Problem2

算法 1

题目要求：对于 G 中的任两个顶点，检查它们之间是否有路径存在。如有，输出其中一条路径。如无，返回 false。

程序设计思路

总体上：使用深度优先遍历来实现，从起点开始深度优先遍历，在遍历中遇到终点后便返回 true，结束。如果遍历了所有路径都没遇到终点，返回 false。

具体实现思路：公有的 checkroute 函数检查起点终点是否存在，并将其转化为内部表示序号，然后调用私有的 checkroute1 函数进行深度优先遍历，检查是否有路径，如果路径存在则改变 visited 数组相对应存储数值，不存在，则设置相应存储数值为特定值-1 进行识别。最后通过数组 visited 将路径进行输出（输出在 main 函数进行实现）

代码实现：

```
bool adjListGraph::checkroute(int x, int y,int visited[],int N) {
    int startNo,endNo;

    for(int i=0;i<ver_s;++startNo)
        if(verList[startNo].ver==x)break;
    if(startNo==ver_s)return false;

    for(endNo=0;endNo<ver_s;++endNo)
        if(verList[endNo].ver==y)break;
    if(endNo==ver_s)return false;

    return checkroute1(startNo,endNo,visited);
}
bool adjListGraph::checkroute1(int start, int end, int *visited) {
    edgeNode *p=verList[start].head;
    bool flag;
    visited[start]=count;                    //为存储相应的路径而设立的数组
    count++;                                 //全局变量进行控制顺序
    while (p!=NULL){
        if(p->end==end)return true;
        if(!visited[p->end]){
            flag=checkroute1(p->end,end,visited);
            if(flag)return true;
            else {
                --visited[p->end];
                --count;
            }
        }
```

```
            p=p->next;
        }
        return false;
}
```

时间复杂度分析：判断是否存在复杂度 O(N),深度优先搜索，时间复杂度为 $O(N^2)$其中顶点数目为 N，所以总的算法时间复杂度为 $O(N)+ O(N^2)= O(N^2)$.

算法 2

题目要求：用深度优先搜索实现拓扑排序, 判断该图是否为无环图, 是则给出拓扑排序结果, 否则返回 loop。

程序设计思路：

分为两部分：判断是否为无环图、使用深度优先搜索实现拓扑排序。

判断是否为无环图：可以从这个图的每个节点出发进行一次一笔画，如果在一笔画的过程中遇到了已经访问过的节点，则表示已经出现了环。（函数名称为 isDAG，并分为私有和公有函数）

深度优先搜索实现拓扑排序：无后继节点优先法，每一步输出的节点是无后继的节点。当从某节点 v 出发的 DFS 搜索完成时，v 的所有后继节点必定已经被访问过，此时的 v 相当于是无后继节点，在算法返回之前将节点 v 保存在事先准备好的数组里。（函数名称为 topSort，分为私有和共有函数）

代码实现：

```
bool adjListGraph::isDAG() {                              //判断是否无环图
        bool *visited=new bool[ver_s];
        bool flag;

        for(int i=0;i<ver_s;++i){
            flag=isDAG(i,visited);
            if(!flag)return false;
            visited[i]=false;
        }
        return true;
}

bool adjListGraph::isDAG(int start, bool *visited) {
        edgeNode *p=verList[start].head;
        bool flag;

        visited[start]= true;
        while(p!=NULL){
            if(visited[p->end])return false;
            else flag=isDAG(p->end,visited);
```

```
            if(!flag)return false;
            visited[p->end]= false;
            p=p->next;
        }
        return true;
}

void adjListGraph::topSort(int *Storge) {                    //公有函数，进行拓扑排序
        bool *visited = new bool[ver_s];
        for (int i = 0; i < ver_s; ++i)visited[i] = false;

        for (int i = 0; i < ver_s; ++i) {
            if (visited[i])continue;
            topSort(i, visited, Storge);
        }
}

void adjListGraph::topSort(int start, bool *visited,int *Storge) {
        edgeNode *p=verList[start].head;                     // 私有函数，进行深度优先搜索
        visited[start]=true;
        while(p!=NULL){
            if(!visited[p->end])
                topSort(p->end,visited,Storge);
            p=p->next;

        }
        Storge[count1]=verList[start].ver;
        ++count1;
}
```

时间复杂度分析：

判断是否为无环图，最高时间为 $O(N)* O(N-1)= O(N^2)$

使用深度优先搜索实现拓扑排序，时间复杂度为 $O(N^2)$

所以总的时间复杂度为 $O(N^2)+ O(N^2)= O(N^2)$