《数据结构与算法》课程线上教学设计

一、基本信息

课程名称：《数据结构与算法》

课程类型：☐通识课　☐学科基础课　☑专业核心课
　　☐专业方向课　☐选修课

开课年级：2019 级

面向专业：软件工程、智能科学与大数据技术

教学章节：第 8 章　快速排序法

授课学时：1 学时

主讲教师：程菲

授课形式：线上讲解+提问+课堂练习

选用平台及课程链接：

博思智慧学习平台：http://aiit.iflysse.com/Login_aiit.aspx

二、案例背景

1.课程性质

《数据结构与算法》共 32 理论课时+16 实验课时，为软件工程、智能科学与大数据技术专业的必修核心课程，开课学期为 1-2。

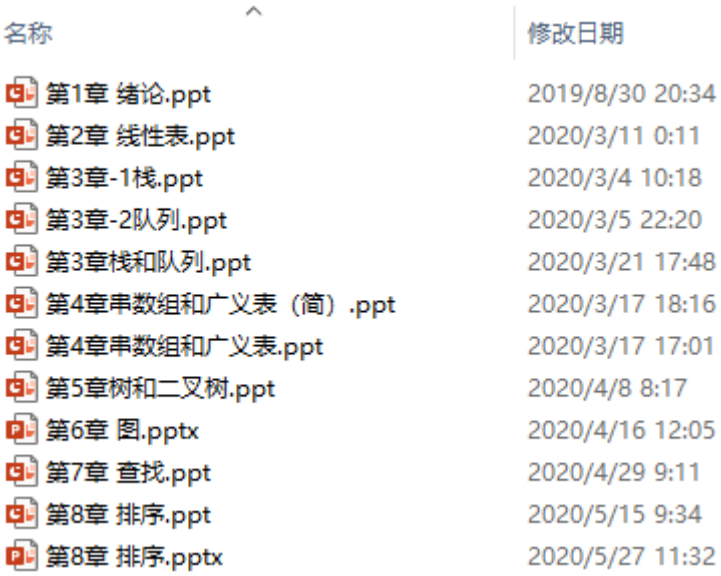
2.课程考核

检验课程目标达成度，评价学生学习成果达成度。考核方式：过程考核（含出勤、平台学习态度、实验等）+期末考试，具体考核标准如下：

考核环节	建议分值比例	考核/评价范围与标准	建议考核形式
平时成绩	60	1. 根据出勤情况打分，如请假、旷课、迟到早退等。 2. 根据线上学习表现打分，如回答问题、参与讨论等。 3. 根据作业情况打分，如按时、按量完成、正确率等。	平时打分
期末成绩	40	1.学科统一组织闭卷考试，卷面成绩 100 分，按比例计入课程总评成绩。 2.综合考察学生对知识的掌握程度以及分析问题、解决问题的能力。 3.考试题型为选择题、判断题、简答题、算法设计题。	线上考试

3.教学内容体系

本课程涉及常用的几种数据结构：线性表、栈和队列、串和数组、广义表、数和二叉树、图，排序算法（内部排序算法、外部排序算法）、查找算法等。对应于计算机学科中问题求解的理论、抽象和设计的方法论，在软件工程课程体系有着承上启下的核心地位。它一方面扩展和深化在离散数学、程序设计语言等课程学到的基本技术和方法，另一方面为进一步学习其他专业课，如操作系统、数据库系统、软件工程等奠定坚实的理论与实践基础。本课程重在培养学生的基本数据结构分析能力和综合程序设计实现能力。具体教学内容有：



名称	修改日期
第1章 绪论.ppt	2019/8/30 20:34
第2章 线性表.ppt	2020/3/11 0:11
第3章-1栈.ppt	2020/3/4 10:18
第3章-2队列.ppt	2020/3/5 22:20
第3章栈和队列.ppt	2020/3/21 17:48
第4章串数组和广义表（简）.ppt	2020/3/17 18:16
第4章串数组和广义表.ppt	2020/3/17 17:01
第5章树和二叉树.ppt	2020/4/8 8:17
第6章 图.pptx	2020/4/16 12:05
第7章 查找.ppt	2020/4/29 9:11
第8章 排序.ppt	2020/5/15 9:34
第8章 排序.pptx	2020/5/27 11:32

4.学生特点及教学条件

在开课前进行学情分析，班级具体情况如下：

（1）教学环境要求：

- 要求学生 PC 端安装 visual studio 编程工具，对电脑配置不作特殊要求
- 要求学生在手机上安装 i 博思 App
- 要求学生 PC 端安装 Google Chrome 浏览器，以便完成作业与考试
- 要求学生 PC 端安装 Microsoft Office
- 要求学习安装腾讯会议客户端
- 教学平台：博思网络教学平台+腾讯会议，其中博思平台完成理论课程教学，腾讯会议进行讨论，PC 端完成实验；学生使用手机或 PC 参加学习
- 课件演示采用 PPT 形式，参考书采用 PDF 电子版教材，编程工具为 VS 或 DEV C++，flash 动画使用 IE 进行播放

（2）学生学习状态数据统计：

- 本课程学习者为来自大数据与人工智能及软件工程专业学生，已修完“C 语言编程”先学课程
- 3 个班级学生共 54+50+50=154 位学生，全部学生均可正常进行线上学习

- 100%学生共同目标通过课程考核

三、案例设计思路

1.章节教学内容：本案例属于教材第 8 章内容。本章教学内容包括：(1) 排序的基本概念，包括正序，逆序，稳定性，排序方法的分类；(2) 插入排序：直接插入排序、折半插入排序；(3) 交换排序：冒泡排序和快速排序；(4) 选择排序：简单选择排序和堆排序；(5) 归并排序：2-路归并排序；(6) 排序算法分析：各种排序算法的比较和移动次数，时间复杂度和空间复杂度的分析。

2.章节教学目标：明确排序的基本概念，排序方法的分类。深刻理解排序算法的过程、特点及其依据的原则，并能加以灵活应用。掌握各种排序方法的时间和空间复杂度的分析方法。能从关键字间的比较次数和移动次数分析算法的平均情况和最坏情况的时间性能。理解排序方法“稳定”或“不稳定”的含义，弄清楚在什么情况下要求应用的排序方法必须是稳定的。快速排序、堆排序和归并排序等高效排序方法是本章的学习重点和难点。

3.拟解决主要问题：快速排序算法及其性能分析；快速排序算法的编程实现。

4.思政结合点：在教学过程中，融入“树立坚忍不拔的科学探索精神，汲取知识，提升职业创造能力”的思政教育。

5.教学方法和载体途径：本案例教学采用教师讲解+课堂互动+学生练习的形式。其中，讲解依托于博思智慧教学平台的直播间实现；互动采用随机抽取、抢答等方式进行；练习通过博思课程的“讨论”模块实现。教学过程中学生通过手机或电脑远程学习；教师课件演示采用 PPT，参考书采用 PDF 电子版教材，编程所用工具为 DEV C++，flash 动画使用 IE 进行播放。

四、教学目标

1.知识与能力目标

掌握快速排序的算法过程，能够运用代码实现快速排序算法；掌握快速排序算法的时间复杂度和空间复杂度；理解快速排序算法的稳定性分析。

2.育人目标

教学过程中融入思政教育，教会学生运用马克思主义辩证哲学思维方法；树立坚忍不拔的科学探索精神，汲取知识，提升职业创造能力；培育学生踏实勤奋、吃苦耐劳、精益求精、实践创新的工匠精神。

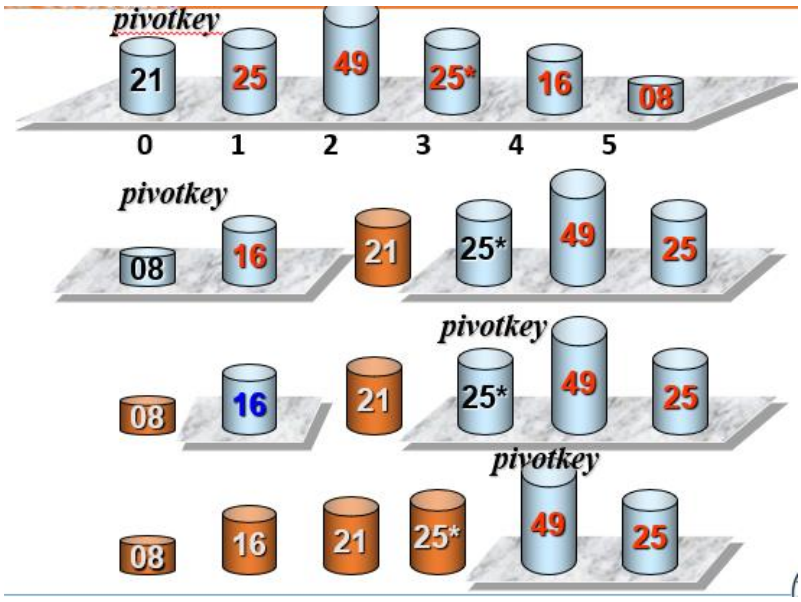
五、教学过程

1.引言（1 分钟）

交换排序是一种内部排序方法，它的基本思想是两两比较，如果发生逆序则交换，直到所有记录都排好序为止，而快速排序法属于交换排序的 1 种。

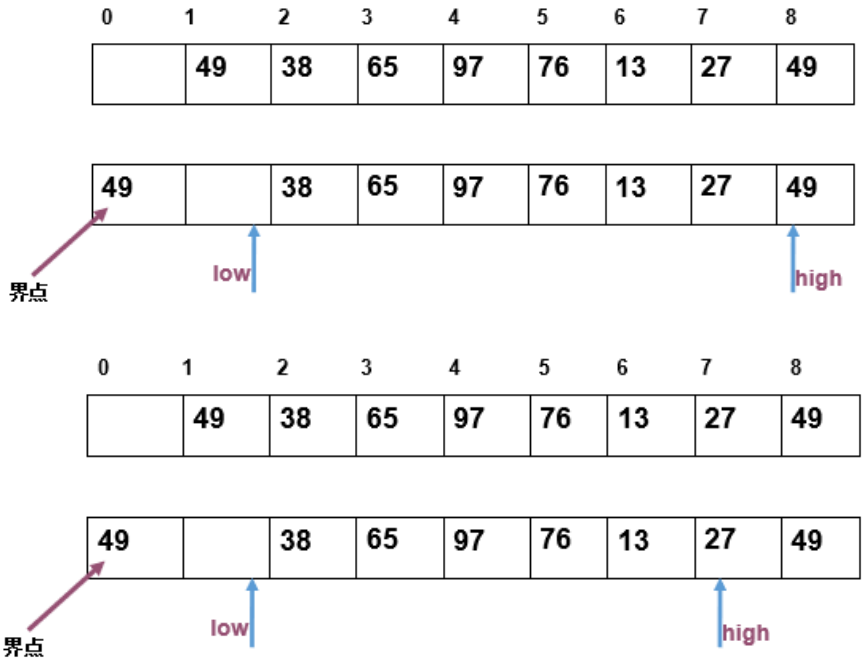
2.快速排序算法讲解（10 分钟）

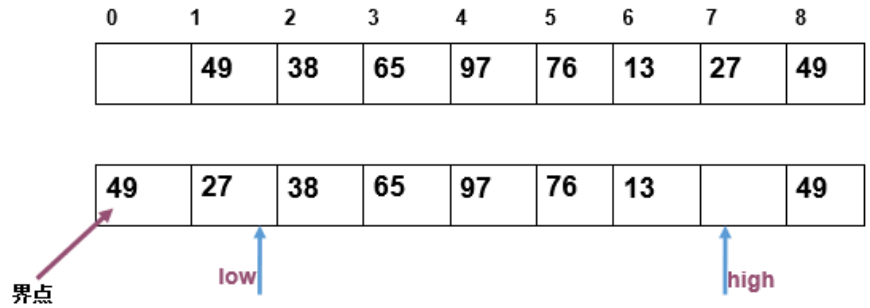
（1）快速排序法的一般做法是：任取一个元素 (如第一个) 为中心,所有比它小的元素一律前放，比它大的元素一律后放，形成左右两个子表；对各子表重新选择中心元素并依此规则调整，直到每个子表的元素只剩一个。如下图所示：

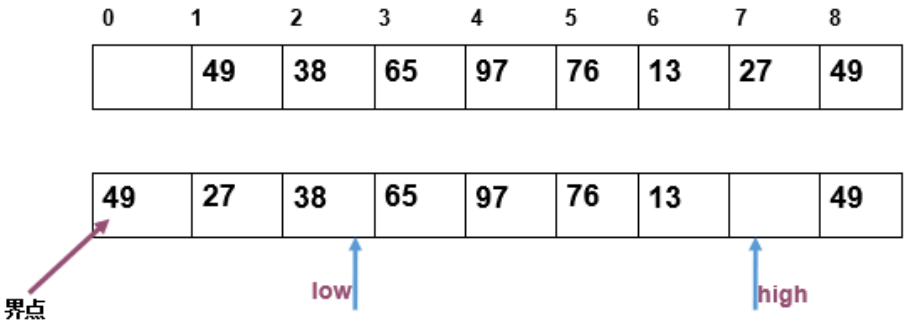


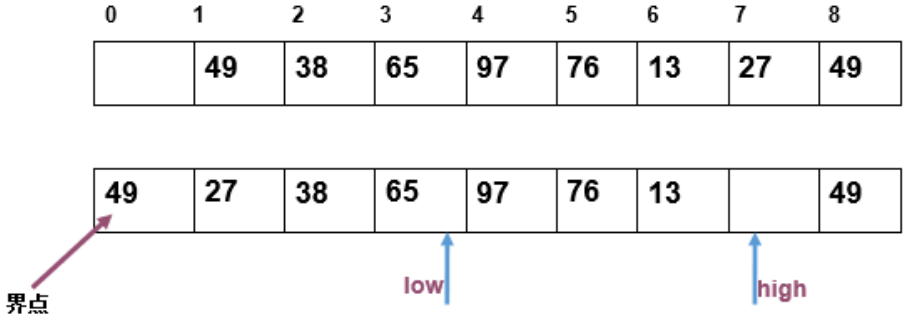
说明其中棕色数据为作为中心的数据，图中 08、16、21、25 为只有一个元素的子表，而 49、25 还需要继续排序。

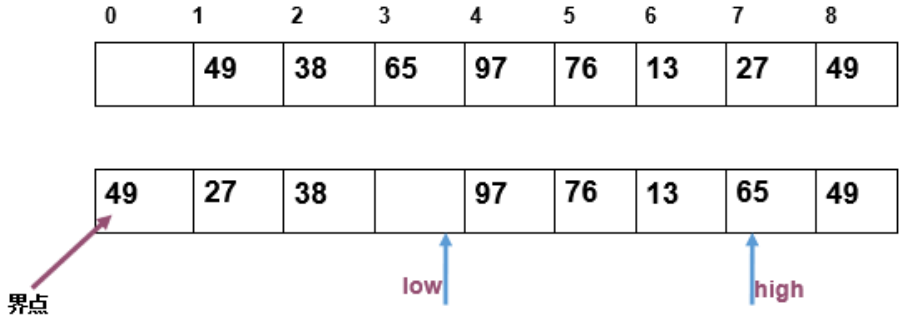
（2）算法过程手动演示

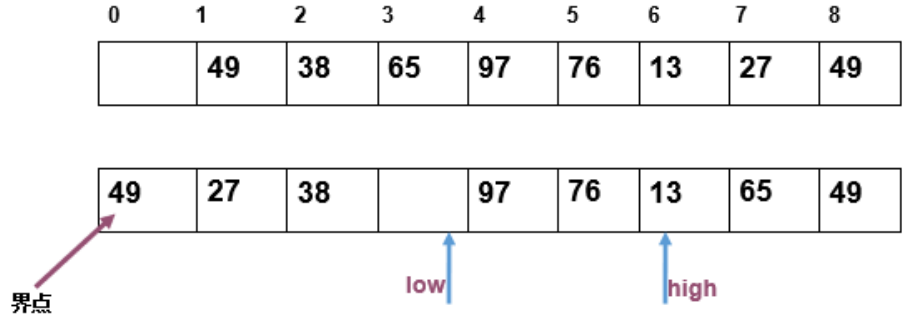


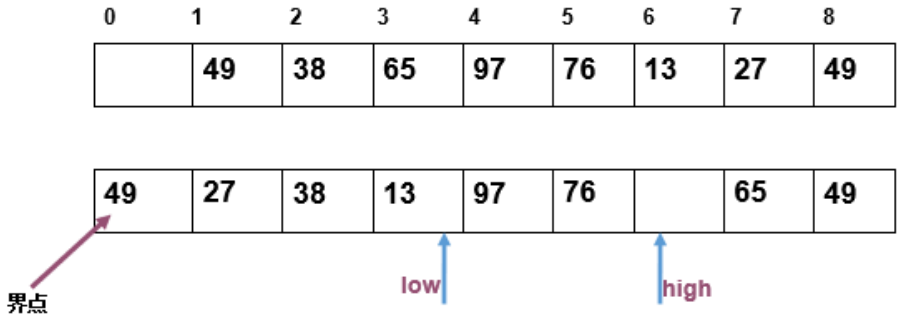

0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 65 97 76 13 49
界点
low
high

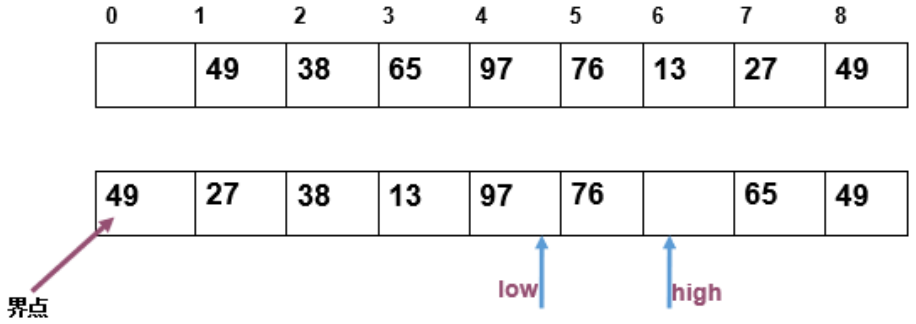

0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 65 97 76 13 49
界点
low
high

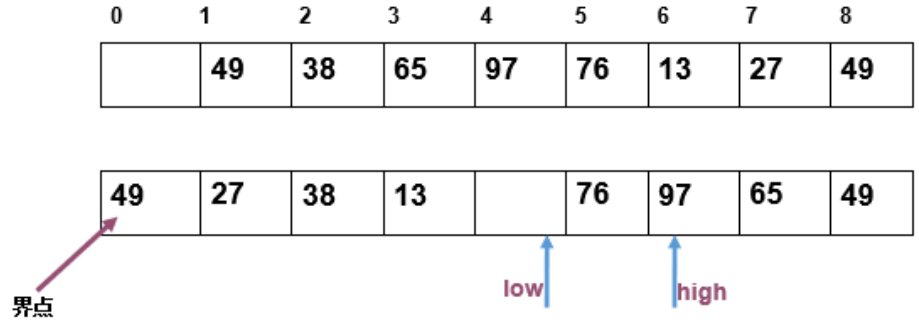

0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 65 97 76 13 49
界点
low
high

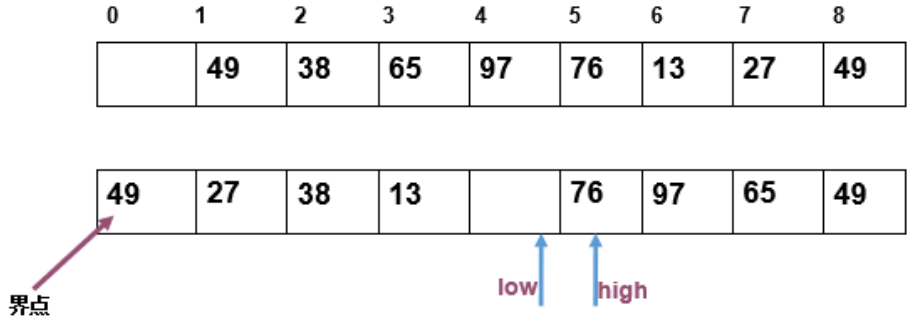

0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 65 97 76 13 65 49
界点
low
high

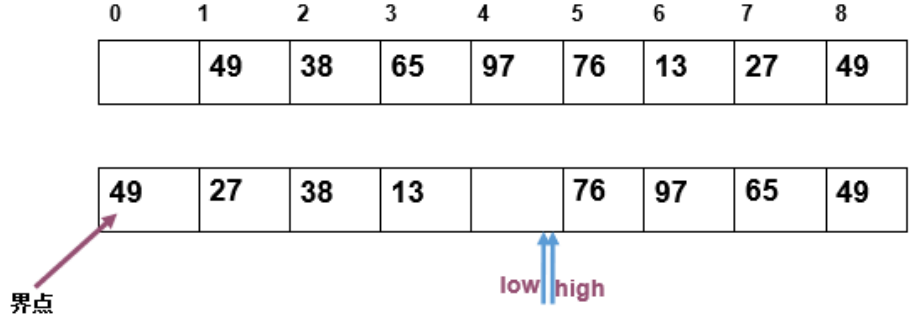

0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 97 76 13 65 49
界点
low
high

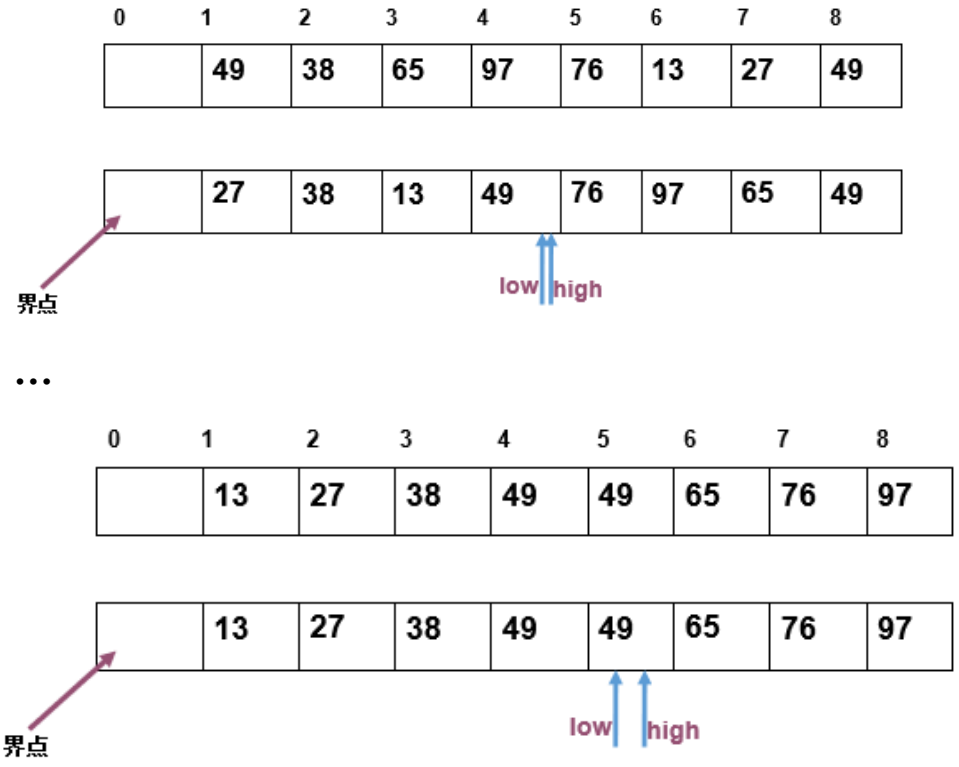

0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 13 97 76 65 49
low
high
界点


0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 13 97 76 65 49
low
high
界点


0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 13 76 97 65 49
low
high
界点

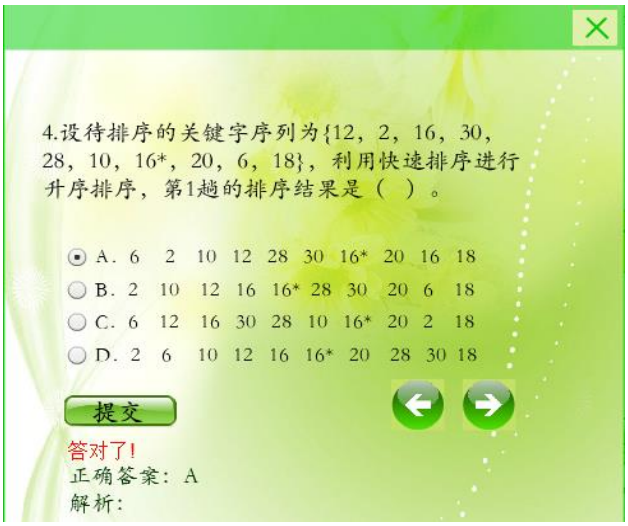

0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 13 76 97 65 49
low
high
界点


0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 49
49 27 38 13 76 97 65 49
low
high
界点



手动演示完成后，以选择题形式，提出问题，通过随机选人手段，选择学生回答。

【互动 1】



【互动 2】

3.算法代码解析（10 分钟）

说明：①每一趟的子表的形成是采用从两头向中间交替式逼近法；

②由于每趟中对各子表的操作都相似，可采用递归算法。

（1）展示递归算法代码：

```
void main ( )
 {    QSort ( L, 1, L.length ); }
void QSort ( SqList &L，int low,  int  high )
{  if  ( low < high )
     {  pivotloc = Partition(L, low, high ) ;
        Qsort (L, low, pivotloc-1) ;
        Qsort (L, pivotloc+1, high )
      }
}
```

指出上述代码中 Partition, Qsort 均为递归。

（2）列出 Partition 代码：

```
int Partition ( SqList &L，int low,  int  high )
{  L.r[0] = L.r[low];    pivotkey = L.r[low].key;
   while  ( low < high )
    { while ( low < high && L.r[high].key >= pivotkey )  --high;
                  L.r[low] = L.r[high];
      while ( low < high && L.r[low].key <= pivotkey )  ++low;
                  L.r[high] = L.r[low];
     }
    L.r[low]=L.r[0];
    return low;
}
```

说明为一趟快速排序算法，并对代码逐行解释，特别提醒学生注意①high 指针和 low 指标的移动规律：high 由右向左，所以是自减；low 由左向右，因此是自加；②当 low 指针和 high 指向同一位置时，将参照位数据插入，一趟排序结束。

（3）用 DEV C++演示快速排序的程序

```
//算法8.5 快速排序
#include <iostream>
using namespace std;
#define  MAXSIZE  20                    //顺序表的最大长度
typedef struct
{
    int key;
    char *otherinfo;
}ElemType;
//顺序表的存储结构
typedef struct
{
    ElemType *r;                                    //存储空间的基地址
    int  length;                                    //顺序表长度
}SqList;                                            //顺序表类型


int Partition(SqList &L,int low,int high)
{
    //对顺序表L中的子表r[low..high]进行一趟排序，返回枢轴位置
    int pivotkey;
    L.r[0]=L.r[low];                        //用子表的第一个记录做枢轴记录
    pivotkey=L.r[low].key;                  //枢轴记录关键字保存在pivotkey中
    while(low<high)
    {                                       //从表的两端交替地向中间扫描
        while(low<high&&L.r[high].key>=pivotkey) --high;
        L.r[low]=L.r[high];                 //将比枢轴记录小的记录移到低端
        while(low<high&&L.r[low].key<=pivotkey) ++low;
        L.r[high]=L.r[low];                 //将比枢轴记录大的记录移到高端
    }//while
    L.r[low]=L.r[0];                        //枢轴记录到位
    return  low;                            //返回枢轴位置
}//Partition

void QSort(SqList &L,int low,int high)
{   //调用前置初值：low=1; high=L.length;
    //对顺序表L中的子序列L.r[low..high]做快速排序
    int pivotloc;
    if(low<high)
    {                                       //长度大于1
       pivotloc=Partition(L,low,high);      //将L.r[low..high]一分为二，pivotloc是枢轴位置
       QSort(L,low,pivotloc-1);             //对左子表递归排序
       QSort(L,pivotloc+1,high);            //对右子表递归排序
    }
}                                           //QSort

void QuickSort(SqList &L)
{
    //对顺序表L做快速排序
    QSort(L,1,L.length);
}                                           //QuickSort
```

```
53  void Create_Sq(SqList &L)
54  {
55      int i,n;
56      cout<<"请输入数据个数，不超过"<<MAXSIZE<<"个。"<<endl;
57      cin>>n;                                          //输入个数
58      cout<<"请输入待排序的数据:\n";
59      while(n>MAXSIZE)
60      {
61          cout<<"个数超过上限，不能超过"<<MAXSIZE<<"，请重新输入"<<endl;
62          cin>>n;
63      }
64      for(i=1;i<=n;i++)
65      {
66          cin>>L.r[i].key;
67          L.length++;
68      }
69  }
70
71
72  void show(SqList L)
73  {
74      int i;
75      for(i=1;i<=L.length;i++)
76          cout<<L.r[i].key<<endl;
77  }
78
79
```

```
80  main()
81  {
82      SqList L;
83      L.r=new ElemType[MAXSIZE+1];
84      L.length=0;
85      Create_Sq(L);
86      QuickSort(L);
87      cout<<"排序后的结果为："<<endl;
88      show(L);
89  }
```

①将快速算法涉及到的语句段标出；

②说明：虽然教材上先介绍 Qsort 程序段，再介绍 Partition 程序段，但在编程实现时，由于 Qsort 程序段要调用 Partition 程序段，所以 Partition 程序段需要写在前面；

③在主程序中调用上述程序段。

4.算法性能分析（6 分钟）

可以证明，平均计算时间是 O(nlog2n)。

实验结果表明：就平均计算时间而言，快速排序是我们所讨论的所有内部排序方法中最好的一个。

快速排序是递归的，需要有一个栈存放每层递归调用时参数（新的 low 和 high）。

最大递归调用层次数与递归树的深度一致，因此，要求存储开销为 O(log2n) 。

最好：划分后，左侧右侧子序列的长度相同。

【互动 3】提问：什么情况是“最好”情况？

【互动 4】提问：什么情况是“最坏”情况？

最坏：从小到大排好序，递归树成为单支树，每次划分只得到一个比上一次少一个对象的子序列，必须经过 n-1 趟才能把所有对象定位，而且第 i 趟需要经过 n-i 次关键码比较才能找到第 i 个对象的安放位置。

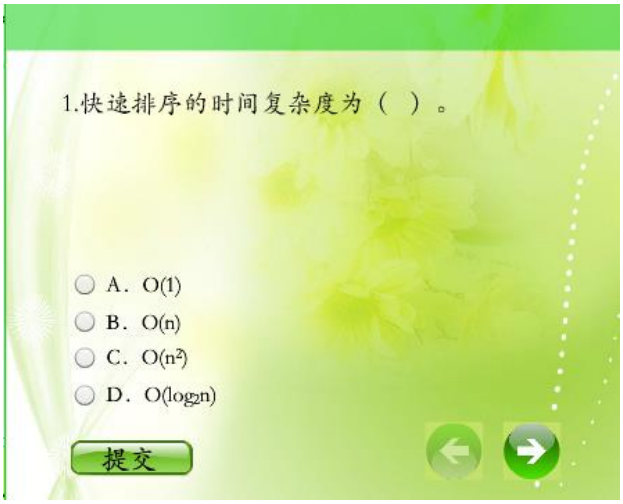
$$\sum_{i=1}^{n-1}(n-i)=\frac{1}{2}n(n-1)\approx\frac{n^2}{2}$$

时间效率：O(nlog2n) —每趟确定的元素呈指数增加；

空间效率：O（log2n）—递归要用到栈空间；

稳 定 性：不稳定—可选任一元素为支点。

【互动 5】



【互动 6】

【互动 7】


3.快速排序的稳定性为（ ）。
A．稳定
B．不稳定
C．无法确定
提交

【互动 8】


5.快速排序在下列（ ）情况下最易发挥其长处。
A．被排序的数据中含有多个相同排序码
B．被排序的数据已基本有序
C．被排序的数据完全无序
D．被排序的数据中的最大值和最小值相差悬殊
提交
答对了!
正确答案：C
解析：B选项是快速排序的最坏情况。

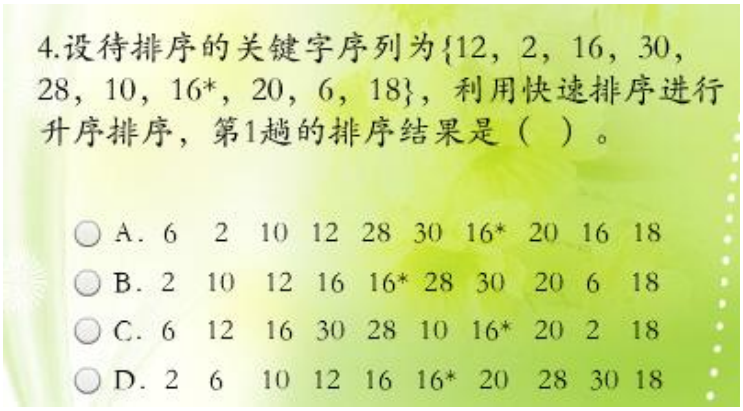
【互动 9】


6.对n个关键字作快速排序，在最坏情况下，算法的时间复杂度是（ ）。
A．O(n)
B．O(n2)
C．O(nlog2n)
D．O(n3)
提交
答对了!
正确答案：B
解析：快速排序的平均时间复杂度为O(nlog2n)，但在最坏情况下，即关键字基本排好序的情况下，时间复杂度为O(n2)。

5.算法动画演示（3 分钟）

将代码与实现效果结合起来，用动画演示整个快速排序过程，加深对过程和算法代码的理解。

6.排序练习（12 分钟）



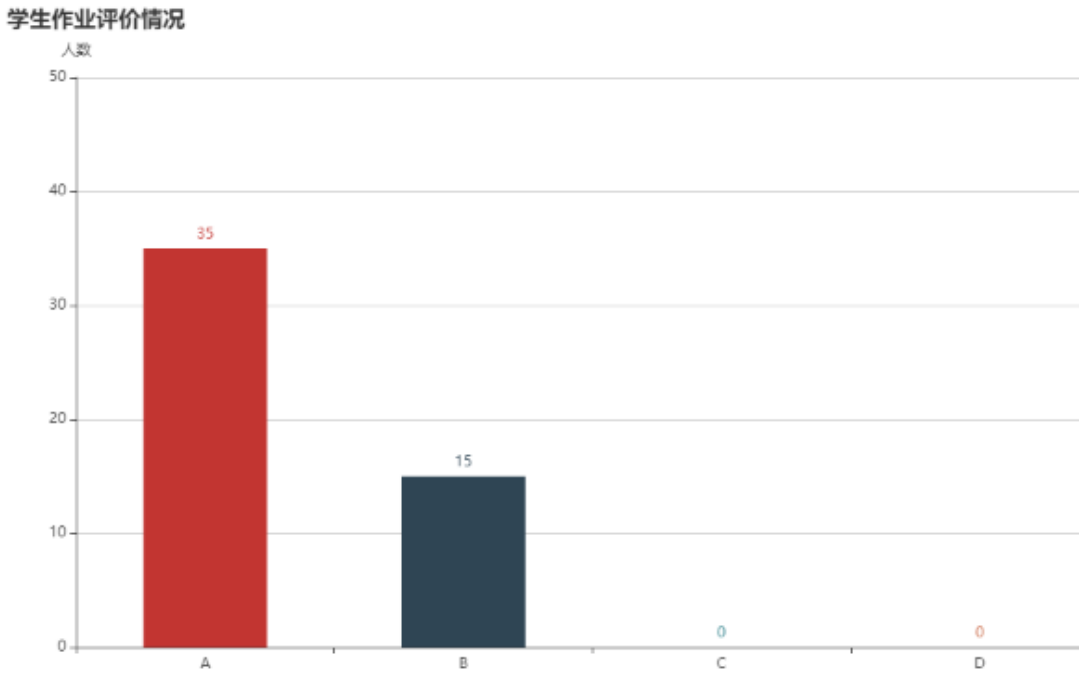
设待排序的关键字序列为{12，2，16，30，28，10，16*，20，6，18}，使用快速排序法进行排序。要求：写出每一趟的排序过程与结果，并将答案截图上传到博思讨论区。最先完成学生+3 分；完成且正确的学生+2 分；完成但不正确的学生+1 分。

7.课程小结（3 分钟）

对快速排序法的方法、算法及性能进行回顾总结。

8.线上作业与集中答疑

通过博思平台发布作业，要求学生在规定时间内提交。为帮助所有学生掌握基础知识内容，且减少抄袭现象，在截止时间后开设一次习题课，对每道题均进行讲解，对集中出现的问题着重介绍，并且允许之前做得不满意学生撤回作业并修改后重新提交。



⑤ 快速排序↵

12 [6 2 10] **12** [28 30 16* 20 16 18]

6 [2] **6** [10] 12 [28 30 16* 20 16 18]

28 2 6 10 12 [18 16 16* 20] **28** [30]

18 2 6 10 12 [16* 16] **18** [20] 28 30

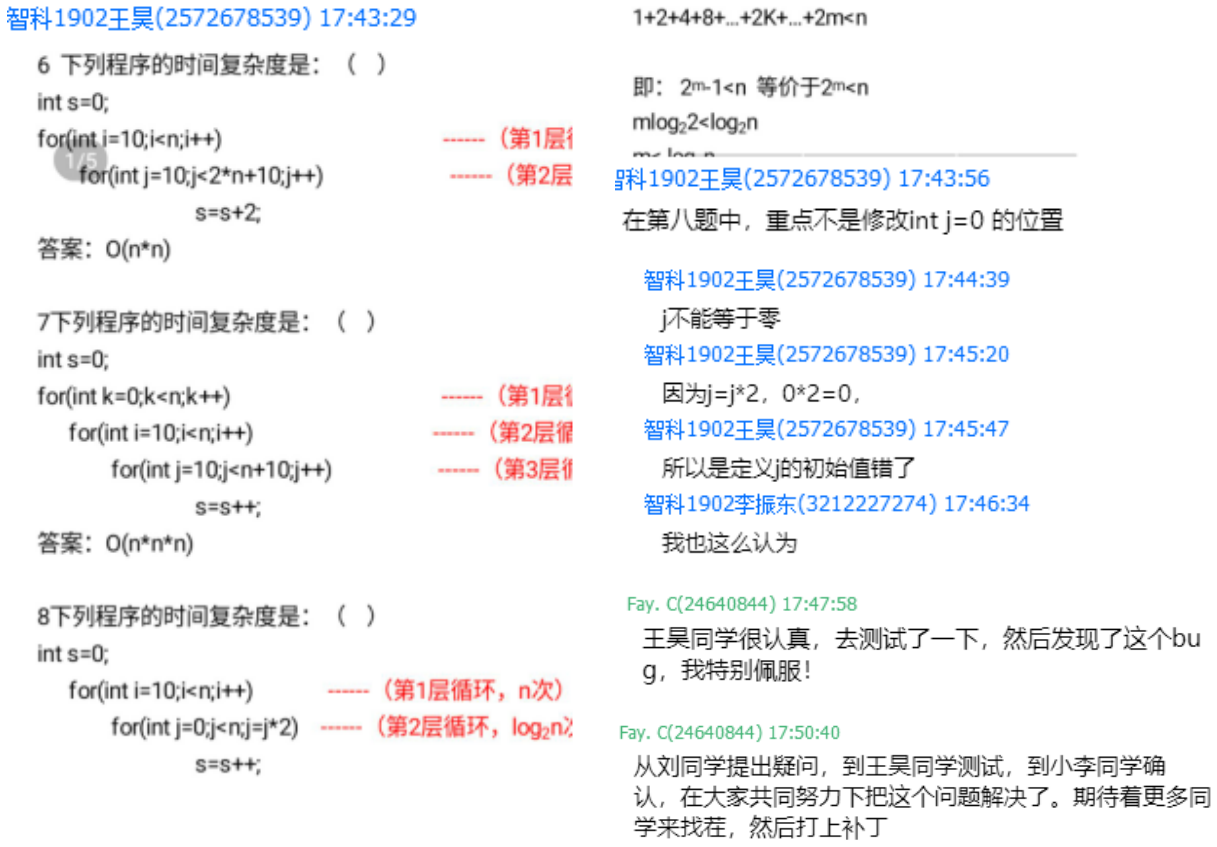
16* 2 6 10 12 16* [16] 18 20 28 30

左子序列递归深度为 1，右子序列递归深度为 3

作业提交率为 100%，且重复率很低，说明绝大部分学生均能独立完成作业。

9.QQ 在线答疑

除习题课之外的问题，对于大部分学生均存在的公共问题，通过 QQ 群讨论、答疑，对于个别学生的问题，通过 QQ 答疑。



六、教学效果与特色创新

1.将算法与代码运行效果对照起来，用动画展示，便于学生直观理解抽象的程序代码。如图所示：

2.教学过程中采用随机抽人回答问题的形式，一方面可以考察学生对知识的掌握程度，同时也可以随时检查学生是否坚持在线听课。

3.练习的布置让学生加深对快速排序方法的理解，有益于牢固掌握新知识。

4.练习时对于有尝试但未获得正确答案的学生一样有加分，有助于鼓励更多的学生参与到课堂学习中。

七、教学反思

无论课堂教学还是线上教学，均需要学生对课程抱有兴趣、并且愿意参与进来才能获得最好的效果。此次线上教学过程中，以能力培养为宗旨、知识传递为目标、参与互动为手段，让学生由“要我学”变成“我爱我”，在教师与学生的

共同努力下达成教学目标。此外，由于网络卡顿等原因，有些内容学生可能无法听清，因而线上教学需要更多的耐心，以及对重点问题的不断强调与重复。本课程共 154 位学生选课，最终考核成绩只有 3 位学生不合格，不合格率仅有 2%，说明本课程所采用教学手段较为有效。

此外，本课程需要与实践相结合，原本应该由学生完成算法编程的实验，由于疫情关系，无法在实验室实现，因此，线上教学迫切需要有效的实验设计。

八、教学资源

1.教材:严蔚敏 李冬梅. 数据结构(C 语言版)（“十二五”国家级规划教材)[M]. 人民邮电出版社, 2015. 教材电子版地址：
http://learn.iflysse.com/web/teacher/mycourse?id=9dd80309-1f53-43e9-890f-9ed3ddb18ac5

2.课程学习平台：
http://learn.iflysse.com/web/teacher/online-class?id=4337b793-87ca-4519-b77d-5949a35a8c48

3.QQ 讨论群：