**有向图**

在有向图中，边是单向的：每条边所连接的两个顶点都是一个有序对，邻接性是单向的。

**4.2.1 术语**

一幅有方向性的图是由一组顶点和一组有方向的边组成的，每条有方向的边都连接着有序的一对顶点。

我们称一条有向边由第一个顶点指出并指向第二个顶点。

在一幅有向图中，一个顶点的出度为由该顶点指出的边的总数；一个顶点的入度为指向该顶点的边的总数。

一条有向边的第一个顶点称为它的头，第二个顶点称为它的尾。

我们约定，每个顶点都能达到它自己。

**4.2.2 有向图的数据类型**

有向图的API

构造函数：Digraph(int V)

顶点总数：int V()

边总数：int E()

添加一条边：void addEdge(int v,int w)

由v指出的边所连接的所有顶点：Iterable<Integer> adj(int v)

该图的反向图：Digraph reverse()

对象的字符串表示：String tostring()

编程：Digraph数据类型（Digraph）

**4.2.3 有向图中的可达性**

单点可达性：给定一幅有向图和一个起点s，回答“是否存在一条从s到达给定顶点v的有向路径”等类似问题。

有向图的可达性API

在G中找到从s可达的所有顶点：DirectedDFS(Digraph G, int s)

在G中找到从sources中所有顶点可达的所有顶点:

DirectedDFS(Digraph G, Iterable<Integer> sources)

v是可达的吗:boolean marked(int v)

多点可达性：给定一副有向图和顶点的集合，回答“是否存在一条从集合中的任意顶点到达给定顶点v的有向路径”等类似问题。

编程：有向图的可达性（DirectedDFS）

**4.2.4 环和有向无环图**

优先级限制指明了哪些任务必须在哪些任务之前完成。

优先级限制下的调度问题等价于拓扑排序。

**拓扑排序**：给定一幅有向图，将所有的顶点排序，使得所有的有向边均从排在前面的元素指向排在后面的元素。

如果一个有优先级限制的问题中存在有向环，那么这个问题肯定是无解的。

有向环的API

寻找有向环的构造函数：DirectedCycle(Digraph G)

G是否含有有向环：hasCycle()

有向环中的所有顶点：Iterable<Integer> cycle()

编程：寻找有向环（DirectedCycle）

优先级限制下的调度问题等价于计算有向无环图中的所有顶点的拓扑顺序。

拓扑排序API

构造函数：Togological(Digraph G)

G是有向无环图吗：isDAG()

拓扑有序的所有顶点：Iterable<Integer> order()

当且仅当一幅有向图是无环图时它才能进行拓扑排序

在典型的应用中，人们感兴趣的是顶点的以下3种排列顺序。

前序：在递归调用之前将顶点加入队列。

后序：在递归调用之后将顶点加入队列。

逆后序：在递归调用之后将顶点压入栈。

参考文档：计算有向图中顶点的深度优先次序参考图

编程：有向图中基于深度优先搜索的顶点排序（DepthFirstOrder）

编程：拓扑排序（Topological）

**一幅有向无环图的拓扑顺序即为所有顶点的逆后序排列。**

4.2.5 有向图中的强连通性

如果两个顶点v和w是互相可达的，则称它们为强连通的。

如果一幅有向图中的任意两个顶点都是强连通的，则称这幅有向图也是强连通的。

两个顶点是强连通的当且仅当它们都在一个普通的有向环中。

强连通性将所有顶点分为了一些等价类，每个等价类都是由相互均为强连通的顶点的最大子集组成。

我们将这些子集称为强连通分量。

一个强连通图只含有一个强连通分量，而一个有向无环图中则含有V个强连通分量。

强连通分量API

预处理构造函数：SCC(Digraph G)

v和w是强连通的吗：stronglyConnected(int v, int w)

图中的强连通分量的总数：int count()

V所在的强连通分量的标识符：int id(int v)

**编程：计算强连通分量的Kosaraju算法（SCC或者KosarajuSCC）**

1. 在给定的一幅有向图G中，使用DepthFirstOrder来计算它的反向图GR的逆后序排列。
2. 在G中进行标准的深度优先搜索，但是要按照刚才计算得到的顺序而非标准的顺序来访问所有未被标记的顶点。
3. 在构造函数中，所有在同一个递归dfs()调用中被访问到的顶点都在同一个强连通分量中，将它们按照和CC相同的方式识别出来。

使用深度优先搜索查找给定有向图G的反向图GR，根据由此得到的所有顶点的逆后序再次用深度优先搜索处理有向图G，其构造函数中的每一次递归调用所标记的顶点都在同一个强连通分量之中。

强连通性：给定一幅有向图，回答“给定的两个顶点是强连通的吗？这幅有向图中含有多少个强连通分量？”等类似问题。

Kosaraju算法的预处理所需的时间和空间与V+E成正比且支持常数时间的有向图强连通性的查询。

有向图G的传递闭包是由相同的一组顶点组成的另一幅有向图，在传递闭包中存在一条从v指向w的边当且仅当在G中w是从v可达的。

顶点对可达性的API

构造函数TransitiveClosure(Digraph G)

W是从v可达的吗：boolean reachable(int v, int w)

编程：顶点对的可达性（TransitiveClosure）

参考文档：在本节中得到解决的有向图处理问题

单点和多点的可达性：DirectedDFS

单点有向路径：DepthFirstDirectedPaths

单点最短有向路径：BreadthFirstDirectedPaths

有向环检测：DirectedCycle

深度优先的顶点排序：DepthFirstOrder

优先级限制下的调度问题（拓扑排序）：Topological

强连通性：KosarajuSCC

顶点对的可达性：TransitiveClosure