学生需要选修一定数目的课程才能毕业，这些课程之间有先导关系。假设所有的课程在每学期都能提供学生选修，学生每学期可以选修的课程数不限。给出一些课程以及课程之间的关系，安排一个计划，用最少的学期数修完所有的课程。

【解】这还是一个拓扑排序问题，在拓扑排序时按批输出满足条件的结点，即入度为 0 的结点。每一批就是一个学期可以选修的课程。这可以对拓扑排序函数稍加修改。在代码清单 12-25 中，我们设置了两个队列 q1 和 q2，一个队列保存一个学期的课。所有没有先导课程的课在第一学期都可以选，所以先将入度为 0 的所有结点进入队列 q1，这是第一学期可以选修的课程。然后将 q1 中的元素依次出队，将它们的后继的入度减 1。如果后继结点的入度减到了 0，则让它进入队列 q2，表示是第二学期可以选修的课。然后再将 q2 中的元素依次出栈，将它们的后继的入度减 1。如果后继结点的入度减到了 0，则让它进入队列 q1，表示是第三学期可以选修的课。重复上述过程，直到所有的课程都已被选修。

代码清单 12-25 程序设计题 6 的解

```
1.  template <class TypeOfVer, class TypeOfEdge>
2.  void adjListGraph<TypeOfVer, TypeOfEdge>::findLessTime( ) const
3.  { linkQueue<int> q1, q2;
4.    edgeNode *p;
5.    int current, semester = 1;
6.    int *inDegree = new int[Vers];
7.
8.    for (int i = 0; i < Vers; ++i) inDegree[i] = 0;
9.    for ( i = 0; i < Vers; ++i)                    // 计算结点的入度
10.         for (p = verList[i].head; p != NULL; p = p->next)
11.              ++inDegree[p->end];
12.
13.   for (i = 0; i < Vers; ++i)              // 将入度为 0 的结点放入 q1
14.         if (inDegree[i] == 0) q1.enQueue(i);
15.
16.   cout << "课程安排为：" << endl;
17.   while(true) {
18.         cout << "\n 第" << semester << "学期的课为：";
19.     while( !q1.isEmpty( ) ) {                  // 输出一学期的课
20.           current = q1.deQueue( );
21.           cout << verList[current].ver << '\t';
22.           for (p = verList[current].head; p != NULL; p = p->next)
23.            if( --inDegree[p->end] == 0 ) q2.enQueue( p->end );
24.         }
25.         if (q2.isEmpty()) break;                  // 所有课程都已输出
26.         ++semester;
27.         cout << "\n 第" << semester << "学期的课为：";
28.     while( !q2.isEmpty( ) ) {                  // 输出一学期的课
29.           current = q2.deQueue( );
30.           cout << verList[current].ver << '\t';
```

```
31.              for (p = verList[current].head; p != NULL; p = p->next)
32.               if( --inDegree[p->end] == 0 ) q1.enQueue( p->end );
33.          }
34.          if (q1.isEmpty()) break;                    // 所有课程都已输出
35.          ++semester;
36.    }
37.    cout << endl;
38. }
```