算法 1：循环输出并删除入度为 0 的节点。

伪代码

```
1   队列 Q = {图中所有入度为 0 的节点}
2   While(Q 非空)
3       从 Q 中取出第一个节点 v 并输出
4       对于节点 v 的每条出边(v,v_i)，把节点 v_i 的入度减 1
5       If v_i 的入度为 0
6           将 v_i 加入队列 Q 中
7       End
8   End
```

Algorithm 1

```
1   /* Topological Sort
2    * Input:
3    *    lnklist:邻接链表
4    * Output:
5    *    拓扑顺序
6    * 时间复杂度: o(m+n)
7    */
8   vector<int> topsort(const vector<vector<int> > &lnklist)
9   {
10      int n = lnklist.size();
11      vector<int> d(n);//入度
12      //遍历计算图中每个顶点的入度
13      //时间复杂度 o(m+n)
14      for(int i = 0; i < n; i++)
15      {
16          for(int j = 0; j < lnklist[i].size; j++)
17          {
18              int  v = lnklist[i][j];//第 i 个顶点的第 j 条边指向的顶点
19              d[v]++;
20          }
21      }
22      //队列 q 记录入度为 0 的节点
23      //时间复杂度 o(n)
24      queue<int> q;
25      for(int i = 0; i < n; i++)
26      {
27          if(!d[i]) q.push(i);
28      }
29
30      vector<int> result;
31      //循环输出队列 q 中入度为 0 的节点 u，并更新节点 lnklist[u][i]的入度
```

```
32      //对所有的节点一次遍历，对每个节点的所有边一次遍历
33      //时间复杂度 o(m+n)
34      while(!q.empty())
35      {
36          int u = q.front();
37          q.pop();
38          result.push_back(u);
39          for(int i = 0; i < lnklist[u].size; i++)
40          {
41              int v = lnklist[u][i];
42              d[v]--;
43              if(!d[v]) q.push(v);
44          }
45      }
46      return result;
47  }
```