研究生算法课课堂笔记

上课日期：2016.11.24　　第(1)节课

组长学号及姓名：迟敬泽 1601214506
组员学号及姓名：綦金玮 1601214516
袁明宽 1601111314

内容概要：

1. 拓扑排序
2. 蝴蝶分类问题
3. 贪心算法

详细内容：

1. 拓扑排序

对一个有向无环图(Directed Acyclic Graph 简称 DAG)*G* 进行拓扑排序，是将 *G* 中所有顶点排成一个线性序列，使得图中任意一对顶点 *u* 和 *v*，若边(*u*, *v*) ∈*E*(*G*)，则 *u* 在线性序列中出现在 *v* 之前。

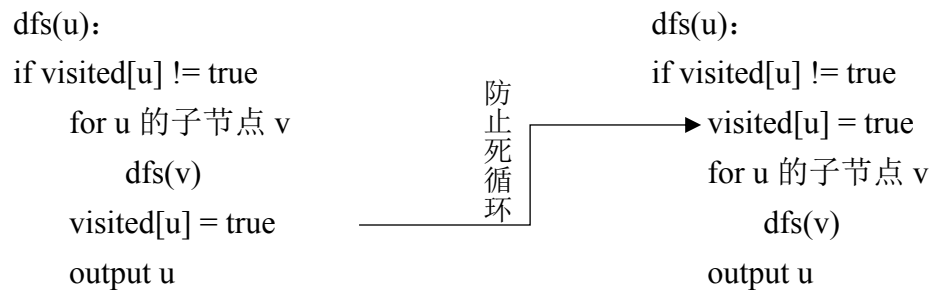
计算 *G* 的一个拓扑排序的两种方法：

1) 无前趋的顶点优先的拓扑排序方法，即该方法的每一步总是输出当前入度为 0 的顶点，具体流程如下：
   a) 维护一个入度为 0 的顶点的集合
   b) 每次从该集合中取出(没有特殊的取出规则，随机取出也行，使用队列/栈也行，下同)一个顶点，将该顶点放入保存结果的 List 中
   c) 循环遍历由该顶点引出的所有边，从图中移除这条边，同时获取该边的另外一个顶点，如果该顶点的入度在减去本条边之后为 0，那么也将这个顶点放到入度为 0 的集合中。然后返回步骤 b)
   d) 当集合为空之后，检查图中是否还存在任何边，如果存在的话，说明图中至少存在一条环路。不存在的话则返回结果 List，此 List 中的顺序就是对图进行拓扑排序的结果。

2) 利用深度优先遍历（DFS）对 DAG 拓扑排序

当从某顶点 *v* 出发的 DFS 搜索完成时，*v* 的所有后继必定均已被访问过，此时的 *v* 相当于是无后继的顶点，因此在 DFS 算法返回之前输出顶点 *v* 即可得到 DAG 的**逆**拓扑序列。

问题代码：



存在错误：

上述伪代码可以避免死循环的发生，但是如果有环也会输出一个序列，这个序列在某种意义上可以看做是最近似的拓扑排序。这样我们就无法判断该图是否有环。我们希望可以通过代码对是否有环的情况进行判断。因此，需要进行环检测。

证明：

对图G进行DFS发现存在子孙节点指回祖先节点的边是图中存在环的充分必要条件。

充分性：DFS 过程中，在祖先结点到子孙结点之间显然存在一条通路，若还存在子孙结点到祖先结点的边，则说明图中存在环。

必要性：因为若存在环，无论从环中的哪个结点出发，该环上所有其他结点都将是开始结点的子孙结点。

修正算法：

在对某个结点 v 进行 DFS 遍历的过程中，如果遍历到一个结点 u，存在边（u, v），则说明存在子孙节点指回父节点的边，即图中存在环。

伪代码：

```
// 结点三种颜色标记：
// 白色 0-结点没有被访问过
// 灰色 1-开始 DFS 但还没有完成
// 黑色 2-已经遍历完
init:所有结点标为白色
dfs(u)：
        if color[u] == 0
            color[u] = 1
            for u 的子节点 v
                dfs(v)
            color[u] = 2
            output u
        else if color[u] == 1
            有环
```

```
        end
    else
        continue
```

2. **蝴蝶分类问题** (算法设计（中文版）第三章练习题 4，p78)

题目：有 *n* 只蝴蝶，每一只都属于两个不同种类中的一种，称为 A 和 B。这里有 *m* 个判断，每个判断是对于一对标本 *i* 和 *j* 是否属于同一个种类的判断，即 *i* 和 *j* 属于同一类或是不同类。需要判断这 *m* 个判断是否一致。

解决方法：1.并查集（本次不涉及） 2. 二部图改进 3. 遍历染色

算法流程：

a) 对所有结点和边建图
b) 遍历所有结点，同类边两端染相同颜色，异类边两端染不同颜色
c) 如果出现染色冲突时，则不一致
d) 遍历完所有结点未出现染色冲突则一致

伪代码：

```
//0 为未染色，1 为一类，-1 为另一类
init：所有结点颜色标为 0
dfs(u)：
    if color[u] == 0
        color[u] = 1
        for u 的子节点 v
            if color[v] == 0
                if e(u,v)是异类边
                    color[v] = -color[u]
                else
                    color[v] = color[u]
                dfs(v)
            else
                if e(u,v)是异类边
                    if(color[v] == color[u])
                        不一致
                        end
                else
                    if(color[v] != color[u])
                        不一致
                        end
```

问题 1：是否可以在从输入数据中读取边表时增量进行染色？

不能，因为这样不能保证某个类别用了固定的颜色，无法判断是否真的染色冲突了。而建图之后再遍历则不会。

问题 2：把同类结点都合并，剩下如果有自环，说明不一致。这种方法是否可行？

这种方法考虑情况不全。如果同类边相连的顶点合并后，剩下的异类边出现自环，说明不一致。但是如果三个结点两两之间都用异类边相连，那么同类结点合并后结构不变（因为没有同类的边），未出现自环但也是属于不合题意的情况。

此方法将同类结点合并后，正确做法是剩下的点和边还要判断是否为二部图，过于复杂。

3. 贪心算法

收银员算法：

算法的目标：现有一堆各种面值的硬币，你需要用最少数目的硬币凑出目标金额。

算法的内容：先找面值最大且不超出目标金额的硬币，然后总金额减去这个硬币面值再进行迭代。

这是美国收银员找钱的方法。是一种贪心算法。

但收银员算法不是对所有面值都是最优的。例如目标金额为 140，硬币面值为 100、90、70、50、20、10。这种情况下收银员算法得到的解是 100+20+20，但最优解是 70+70。

而且收银员算法可能找不到可行解。例如目标金额是 15，硬币面值为 7、8、9。这种情况下最优解是 7+8，但直接使用收银员算法会先选择面值为 9 的硬币，从而找不到可行解。

问题：什么情况下贪心法是最优的？

充分条件：当大面值能被次大面值组成得到时，贪心一定最优。