cstdio>
#include <cstdlib>
#include <queue>
using namespace std;
#define MAX 105
#define INFINITY 65535
int N, M, A[MAX][MAX], ECT, EarliestTime[MAX] = {0},
LatestTime[MAX], D[MAX][MAX], idx;  //ETC--earliest completion
time
int getMax( int arr[] ) {
    int max = 0;
    for(int i = 0; i < N; i++)
        if( max < arr[i] ){
            max = arr[i];
            idx = i;
        }
    return max;
}
int TopSort_Earliest(){
    int V, cnt = 0, Indegree[MAX] = {0};
    queue<int> q;
    //计算各结点的入度
    for( int i = 0; i < N; i++ )
        for( int j = 0; j < N; j++ )
            if( A[i][j] != INFINITY )
                Indegree[j]++;  //对于有向边<i,j>累计终点j的入度
    //入度为0的入队
    for( int i = 0; i < N; i++ )
        if( Indegree[i] == 0 )
            q.push(i);
    while( !q.empty() ) {
        V = q.front();
        q.pop();
        cnt++;
        for( int j = 0; j < N; j++ )
            if( A[V][j] != INFINITY ) {  //<V, j>有有向边
                if( EarliestTime[V] + A[V][j] > EarliestTime[j] ){
//如果 V的最早完成时间 + j所需时间 > j的最早完成时间
                    EarliestTime[j] = EarliestTime[V] + A[V][j];
                }
                if( --Indegree[j] == 0 )  //去掉V后，如果j的入度为0
                    q.push(j);
            }
    }
    ECT = getMax(EarliestTime); //最早完成时间应是所有元素中最大的
    if( cnt != N ) return 0;    //如果没有取出所有元素，说明图中有回路
```

```
    else return 1;
}
void TopSort_Latest() {
    int V, Outdegree[MAX] = {0};
    queue<int> q;
        //计算各结点的出度
    for( int i = 0; i < N; i++ )
        for( int j = 0; j < N; j++ )
            if( A[i][j] != INFINITY )
                Outdegree[i]++; //对于有向边<i,j>累计起点i的出度
    //出度为0的入队
    for( int i = 0; i < N; i++ )
        if( Outdegree[i] == 0 )
            q.push(i);
    //初始化LatestTime
    for( int i = 0; i < N; i++ )
        LatestTime[i] = INFINITY;
    LatestTime[idx] = ECT;  //将最后一个活动的最晚完成时间设为它的最早完成
时间
    while( !q.empty() ) {
        V = q.front();
        q.pop();
        //cnt++;    //不需要再算cnt了
        for( int j = 0; j < N; j++ )
            if( A[j][V] != INFINITY ) {  //<j, V>有有向边
                if( LatestTime[V] - A[j][V] <= LatestTime[j] ) {
//必须用<=,只<的话只能算一条关键路径，<=才能算出所有的关键路径(错误原因)
                    LatestTime[j] = LatestTime[V] - A[j][V];
                    D[j][V] = LatestTime[V] - EarliestTime[j] -
A[j][V];
                }
                if( --Outdegree[j] == 0 )  //去掉V后，如果j的出度为0
                    q.push(j);
            }
    }
}
void PrintKeyRoute() {
    for( int i = 0; i < N; i++ )
        for( int j = N - 1; j >= 0; j-- )   //根据题目要求，i相同时要j
要逆序输出
            if( D[i][j] == 0 )
                printf("%d->%d\n", i + 1, j + 1);
}
int main(){
    int a, b;
    scanf("%d %d", &N, &M);
    //初始化图的边A，各组的机动时间D
```

```
    for( int i = 0; i < N; i++ )
        for( int j = 0; j < N; j++ )
            D[i][j] = A[i][j] = INFINITY;
    //read
    for( int i = 0; i < M; i++ ) {
        scanf("%d %d", &a, &b);
        scanf("%d", &A[--a][--b]);  //题目中编号从1开始（错误原因）
    }
    if( !TopSort_Earliest() )
        printf("0\n");
    else {
        printf("%d\n", ECT);
        TopSort_Latest();
        PrintKeyRoute();
    }
    system("pause");
    return 0;
}
```