

6. 图

拓扑排序

零出度算法

邓俊辉

deng@tsinghua.edu.cn

算法B：逆序输出零出度顶点

❖ /* 基于DFS，借助栈S */

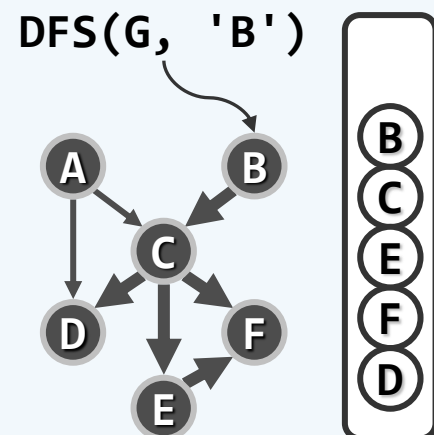
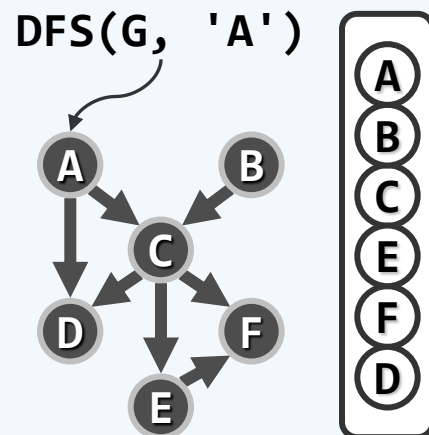
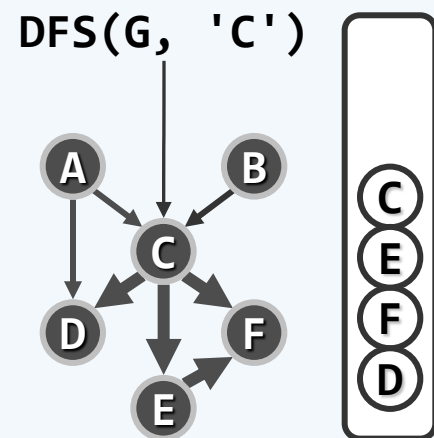
对图G做DFS，其间 //得到组成DFS森林的一系列DFS树

每当有顶点被标记为VISITED，则将其压入S

一旦发现有后向边，则报告非DAG并退出

DFS结束后，顺序弹出S中的各个顶点

❖ 复杂度与DFS相当，也是 $O(n + e)$



实现 (1/2)

❖ template <typename Tv, typename Te> //顶点类型、边类型

```
bool Graph<Tv, Te>::TSort(int v, int & clock, Stack<Tv>* S) {
```

```
    dTime(v) = ++clock; status(v) = DISCOVERED; //发现顶点v
```

```
    for ( int u = firstNbr(v); -1 < u; u = nextNbr(v, u) ) //枚举v所有邻居u
```

```
        /* ... 视u的状态, 分别处理 ... */
```

```
    status(v) = VISITED; S->push( vertex(v) ); //顶点被标记为VISITED时入栈
```

```
    return true;
```

```
}
```

实现 (2/2)

```
❖ for ( int u = firstNbr(v); -1 < u; u = nextNbr(v, u) ) //枚举v所有邻居u
    switch ( status(u) ) { //并视u的状态分别处理
        case UNDISCOVERED:
            parent(u) = v; type(v, u) = TREE; //树边(v, u)
            if ( ! TSort(u, clock, S) ) return false; break; //从顶点u处深入
        case DISCOVERED: //一旦发现后向边 ( 非DAG )
            type(v, u) = BACKWARD; return false; //则退出而不再深入
        default: //VISITED (digraphs only)
            type(v, u) = dTime(v) < dTime(u) ? FORWARD : CROSS; break;
    }
```