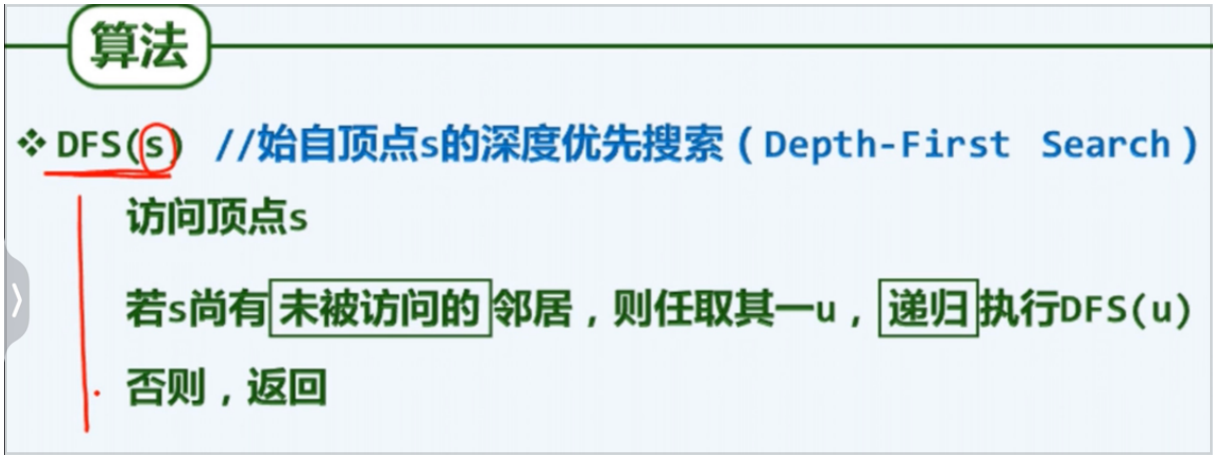
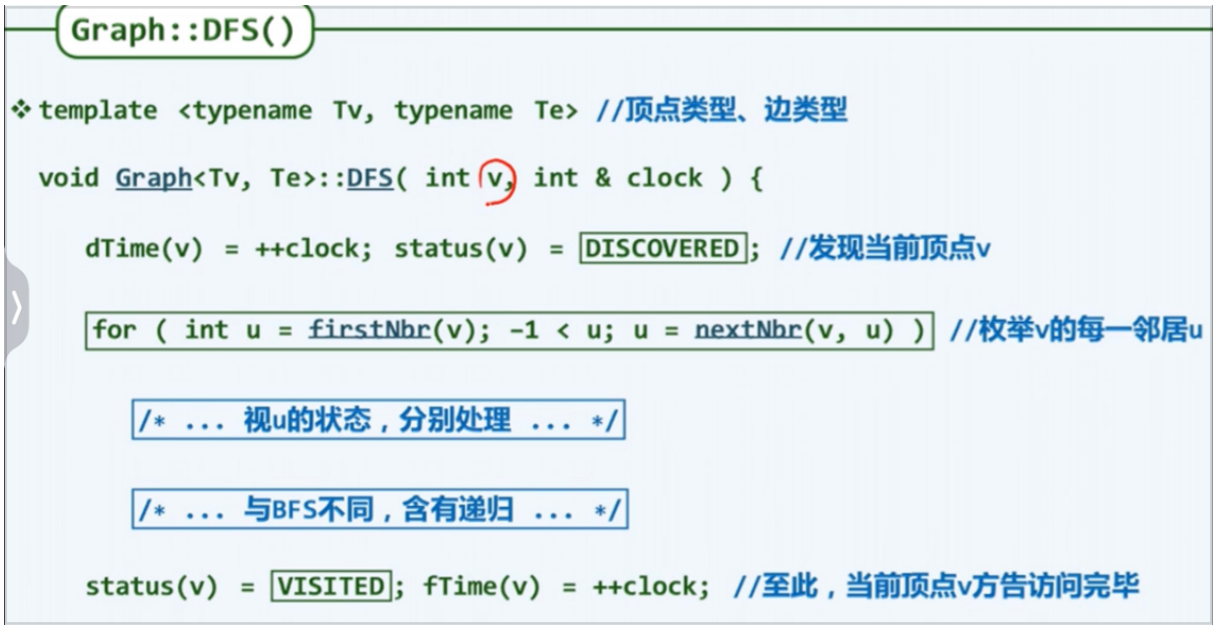
# 06E 深度优先搜索 DFS

#数据结构邓神

## 算法




## 框架

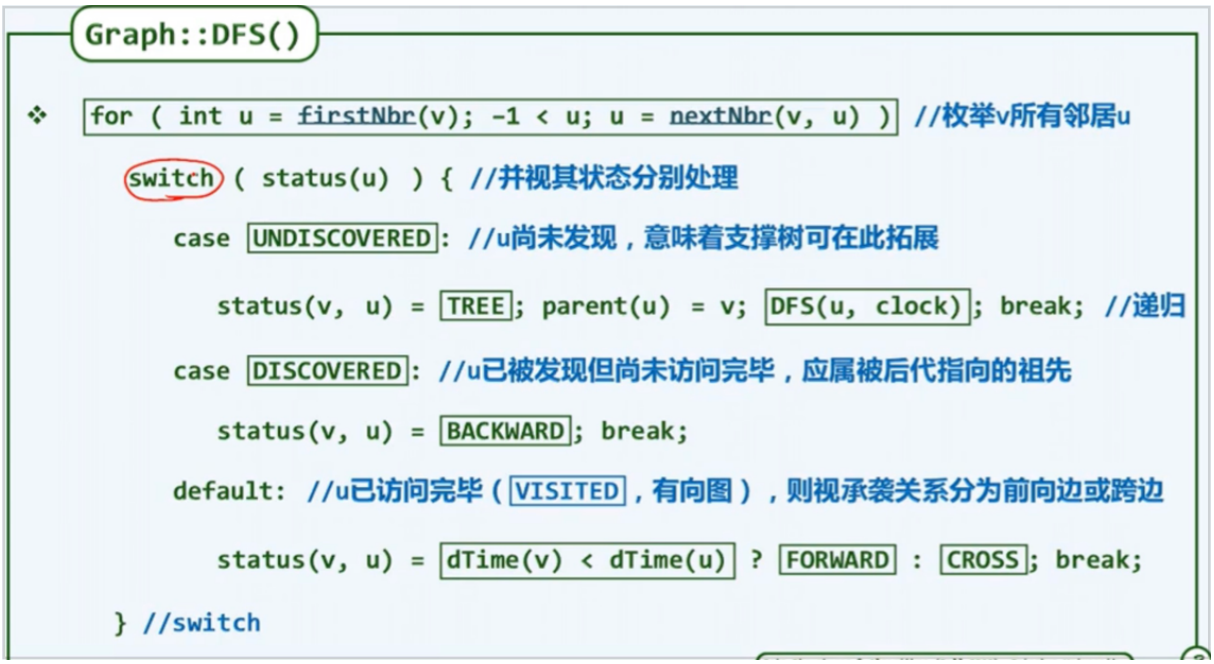



```
template <typename Tv,typename Te> void Graph<Tv,Te>::DFS(int s,int clock){
    dTime(v) = ++clock;
    status(v) = DISCOVERED;
    for (int u = firstNbr(v); -1 <u < ; u = nextNbr(v,u)) {
        /** 看 u 状态处理 **/

        /** 与BFS不同这里可能有递归 **/
    }
    status(v) = VISITED;
    fTime = ++clock;
```

```
}
```

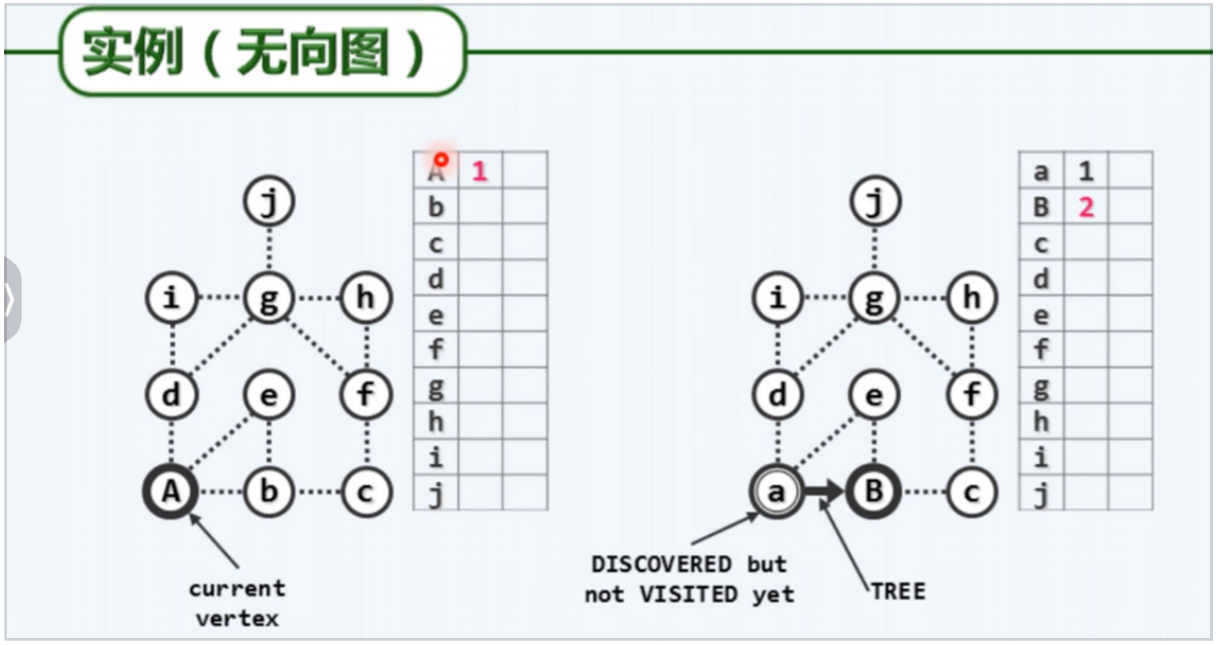
## 细节




```
template <typename Tv,typename Te> void Graph<Tv,Te>::DFS(int s,int clock){
    dTime(v) = ++clock;
    status(v) = DISCOVERED;
    for (int u = firstNbr(v); -1 <u < ; u = nextNbr(v,u)) {
        switch(status(u)){
            case UNDISCOVERD:{
                status(v,u) =TREE;
                parent(u) = v;
                DFS(u,clock);
                break;
            }
            case DISCOVERED:{
                status(v,u) = BACKWARD;
                break;
            }
            default:{
                status(v,u) = dTime(v) < dTime(u) ? FORWARD : CROSS;
                break;
            }
        }
    }
```

```
    status(v) = VISITED;
    fTime = ++clock;
}
```

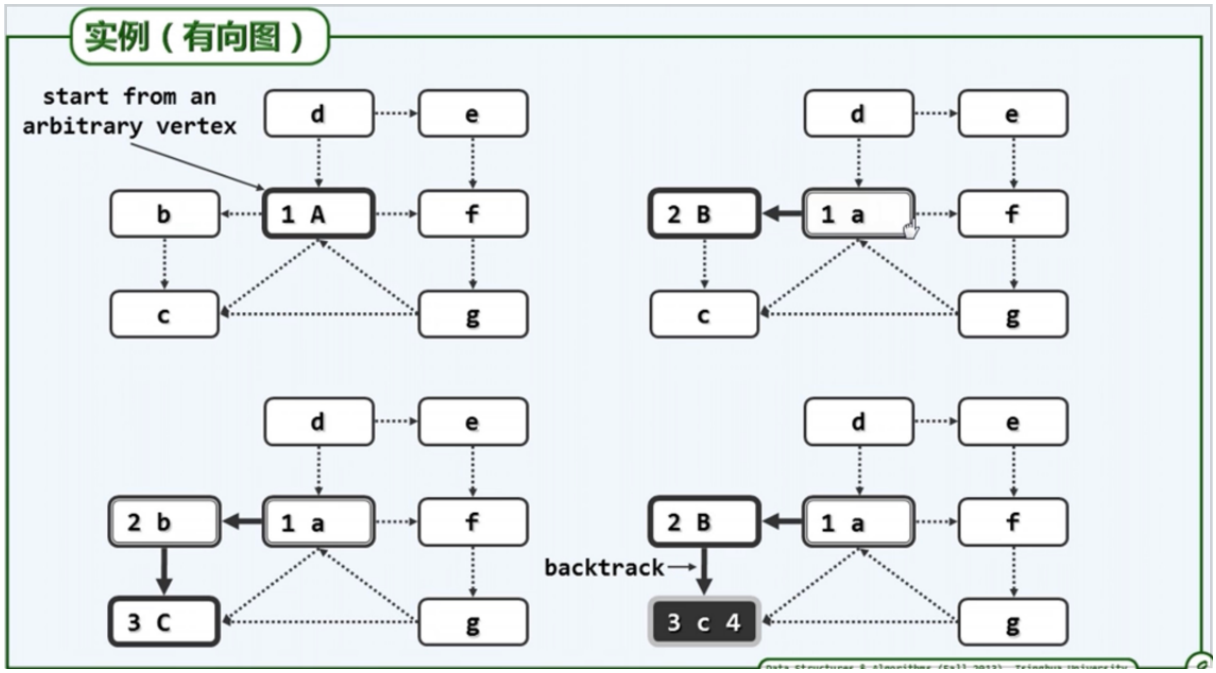
## 无向图




分别为标识符，dTime和fTIme

但是这个递归方法如果图很大会很多的递归层数，那么能否用迭代解决呢?

## 有向图



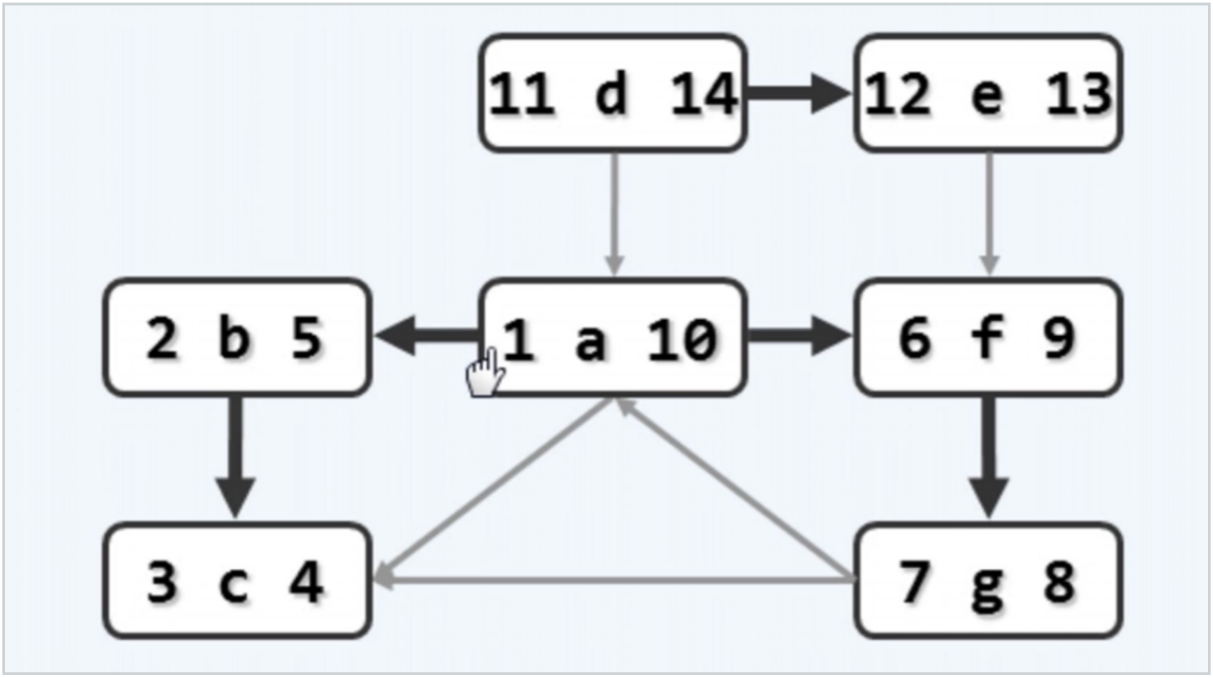

从a出发是无法达到d,e的，a的可达区域为 abcfg

## 多可达域

我们可以类似于广度优先搜索的算法

在包装一层循环来枚举图中所有的节点，保证完成所有的可达域

## 遍历成果




大黑边：构成了遍历森林 是最为重要的
对于所有小灰边 进行了仔细的分类

**其中时间标签的作用是非常巨大的！！！**

## 嵌套引理

顶点的活动期: active[u] = (dTime[u],fTime(u))
只有存在包含和被包含的关系才存在血缘关系
祖先的活跃期一定包含后代的
后代的活跃期一定被祖先的活跃期包含。


上面的图可以很直观的表现

这样的我们就可以在（很小的常系数 只需要两个比较）O(1)的时间下确定是否有直系的血缘关系