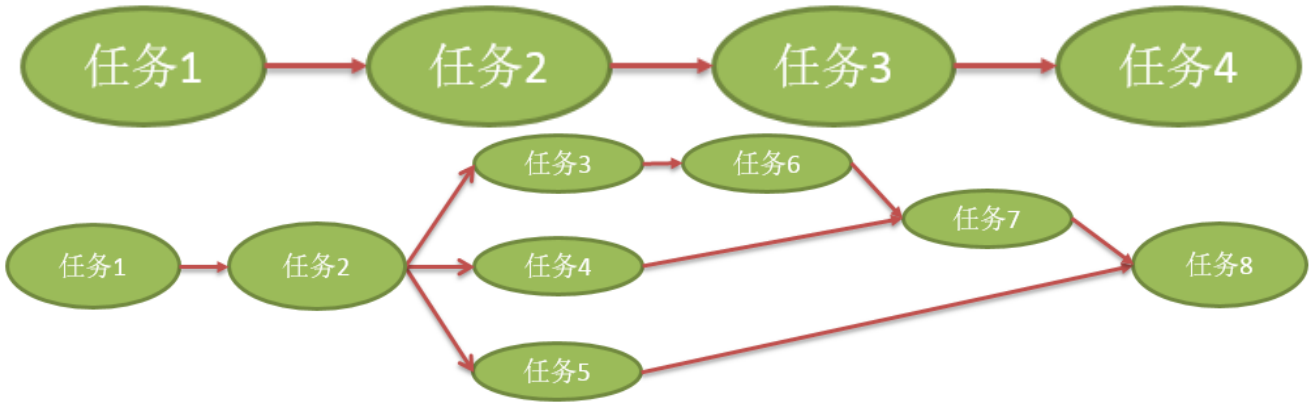
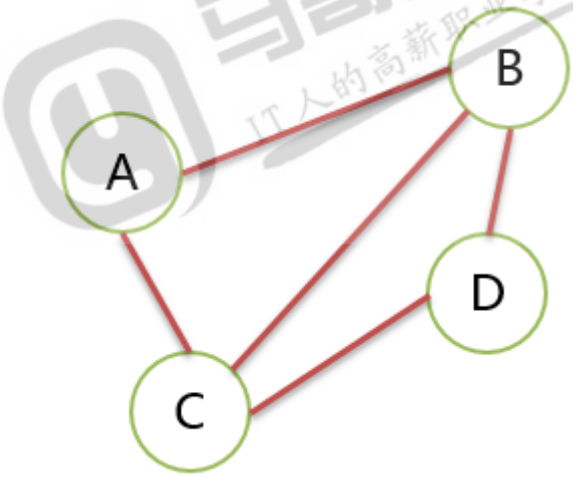
概述

某一个节点上有一批任务需要执行，如何执行？
一个接一个排好队开始执行，但是这样的执行可能很没有效率，而且没有必要。举例，获取主机名和获取IP地址不需要严格的顺序，谁先执行都可以，同时执行也没有问题，这样的任务能并行就并行，不需要串行。



如何实现多个任务的流程执行呢？如何描述这样的任务呢？
看上面的任务并行图，不能使用“树”来描述任务的关系。这里需要使用“图”来描述。

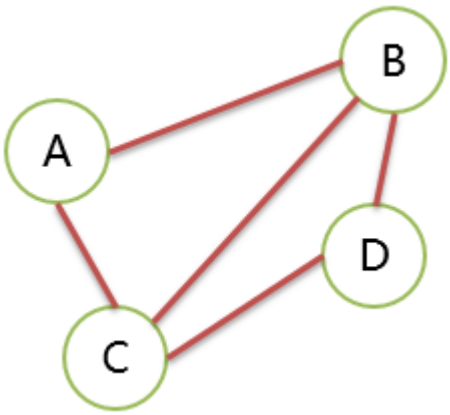
图Graph



经典定义：图Graph由**顶点**和**边**组成，顶点的有穷非空集合为V，边的集合为E，记作`G(V, E)`。
顶点Vertex，数据元素的集合，顶点的集合，有穷非空；
边Edge，数据元素关系的集合，顶点关系的集合，可以为空。

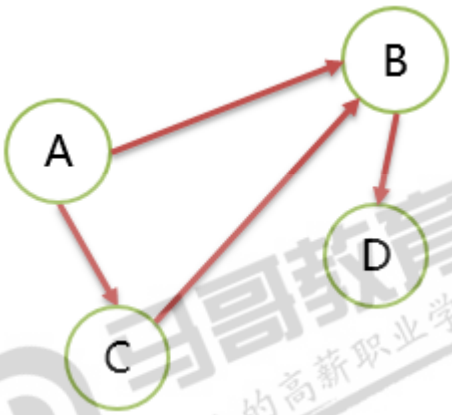
边可以有方向。
无向边记作`(A,B)`或者`(B,A)`，使用小括号。
有向边记作`<A,B>`，即从顶点A指向顶点B。`<B,A>`表示顶点B指向顶点A。使用尖括号。
有向边也叫做弧，边表示为弧尾指向弧头。

图的重要概念



无向图

Undirected Graph
无方向的边构成的图。`G=(V,E) V={A,B,C,D}  E={(A,B),(A,C),(B,C),(B,D),(C,D)}`



有向图

Directed Graph
有方向的边构成的图。`G=(V,E) V={A,B,C,D}  E={<A,B>,<A,C>,<C,B>,<B,D>}`

稀疏图

Sparse Graph
图中边很少。最稀疏的情况，只有顶点没有边，这就是数据结构Set。

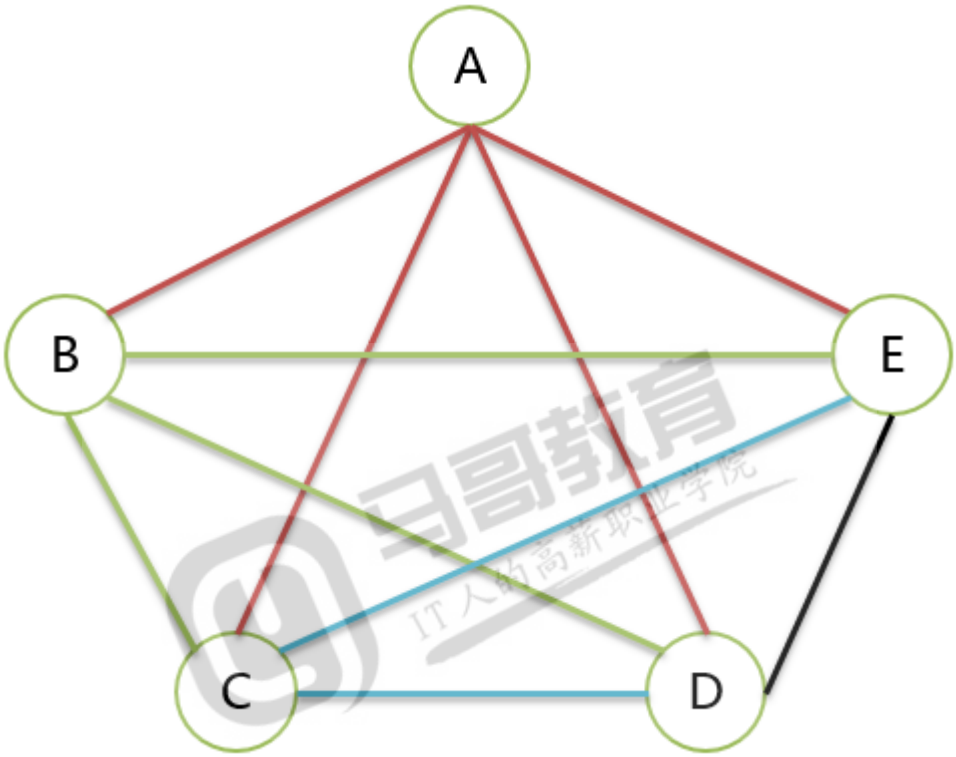
稠密图

Dense Graph
图中边很多。最稠密的情况，任意2个顶点之间都有关系。

完全图

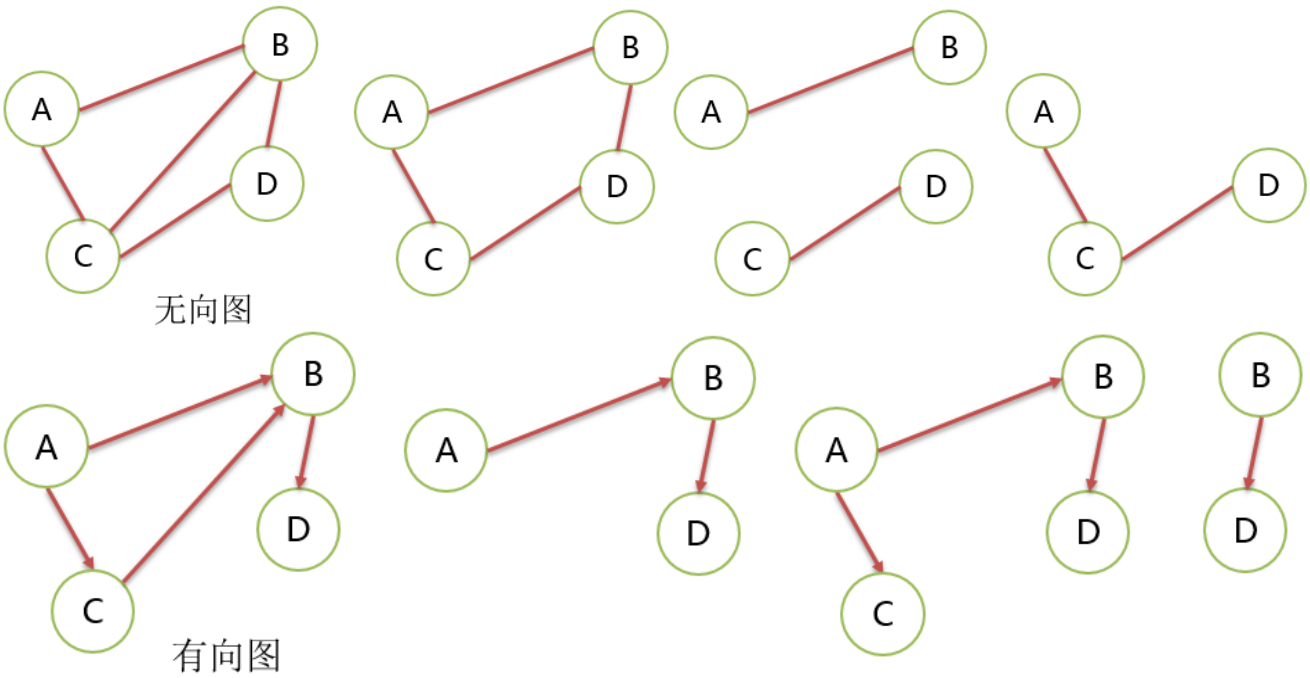
Complete Graph
包括了所有可能的边，达到了稠密图**最稠密**的情况，任意2个顶点之间都有边相连。
有向的边的完全图，叫做有向完全图。边数为`n(n-1)`
无向的边的完全图，叫做无向完全图。边数为`n(n-1)/2`

顶点数	无向	有向
1	0	0
2	1	2
3	3	6
4	6	12
5	10	20



子图

如果图G(V, E)和G'(V', E') 满足V'≤ V 且 E'≤E，则G'是G的子图。
换句话说，就是一个图的部分顶点和部分边组成的图为子图，有向图要注意边的方向。

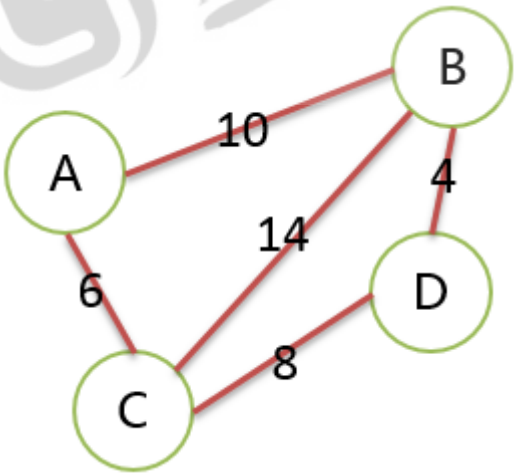


边的权Weight

给边赋予的值称为权。权可以表示距离、所需时间、耗费的时间等。
约定，后面默认说的图，都是不带权的。

网network

图中的边有权，图称为网



自环Loop

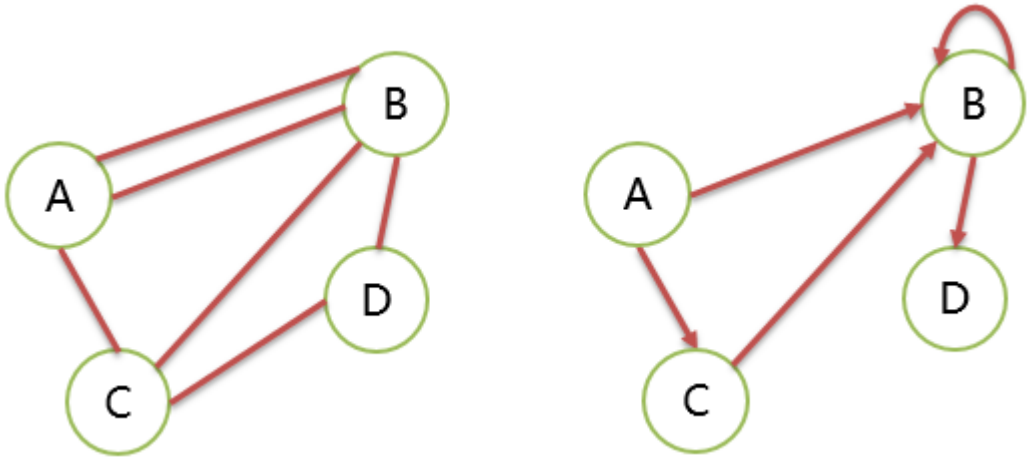
若一条边的两个顶点为同一个顶点，则此边称作自环。
边中存在这样一个边`(u,v)`或者`<u,v>`，`u=v`。

简单图

无重复的边或者顶点到自身的边（自环）的图。

我们以后讨论的是简单图的性质。

下面2个图都不是简单图



邻接

图的边集为E。
无向图，若`(u,v) ∈ E`，则称u和v相互邻接，互为邻接顶点。
有向图，若`<u,v> ∈ E`，则称u邻接到v，或v邻接于u
简单说，就是2点之间有条边，2点邻接。

关联（依附）

若`(u,v) ∈ E`或者 若`<u,v> ∈ E`，则称边依附于顶点u、v 或 顶点u、v与边相关联。

度Degree

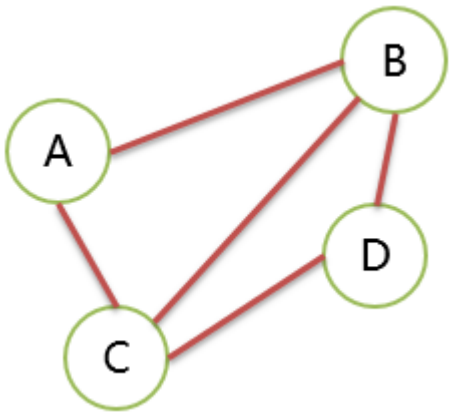
一个顶点的度是指与该顶点相关联的边的条数，顶点v的度记作TD(v)。
无向图顶点的边数叫做度。
有向图的顶点有入度和出度，顶点的度数为入度和出度之和TD(v)=ID(v)+OD(v)。
入度（In-degree）：一个顶点的入度是指与其关联的各边之中，以其为终点的边数
出度（Out-degree）：出度则是相对的概念，指以该顶点为起点的边数。

路径Path

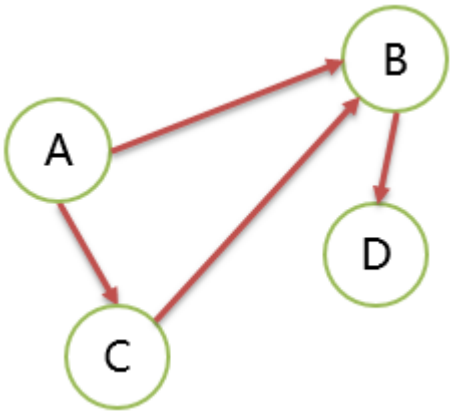
图G(V,E)，其任意一个顶点序列，相邻2个顶点都能找到边或弧依次连接，就说明有路径存在。有向图的弧注意方向。所有顶点都属于V，所有边都必须属于E。

路径长度
等于顶点数-1，等于此路径上的边数

简单路径
路径上的顶点不重复出现，这样的路径就是简单路径



无向图中A到D的路径有ABD、ABCD、ACD、ACBD等

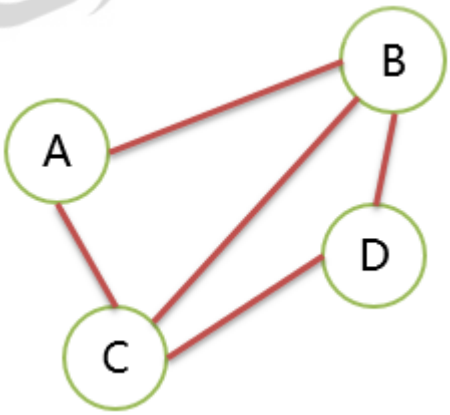


有向图中A到D的路径有ABD、ACBD等

回路

路径的起点和终点相同，这条路径就是回路。

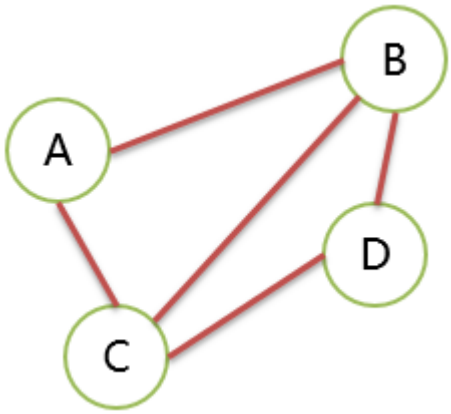
简单回路
除了路径的起点和终点相同外，其它顶点都不同。



ABCD就是简单路径。
ACDBA就是简单回路。
ABCDBA是回路，但是不是简单回路。

连通

无向图中，顶点间存在**路径**，则两顶点是连通的。
注意：连通指的是顶点A、D间有路径，而不是说，这两个顶点要邻接。

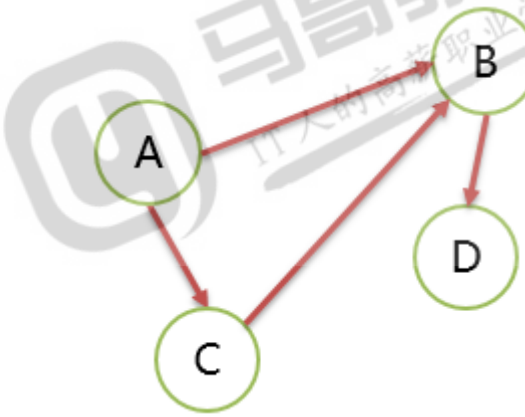


例如A到D存在路径，则A、D顶点是连通的。

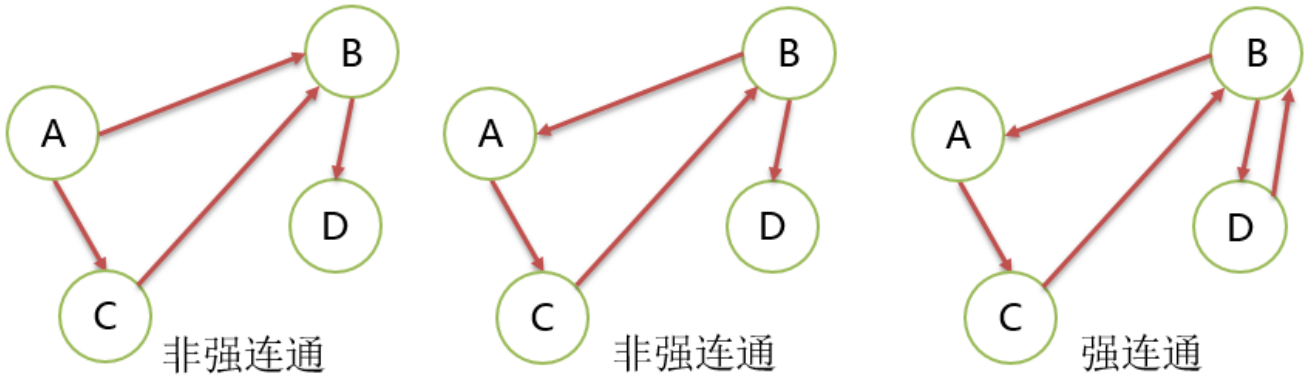
连通图
无向图中，如果图中任意两个顶点之间都连通，就是连通图

连通分量
无向图中，指的是“极大连通子图”。
无向图未必是连通图，但是它可以包含连通子图。

强连通
有向图中，顶点间存在2条相反的路径，即从A到B有路径，也存在从B到A的路径，两顶点是强连通的。
下图就没有2个顶点是强连通的。



强连通图
有向图中，如果图中任意2个顶点都是强连通的图。



强连通分量
有向图中，指的是“极大强连通子图”。
有向图未必是强连通图，但是可以包含强连通的分量。

生成树

它是一个极小连通子图，它要包含图的所有n个顶点，但只有足以构成一棵树的n-1条边。

- 如果一个图有n个顶点，且少于n-1条边，那么一定是非连通图。因为至少要n-1条边才行啊
- 如果一个图有n个顶点，且多于n-1条边，那么一定有环存在，一定有2个顶点间存在第二条路径。但是不一定是连通图，但是一定有环。
- 如果一个图有n个顶点，且有n-1条边，但不一定是生成树。要正好等于n-1条边，且这些边足以构成一棵树

有向树

一个有向图恰好有一个入度为0的顶点，其他顶点的入度都为1。注意，这里不关心出度。

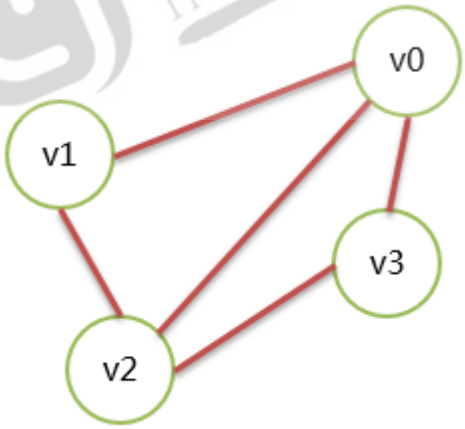
生成树森林
若干有向树构成有向树森林

有向无环图不一定能转化为树（因为可能有交叉），但是树一定是有向无环图

邻接矩阵

图是由vertex和edge组成，所以可以分成2个数组表示。
顶点用一维数组表示，例如v0、v1、v2、v3。
边使用二维数组表示，由顶点构成的二维数组

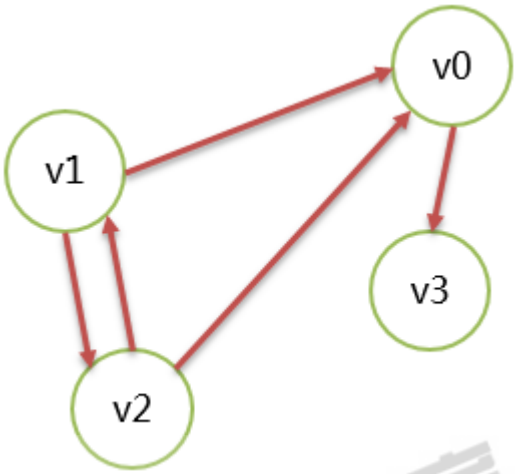
无向图表示
有4个顶点的无向图



	v0	v1	v2	v3	
v0	0	1	1	1	
v1	1	0	1	0	v1度数为2，2条边使用了这个顶点
v2	1	1	0	1	
v3	1	0	1	0	

如果对角线上数字为1，说明出现了自环。
如果除了对角线全是1，说明没有自环，且是一个无向完全图。
上面的矩阵，称为**图的邻接矩阵**。
顶点的度数，等于对应的行或者列求和。
邻接点，矩阵中为1的值对应的行与列对应的顶点就是邻接点。
无向图的邻接矩阵是一个对称矩阵。

有向图表示
有4个顶点的有向图



	v0	**v1**	**v2**	**v3**	
v0	0	0	0	1	
v1	1	0	1	0	v1出度为2，边有`<v1,v0>`、`<v1,v2>`
v2	1	1	0	0	v2出度为2，边有`<v2,v0>`、`<v2,v1>`
v3	0	0	0	0	
	v0入度为2	v1入度为1			

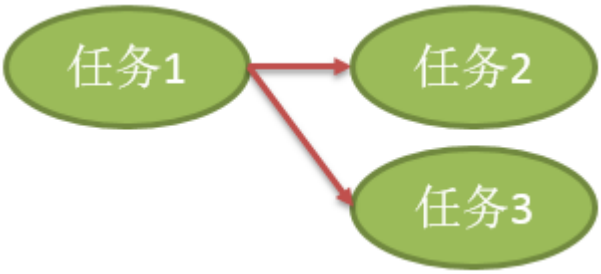
有向图的邻接矩阵不一定对称。对称的说明2个顶点间有环，例如`<v1,v2>`和`<v2,v1>`

如何设计一个有向无环图

有向无环图Directed Acyclic Graph
无环路的有向图。

思考
假设有下面的几种情况

两个任务，任务本身就是顶点，任务先后执行。



三个任务，任务1执行完后，才能分别执行任务2或者任务3。

四个任务，任务1执行完后，才能分别执行任务2或者任务3，最后执行任务4。
要思考任务4执行的前提是`“任务2 or 任务3 做完”`还是`“任务2 and 任务3 做完”`？

可以看到任务的执行过程就是流程的设定（Pipeline），所以要设计一个流程系统来跑任务。

通过上面几个例子，思考：

1. 如何选择执行的起点
2. 如何知道那个任务是终点

起点的选择
入度为0的顶点就是起始的点。
DAG可以有多个起始点。
我们的系统约定有且只能有一个起始点。

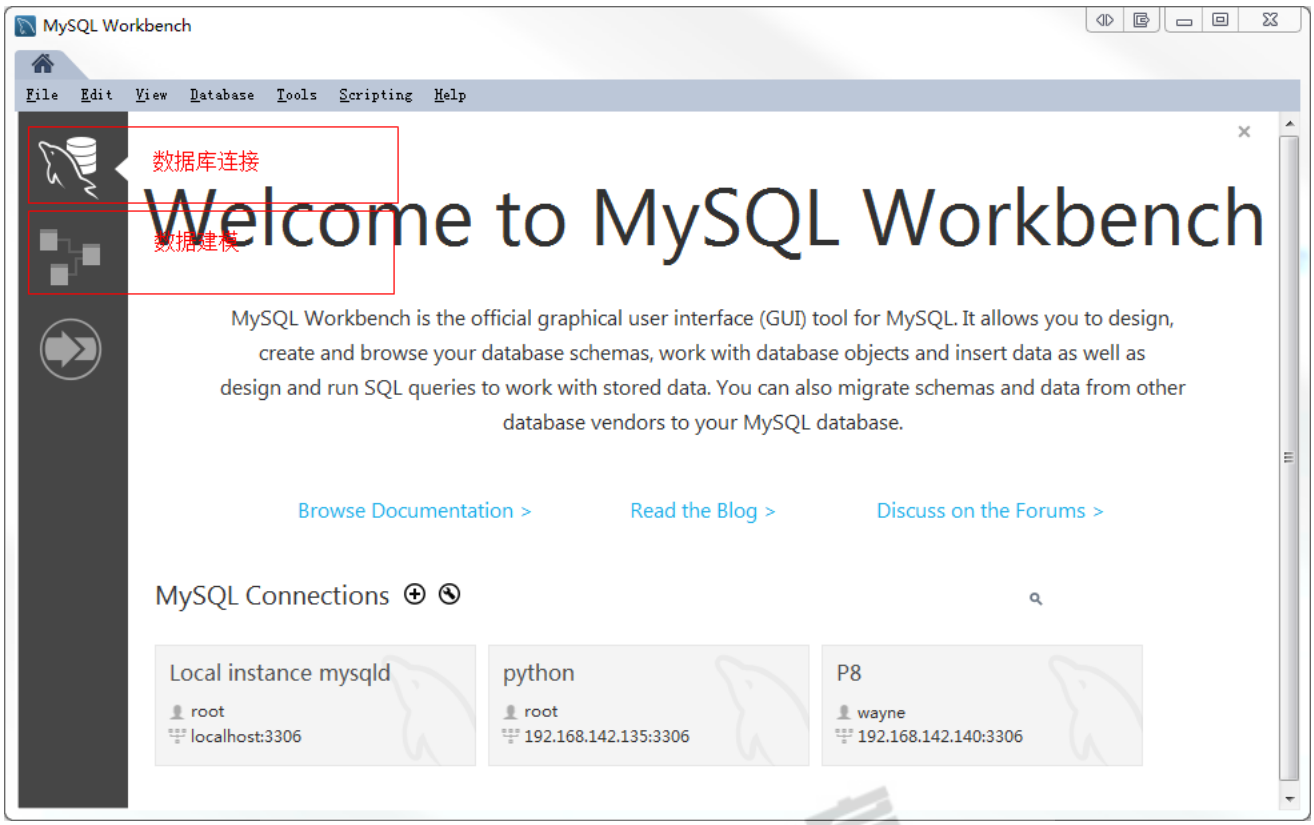
终点的判断
出度为0的顶点，pipeline执行结束。
Pipeline可能有多个终点。

环路检查
Pipeline设计的过程中应当注意避免出现环路，因为出现环路就不是DAG了。
自环检测，弧头指向顶点自身。
多顶点构成环路的检测。
环路检测必须实现，否则当定义好的流程执行起来，有可能进入环路后，永远执行不能终止。

构建模型

工具

模型构建的工具有很多，IBM Rational Rose（现在是Rational Software Architect）、Sybase Power Designer等企业级建模工具。Oracle也提供了一个MySQLWorkbench，使用它的社区版就可以开始模型设计了。



DAG定义

使用数据库表的存储方式定义DAG。
问题是如何使用数据库的表描述一个DAG？
DAG也是图，是图就有顶点、边，所以可以设计2个表，顶点表、边表来描述一个图。为了存储多个图，定义一个图的表。

一个图的定义包含了图的信息、顶点信息、边信息，一张图就是一个流程模板，顶点表示任务，边表示流向

图graph

字段名	类型	说明
id	int	主键
name	varchar	非空、唯一，图的名称
desc	varchar	可为空，描述

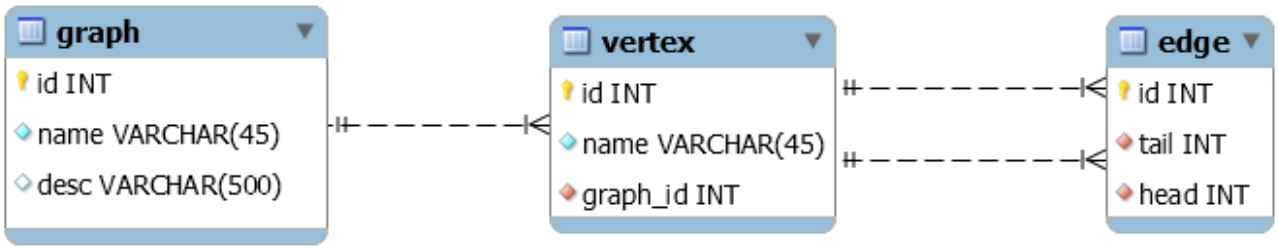
顶点表vertex

字段名	类型	说明
id	int	主键
name	varchar	非空，顶点的名称
g_id	int	外键，描述顶点属于哪一个图

边表edge

字段名	类型	说明
id	int	主键
tail	int	外键，弧尾顶点，顶点在vertex表中必须存在
head	int	外键，弧头顶点，顶点在vertex表中必须存在
g_id	int	外键，描述边属于哪一个图

通过弧尾、弧头顶点来描述有向边



业务设计

“任务”设计

流程定义在表中，"任务"如何描述呢？

方法一
subprocess执行bash脚本script
优点：简单，易行
缺点：要启动外部进程。bash脚本表达能力较弱，难调试

方法二
嵌入其它语言的脚本，例如lua语言
优点：不启动子进程，功能强大。
缺点：技术要求高，需要学习其它脚本语言。

python中执行lua脚本
https://pypi.python.org/pypi/lupa/
lupa目前版本1.6

安装
`$ pip install lupa`
注意安装在windows下出错，原因是要执行`pkg-config`命令，这是Linux命令。

```
import lupa
from lupa import LuaRuntime

import logging
logging.basicConfig(format="%(process)d %(thread)d %(message)s", level=logging.INFO)

lua = LuaRuntime()
print(lua.eval('1+3'))
```

```
def pyfunc(n):
    import socket
    logging.info('hello')
    return socket.gethostname()

luafunc = lua.eval('''
function(f,n)
    return f(n)
end''')
logging.info('main')
print(luafunc(pyfunc, 1))

add = lua.eval('''
function (x,y)
    return x+y
end
''')
print(add(4,5))
```

其实，还可以运行JS脚本。

选择方法一，任务脚本样例如下，存储shell脚本文本就行了

```
"echo magedu"
```

在表vertex中增加一个字段，存储脚本

字段名	类型	说明
id	int	主键
name	varchar	非空，顶点的名称
g_id	int	外键，描述顶点属于哪一个图
script	text	可以为空，存储任务脚本

执行条件

脚本执行之前，可能需要提供一些参数，才能开始执行脚本。
依然在vertex表中增加input字段，定义输入参数的描述。

vertex表

字段名	类型	说明
id	int	主键
name	varchar	非空，顶点的名称
g_id	int	外键，描述顶点属于哪一个图
script	text	可以为空，存储任务脚本
input	text	可以为空，存储json格式的输入参数定义

```
//json定义
{
    "name1":{
        "type":"",
        "required":true
    },
    "name2":{
        "type":"",
        "required":true,
        "default": 1
    }
}
```

name就是参数的名称，后面定义该参数的类型、是否必须提供 等属性。
可以定义多个参数。

作用
进入某个节点的时候，就必须满足条件，提供足够的参数。
如果提供的参数满足要求，就进入节点，否则一直等待到参数满足。
如果满足了，才能去执行script。
input就是一个约束的定义。

“任务”执行

当流程走到某一个顶点的时候，读取任务即脚本，执行这个脚本。

如何执行？
手动执行或自动执行。

1、手动执行
流程走到这个顶点等待用户操作，需要用户手动干预。
例如输入该顶点任务需要的一些配置参数，等待用户输入后才能进行下一步
例如该顶点任务完成后由用户选择下一个执行顶点

2、自动执行
自动填写input，例如使用缺省值，来满足用户为交互式填写的时候自动补全数据。
脚本执行后，自动跳转到下一个顶点。当然这个所谓自动，程序不会智能的选择路径，需要提前指定好，执行完脚本，就可以跳转到下一个顶点了。

“任务”流转设计

当流程走到某一个顶点的时候，读取任务即脚本，或手动流转，或自动流转。
手动流转，需要人工选择下一个顶点，可以提供可视化界面供用户方便的选择。
自动流转，就需要在信息中提供下一个节点的信息，供程序自动完成。

那么，如何区分一个顶点是否是自动执行呢？

如果vertex表中的script字段改为json。
如果next不存在，则不能自动执行，需要手动操作。
如果next存在，则程序自动跳转。

```
{
    "script":"echo magedu"
}

{
    "script":"echo magedu",
    'next':'B'
}
{
    "script":"echo magedu",
    'next':2
}
```

为了方便用户，next可以提供2种类型参数：
int表示使用vertex的id；
str表示使用vertex的name，但是必须是同一个graph id。同一个DAG的定义中，要求顶点的名字不能冲突，所以可以用。

流程结束

如果进入一个节点，执行完脚本，先检测其出度为0，执行完流程就结束了。

如何判断出度为0呢？
在edge表中，使用当前节点的顶点ID作为弧尾t，找不到弧头h的任何记录。

执行引擎设计

pipeline设计

前面设计的仅仅是流程DAG定义，可以认为这是一个模板定义，流程真正执行的时候需要记录执行这个流程的任务流的数据。

创建表pipeline

字段名	类型	说明
id	int	主键
g_id	int	外键，指明使用的是哪一个流程DAG定义
current	int	外键，顶点id，表示当前走到哪一个节点

这个表以后还要添加其它字段，存储一些附加信息，例如谁加入的流程、执行时间等。

一个pipeline应该指定哪一个DAG，并选择DAG的起点。因为DAG可能有多个起点，即入度为0的顶点，需要指定。然后把这些信息记录在pipeline表中，current为起点顶点的id。
提取current顶点的input信息，input为空，直接执行脚本，否则用户输入满足了，才能执行script。
不管是手动流转还是自动流转，如果到了下一个节点，需要修改current字段的值。
任务流执行完毕，修改最后一个节点的状态为完成。

举例
当前节点任务是打包，调用maven命令执行打包，先要提取input，要求用户输入ip地址、输出目录等信息，然后才能执行打包脚本。

历史轨迹设计

pipeline表只能看到有哪些流程正在运行，但是究竟走了DAG中的哪些节点，不清楚，执行节点前有输入了哪些参数也不清楚。
如何查看、回溯当前pipeline的执行轨迹？

track表

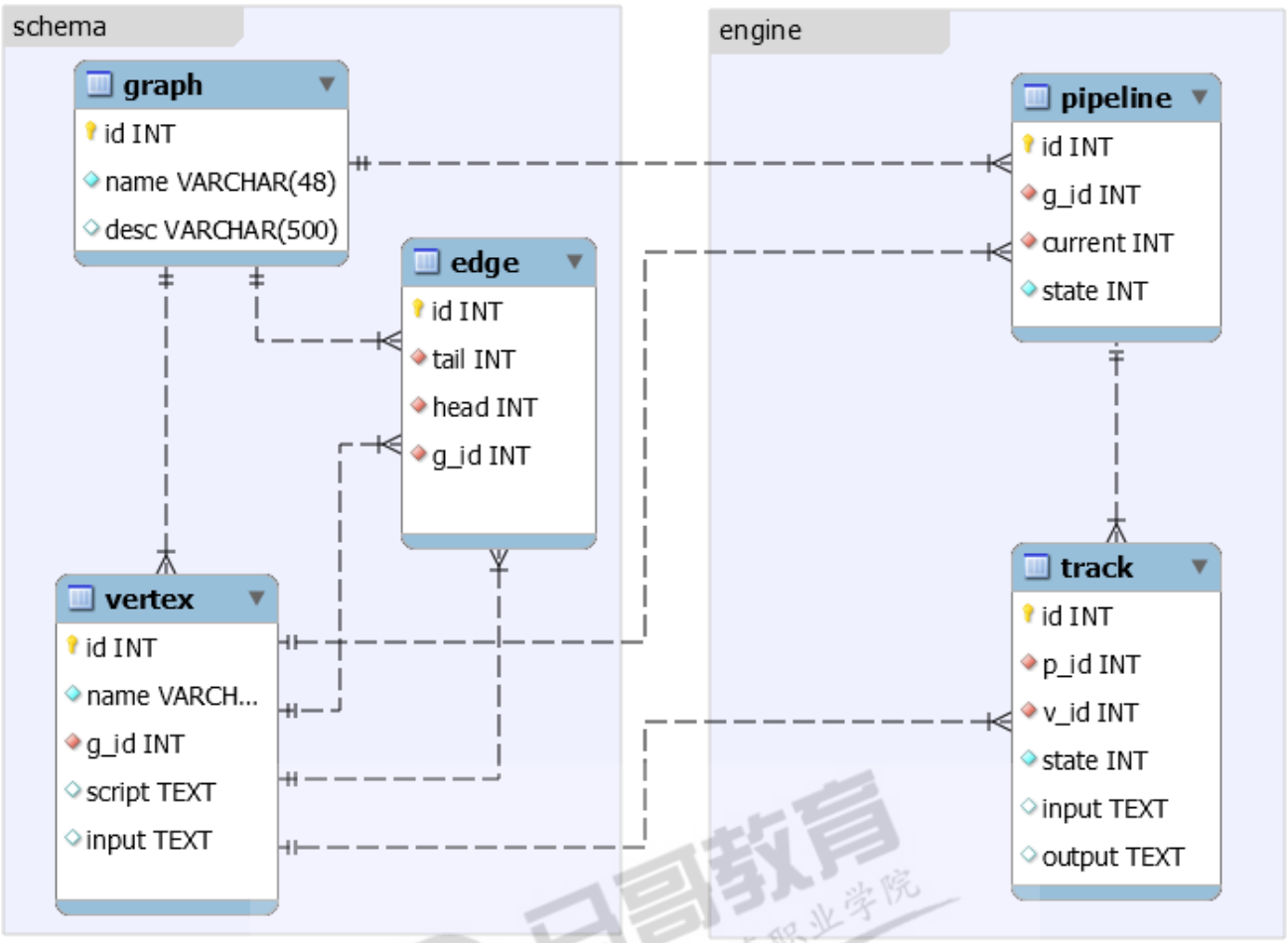
字段名	类型	说明
id	int	主键
p_id	int	外键，哪一个流程的历史
v_id	int	外键，顶点的ID，经过的历史节点
input	text	可以为空，输入的参数值
output	text	可以为空，任务的输出

状态设计

在pipeline表、track表中增加state字段，来描述在某个节点上执行的状态，是等待中，还是正在运行，还是成功或失败，还是执行完毕。
STATE_WAITING = 0
STATE_RUNNING = 1
STATE_SUCCEED = 2
STATE_FAILED = 3
STATE_FINISH = 4

DAG定义，需要graph表、vertex表、edge表。
Pipeline执行，需要pipeline表、track表。

模型



DAG检测

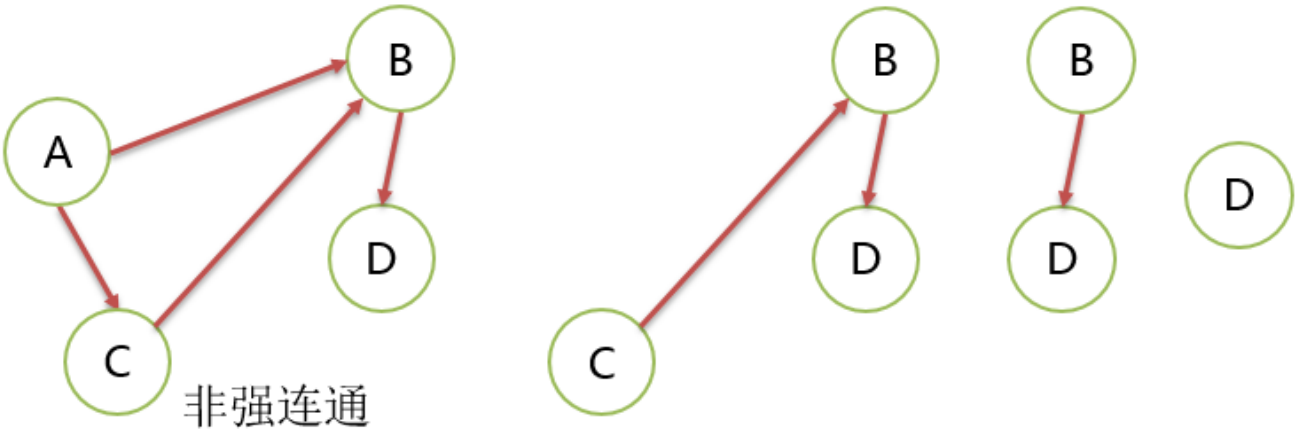
这需要用到图论的知识。

1、DFS算法
DFS（Depth First Search）深度优先遍历，递归算法。
需要改进算法以适用于有向图。
不能直接检测有向图是否有环。

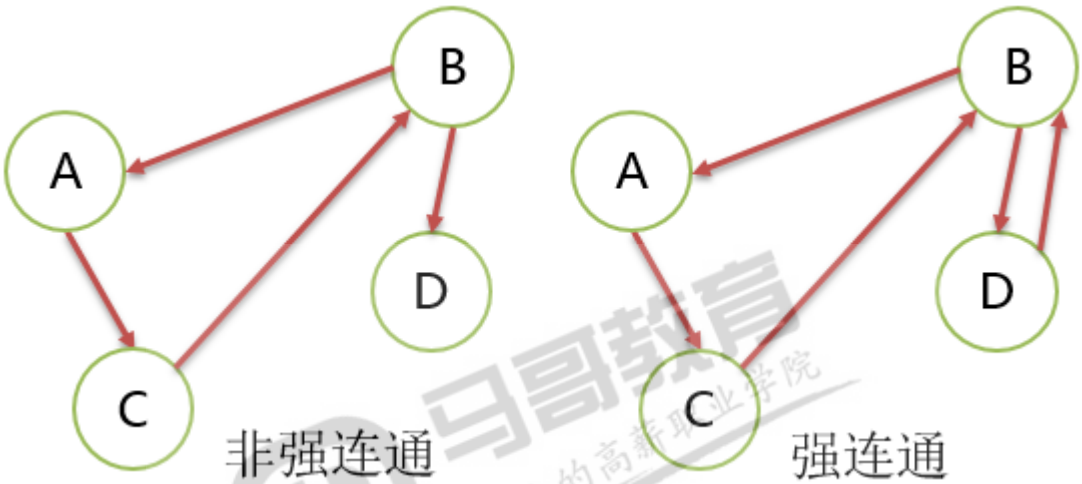
2、拓扑排序算法
拓扑排序就是把有向图中的顶点以线性方式排序，如果有弧`<u,b>`则最后线性排序的结果，顶点u总是在顶点v的前面。
一个有向图能被拓扑排序的充要条件是：它必须是DAG。

kahn算法
（1）选择一个入度为0的顶点并输出它
（2）删除以此顶点为弧尾的弧
重复上面2步，直到输出全部顶点为止，或者图中不存在入度为0的顶点为止。
如果输出了全部顶点，就是DAG

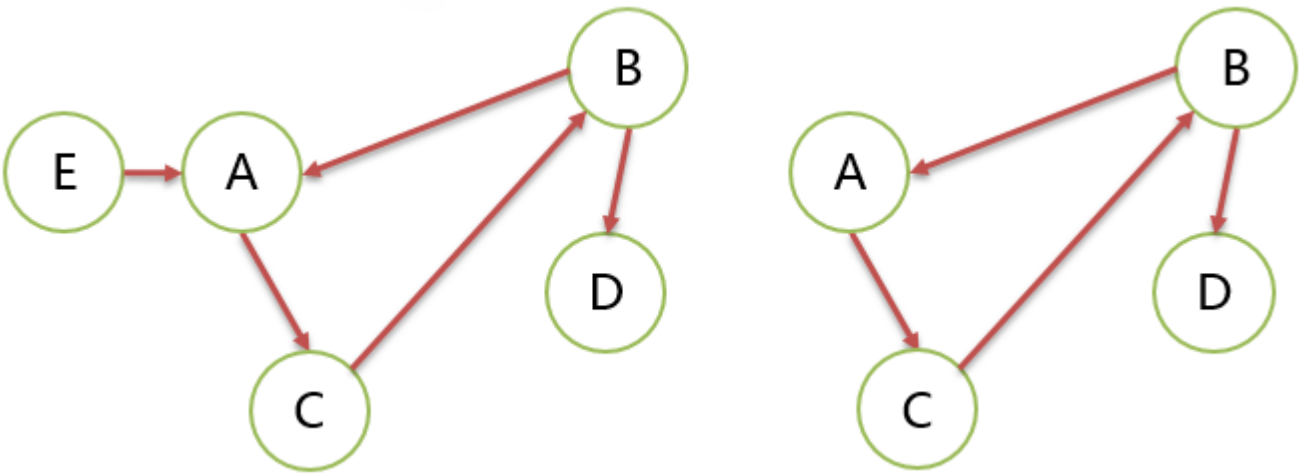
举例



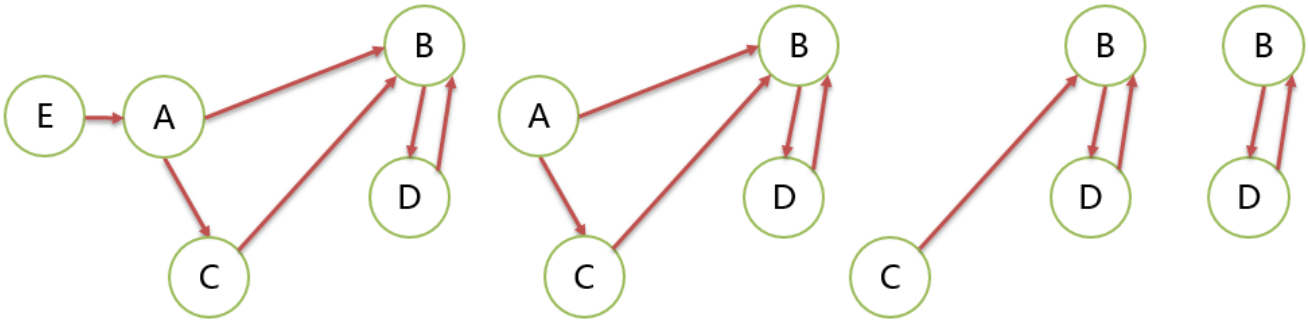
上图可以找到一个入度为0的顶点A，从它开始，可以得到序列ACBD。最后输出了全部顶点。所以这个图是DAG。



上面2个图都不是DAG。左图一个环，右图2个环。
这2个图都找不到入度为0的起始顶点，都不是DAG。



上图虽然可以找到入度为0的顶点，但是移除它和关联的边，剩下顶点找不到入度为0的顶点，它不是DAG。



上图依然不是DAG，B、D之间有环。