# 基于 DAG 图的拓扑排序算法：Kahn算法

## 邻接矩阵和邻接表

在作业《算法2_邻接矩阵和邻接表实现》中已经概括并简要实现过这两种数据结构，故此处省略邻接矩阵和邻接表的相关介绍内容，不加赘述。

## 关于 DAG 图的基础补充知识

### 常见术语

- 度数：由一个顶点出发，有几条边就称该顶点有几度，或者该顶点的度数是几
  - 出度：由一个顶点出发，有几条指向其它顶点的边，就称有几个出度
  - 入度：有几条指向从其它顶点出发且指向该顶点的边，就称有几个入度
- 简单路径：没有重复顶点的路径
- 简单环：不含有重复顶点和边的环
- 连通图：如果一个图中，从任意顶点均存在一条边可以到达另一个任意顶点，我们就说这个图是个连通图
- 稀疏图：图中每个顶点的度数都不是很高，看起来很稀疏
- 稠密图：图中的每个顶点的度数都很高，看起来很稠密
- 二分图：可以将图中所有顶点分为两部分的图

本次作业中要研究的算法是有向无环图的拓扑排序算法

## 基于 DAG 网的拓扑排序实现

### 拓扑排序简介

由某个集合上的一个偏序得到该集合上的一个全序，这个操作称之为拓扑排序。

拓扑排序是一种对一个有向图构造拓扑序列，解决工程是否能顺利进行的问题的算法。

使用通俗的语言解释，拓扑排序就是基于有向无环图遍历所有节点，确保每两个节点之间有先后顺序，并且在搜索下一个节点时，这个节点之前的节点已经全部被搜索过，这种方法就是拓扑排序。

### Kahn算法实现 DAG 图拓扑排序

1. 遍历所有图顶点, 把入度为0的顶点入队
2. 当队列不为空时, 取出一个顶点v输出
3. 将与 v 相邻的顶点入度减1, 然后把减1后入度为0的顶点入队
4. 重复2, 3步, 直到队列为空, 算法结束

经过上面 4 步的重复，即可实现Kahn算法，事实上，该算法的核心就是维护一个入度为 0 的顶点队列

### Kahn算法的时间复杂度

显然，Kahn算法中的主要时间花费用于顶点队列的维护过程，为$O(E + V)$

### Kahn算法的应用

- 判断一个是否有环
- 处理依赖性任务规划问题
- 用于项目管理、 数据库查询优化和矩阵乘法的次序优化

## 程序清单：`/coding/code10.cpp`