**实验十**

**实验目的与要求**：

1. 掌握使用图的深度优先搜索算法实现对有向图中是否包含环的判断；
2. 掌握使用图的深度优先搜索算法实现对有向图的强连通分量的划分。

**1．有向图中环的判断问题**

**【问题描述】**

**给定一个有向图，要求使用深度优先搜索策略，判断图中是否存在环。**

**2．有向图的强连通分量问题**

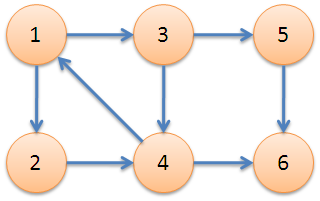
**【问题描述】**

**给定一个有向图，设计一个算法，求解并输出该图的各个强连通分量。**

求有向图的强连通分量(SCC):Tarjan算法

1.在有向图G中，如果两个顶点间至少存在一条路径，称两个顶点强连通(strongly connected)。如果有向图G的每两个顶点都强连通，称G是一个强连通图。非强连通图有向图的极大强连通子图，称为强连通分量(strongly connected component)。

2.下图中，子图{1,2,3,4}为一个强连通分量，因为顶点1,2,3,4两两可达。{5},{6}也分别是两个强连通分量。



3．Tarjan算法是基于对图深度优先搜索的算法，每个强连通分量为搜索树中的一棵子树。搜索时，把当前搜索树中未处理的节点加入一个堆栈，回溯时可以判断栈顶到栈中的节点是否为一个强连通分量。

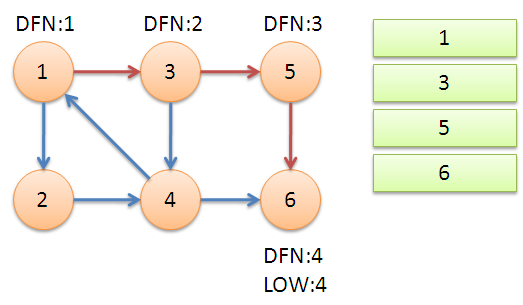
定义几个关键数组:

int DFN[MAX]; //记录节点u第一次被访问时的步数

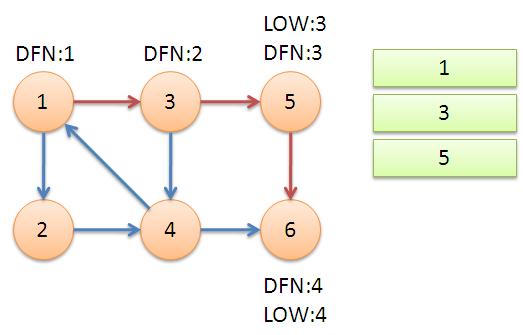
int LOW[MAX]; //记录与节点u和u的子树节点中最早的步数

接下来是对算法流程的演示。

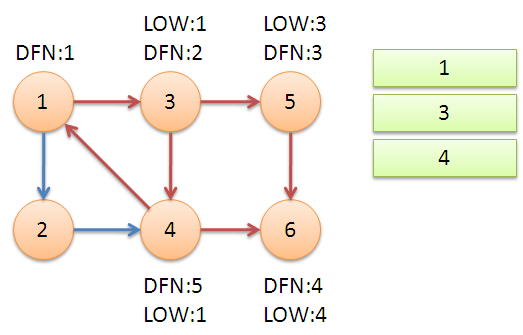
从节点1开始DFS，把遍历到的节点加入栈中。搜索到节点u=6时，DFN[6]=LOW[6]，找到了一个强连通分量。退栈到u=v为止，{6}为一个强连通分量。



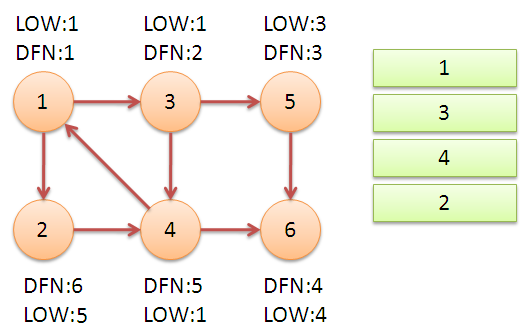
返回节点5，发现DFN[5]=LOW[5]，退栈后{5}为一个强连通分量。



返回节点3，继续搜索到节点4，把4加入堆栈。发现节点4向节点1有后向边，节点1还在栈中，所以LOW[4]=1。节点6已经出栈，(4,6)是横叉边，返回3，(3,4)为树枝边，所以LOW[3]=LOW[4]=1。



继续回到节点1，最后访问节点2。访问边(2,4)，4还在栈中，所以LOW[2]=DFN[4]=5。返回1后，发现DFN[1]=LOW[1]，把栈中节点全部取出，组成一个连通分量{1,3,4,2}。



至此，算法结束。经过该算法，求出了图中全部的三个强连通分量{1,3,4,2},{5},{6}。

**分析:**

运行Tarjan算法的过程中，每个顶点都被访问了一次，且只进出了一次堆栈，每条边也只被访问了一次，所以该算法的时间复杂度为。