**数据结构课程设计**

**-关键活动**

# 题目描述

1. 项目内容

本实验项目是要求在任务调度问题中，如果还给出了完成每个子任务需要的时间，则可以算出完成整个工程项目需要的最短时间。在这些子任务中，有些任务即使推迟几天完成，也不会影响全局的工期；但是有些任务必须准时完成，否则整个项目的工期就要因此而延误，这些任务叫做“关键活动”。

请编写程序判定一个给定的工程项目的任务调度是否可行；如果该调度方案可行，则计算完成整个项目需要的最短时间，并且输出所有的关键活动。

1. 项目功能要求

输入说明：输入第1行给出两个正整数N（N<=100）和M，其中N是任务交接点（即衔接两个项目依赖的两个子任务的结点，例如：若任务2要在任务1完成后才开始，则两个任务之间必有一个交接点）的数量，交接点按1～N编号，M是字任务的数量，依次编号为1～M。随后M行，每行给出3个正整数，分别是该任务开始和完成设计的交接点编号以及完成该任务所需要的时间，整数间用空格分隔。

输出说明：如果任务调度不可行，则输出0；否则第一行输出完成整个项目所需要的时间，第2行开始输出所有关键活动，每个关键活动占一行，按照格式“v->W”输出，其中V和W为该任务开始和完成涉及的交接点编号。关键活动输出的顺序规则是：任务开始的交接点编号小者优先，起点编号相同时，与输入时任务的顺序相反。

# 总体思路

我们直观地想象整个项目最早终止时间一定是所有流程综合最长的时间，代码层面要解决这个问题的关键在于从 AOE 网中找到一条从起始点到结束点长度最长的路径，这样就能保证所有的活动在结束之前都能完成。我们定义起始点是入度为 0 的点，称为“源点”；结束点是出度为 0 的点，称为“汇点”，而这条最长的路径，被称为“关键路径”。

为了求出一个给定AOE网的关键路径，需要知道以下四个统计数据：

对于AOE网中的顶点有两个时间：最早可能开始的时间（用表示）和最迟允许开始时间（用表示）；对于边来说，也有两个时间：最早可能开始的时间（用表示）和最迟允许开始时间（用表示）。

1. ：对于AOE网中的任意一个顶点来说，从源点到该点的最长路径代表着该顶点的最早可能开始的时间。
2. ：表示在不推迟整个工期的前提下，事件允许的最迟允许开始时间，即从该点到汇点的最短路径。
3. ：表示活动的最早可能开始时间，如果活动是由弧表示，那么活动的最早开始的时间就等于事件的最早发生时间，即。
4. ：表示活动的最迟允许开始时间，如果活动是由弧表示，的最晚开始时间要保证的最晚发生时间不拖后，保证汇点结束时间不会延后。即。

在得知以上四种统计数据后，就可以直接求得 AOE 网中关键路径上的所有的关键活动：对于所有的边来说，如果它的最早开始时间等于最晚开始时间，称这条边所代表的活动为关键活动。由关键活动构成的路径为关键路径。但如果图中含有环，则说明AOE网络不成立，需要拓扑排序进行判断是否成环。

# 代码类结构

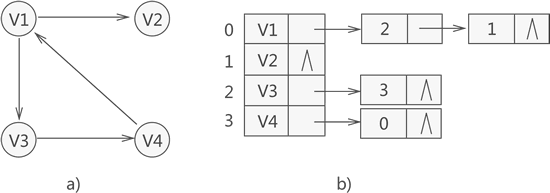
头文件AOE.h中包含两个结构体ArcNode和VNode用于存储边和顶点，和一个AGraph类用于存储图信息和求解关键路径。

以下是用到的数据结构定义：

# 具体实现

## 输入处理

由于建立AOE网络的空间由顶点和边的数量唯一确定，所以所有的输入全放在AGraph的构造函数中，在输入顶点和边的数量后再开辟相关数组空间。本题错误输入仅涉及顶点和边数量的输入和边信息的录入，输入结构相对简单，在这里不做赘述。



如上图所示，AOE网络存储用的邻接表。同一起点的边构成一条链表，每一个顶点对应的链表构成net数组。加入边时，该边起点出度和终点入度加一。每条边按顺序输入，一条边对应一个id，在edgeInfo数组中存储了以id为标识符的边起点终点的信息，由于顶点最多有100个，可以用以下方法存边信息，方便后续按边id打印数组，节省查询邻接表的时间。令该边起点、终点、权重为，则具体实现为，得到的六位数前三位为起点编号u，后三位为终点编号v。

由AOE网络的定义，源点和汇点唯一，因此在进行下述操作拓扑排序和关键路径求解前需要检查是否只有一个入度为零的点和一个出度为零的点，剩下出入度均不为零的点即为中间节点，具体函数实现为checkAOE()。

## 拓扑排序

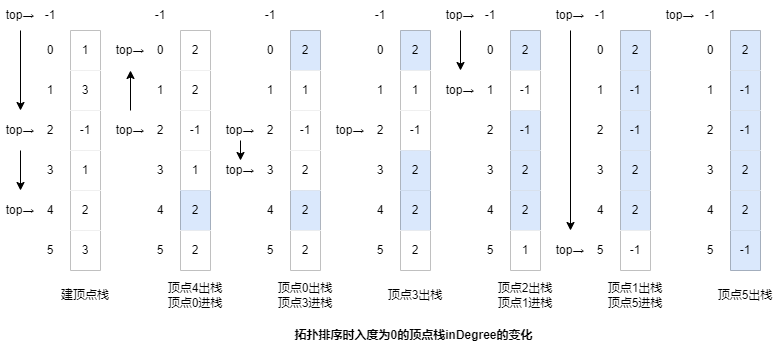
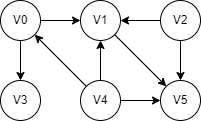
在拓扑排序算法中，需要一个存放入度为零的顶点的链式栈，，供选择输出无前驱的顶点。只要出现顶点入度为零，就将其加入到栈中。具体实现可以描述如下：

1. 建立入度为零的顶点栈
2. 当顶点栈不为空时，重复执行：
3. 从顶点栈退出一个顶点存储在order数组中
4. 在AOE网络中删除所有和这个顶点相关的边，包括它为起点和终点，并调整相应的出度、入度
5. 如果有顶点的入度减为零，则此点入栈
6. 若出栈个数少于节点个数，则说明网络存在有向环

而在算法具体实现时，为了此栈不另设存储空间，可直接利用记录入度的数组inDegree。设定top为该数组栈的栈顶指针，指向一个入度为零的顶点位置。初始化设为-1，表示空栈。出入栈的代码和图例如下：

顶点i入栈：

出栈操作：



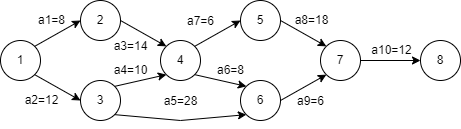
## 关键路径求解

按照拓扑排序的顺序，依次访问当前顶点temp为起点的所有边，更新这些边的终点的，取，而这个操作和拓扑排序可同时进行，代码如下：

与上述求相反，需要按照拓扑排序的逆序进行更新。在order数组中存储了正序拓扑排序，用top指针从后往前遍历，先访问汇点，令其和相等，再访问order数组，得到的顶点v遍历以其为起点的边，更新顶点v的，记访问的某边为,取，实现代码如下：

最后根据求得的和计算和，遍历net数组，数组中的每一项为以为顶点的链表，再遍历该链表，以此遍历所有的边，且每个边有编号id。根据和计算结果。如果，则说明该边为关键路径。

以下的表格1和2为该图例的求解结果：



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  | 0 | 8 | 12 | 22 | 28 | 40 | 46 | 58 |
|  | 0 | 8 | 12 | 22 | 28 | 40 | 46 | 58 |

表 1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| e | 0 | 0 | 8 | 12 | 12 | 22 | 22 | 28 | 40 | 46 |
| l | 0 | 0 | 8 | 12 | 12 | 32 | 22 | 28 | 40 | 46 |

表 2

# 项目小结

1. 性能分析

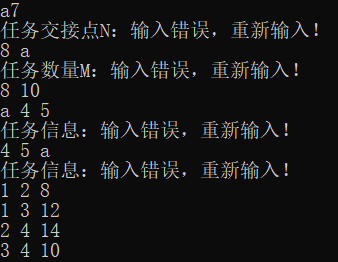
分析拓扑排序算法可知，如果AOE网络有n个顶点，e条边，在TopSort中建立顶点栈需要时间，每个顶点进一次栈、出一次栈，共执行n次。顶点入度和出度的修改共执行了e次，所以拓扑排序总时间复杂度为。逆拓扑排序算时，总迭代了n次，内部遍历的边总数为e，所需时间也为。计算和的时间需求为边的个数。所以整个程序时间复杂度为。在AGraph类中，我使用了8个数组指针，其中四个大小为n，另外四个为e，总空间复杂度为，即。

1. 结果分析

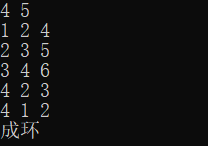
关键路径的求解其实是AOE图最长通路的求解，关键路径决定了整个项目的时间长度，缩短所求的关键路径的进行时间可以缩短整个项目的完成时间。但注意不是任意关键路径加速一定能使整个工程提前，如以上所举的例子，由于1->2->4和1->3->4的总耗时一样，导致这两条线路上的活动均为关键活动，但单一缩短一条支路上的某活动，并能使的1->4的最长路径缩短。因此要想整个工程提前，需要考虑各个关键路径上所有关键活动。

# 项目功能测试

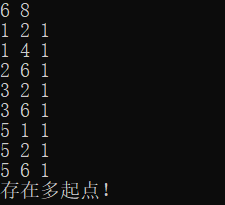
输入错误处理：



成环判断演示：



多起点和多终点：



正常情况：

