西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

实现排序算法

5、快速排序

6、归并排序

7、希尔排序

8、堆排序

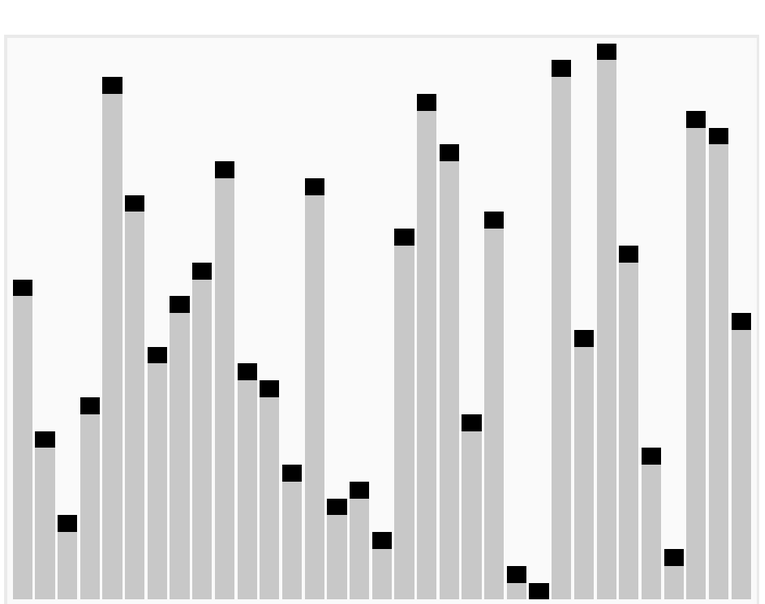
9、非比较型排序

- （4）快速排序法
- 快速排序法（quick sort）的基本思想是：通过一趟排序将要排序的记录分割成独立的两部分，其中一部分的所有记录关键码比另外一部分的记录关键码都要小，然后再按此方法对这两部分数据分别进行递归快速排序，从而使序列成为有序序列。

6 5 3 1 8 7 2 4

6.5 数组应用程序举例

- 设由A[1]～A[n]组成的n个数据，选取第一个数据作为关键数据，然后将所有比它小的数据都放到它前面，所有比它大的数据都放到它后面，称为一趟快速排序。其算法是：



6.5 数组应用程序举例

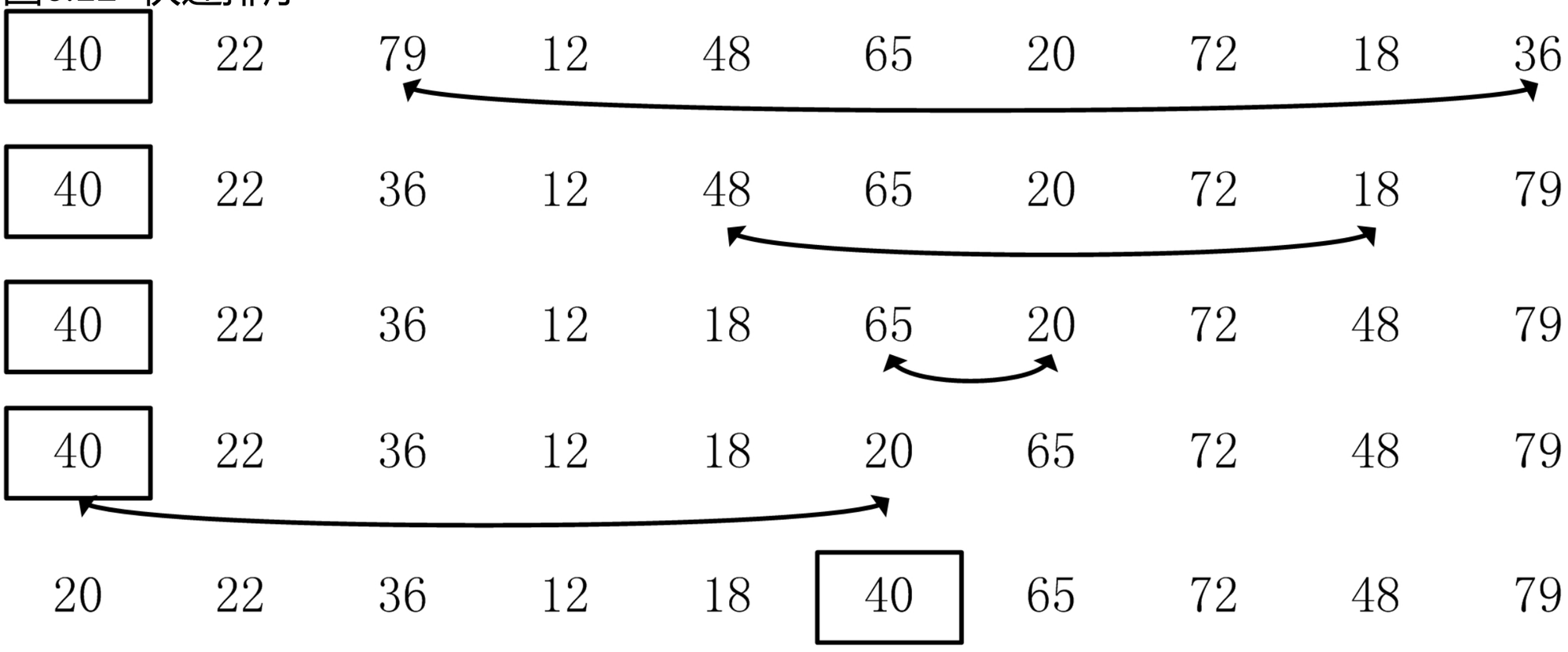
- ①设置两个变量i、j，排序开始的时候i=左边界，j=右边界，令关键数据s=A[i]；

- ②从i开始向后搜索，直到找到大于s的数；

- ③从j开始向前搜索，直到找到小于s的数；
- ④如果i<j，则交换A[i]和A[j]；
- ⑤重复第2、3、4、5步，直到i>=j；将关键数据与A[j]交换。

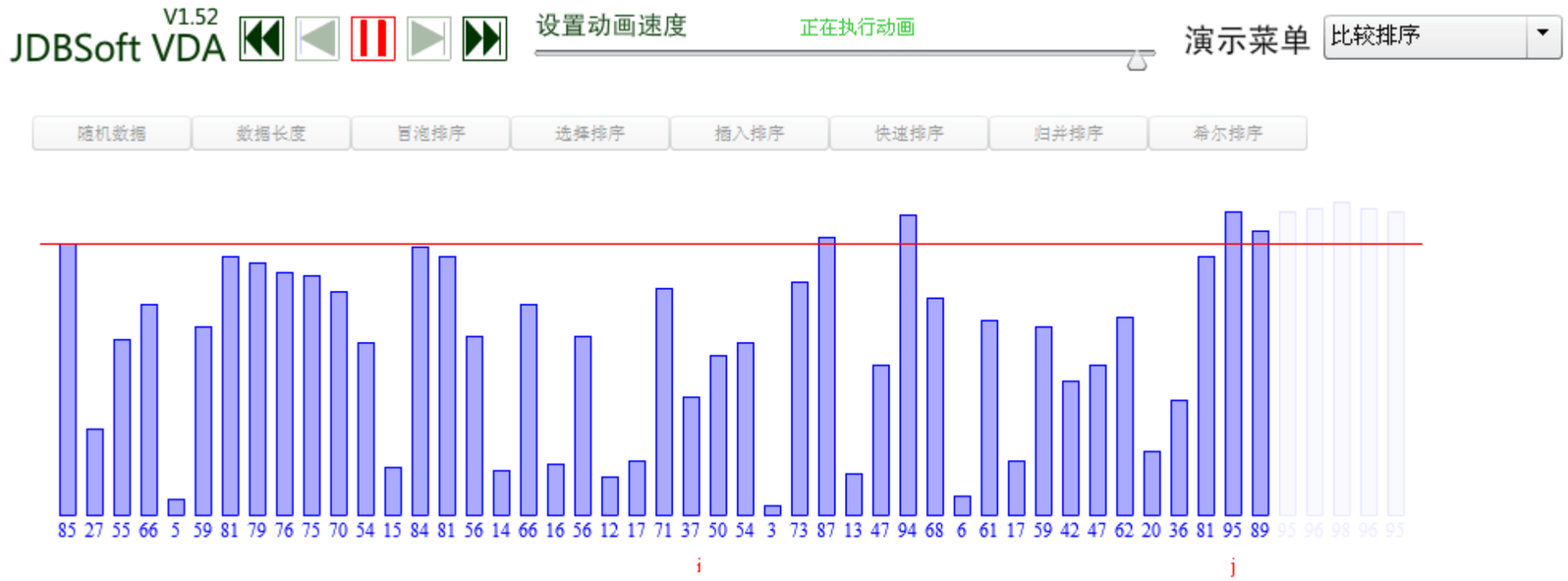
6.5 数组应用程序举例

图6.12 快速排序



6.5 数组应用程序举例

- 快速排序过程演示



6.5 数组应用程序举例

【例6.13】

编写快速排序函数QuickSort，将一个数组由小到大排序。

6.5 数组应用程序举例

例6.13

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void QuickSort(int A[],int n,int left,int right)
{ //快速排序 n为数组元素个数 left=数组左边界 right=数组右边界
  int i,j,t;
  if(left<right) { //一趟快速排序
    i=left; j=right + 1;
    while(1) {
      while(i+1<n&& A[++i]<A[left]); //向后搜索 <升序 >降序
      while(j-1>-1&& A[--j]>A[left]); //向前搜索 >升序 <降序
      if(i>=j) break;
      t=A[i], A[i]=A[j], A[j]=t; //交换
    }
    t=A[left], A[left]=A[j], A[j]=t; //交换
```

6.5 数组应用程序举例

例6.13

```
16      QuickSort(A, n, left, j-1); //关键数据左半部分递归
17      QuickSort(A, n, j+1, right); //关键数据右半部分递归
18    }
19 }
20 #define N 10
21 int main()
22 {
23    int A[N],i;
24    srand((unsigned int)time(0)); //设置随机数种子
25    for(i=0; i<N; i++) { //随机产生N个数
26      A[i] = rand()%100;
27      printf("%d ",A[i]);
28    }
29    QuickSort(A,N,0,N-1);
30    printf("\n");
```

6.5 数组应用程序举例

例6.13

```
31    for(i=0; i<N; i++) printf("%d ", A[i]); //输出排序结果
32    return 0;
33 }
```

6.5 数组应用程序举例

例6.13

程序运行屏幕

```
89 45 66 84 77 1 42 32 8 8
1 8 8 32 42 45 66 77 84 89
```

6.5 数组应用程序举例

- （5）排序算法比较与选择
- 不同排序算法时间和空间性能比较见表。

性能	冒泡排序	选择排序	插入排序	快速排序
时间复杂度	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n\log n)$
空间复杂度	$O(1)$	$O(1)$	$O(1)$	$O(\log n)$
优点	稳定	稳定	快	极快
缺点	慢	慢	数据移动多	不稳定

6.5 数组应用程序举例

- 因为不同的排序方法适应不同的应用环境和要求，所以选择合适的排序方法应综合考虑因素：待排序记录数目、记录规模、关键字结构及其初始状态、稳定性要求、存储结构、时间和辅助空间复杂度等。一般建议：
- ①若n较小（如n≤50），选用插入排序或选择排序；
- ②若数据集合初始状态基本有序，选用插入排序、冒泡排序；
- ③若n较大，选用快速排序。

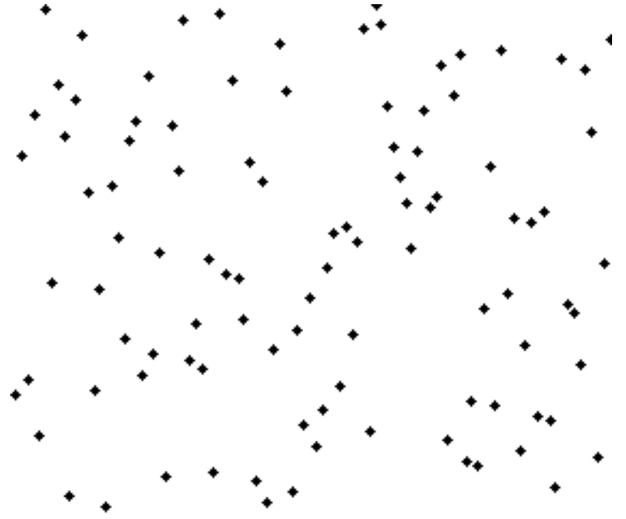
- （6）归并排序法
- 归并排序法（merge sort）是将两个（或两个以上）有序表合并成一个新的有序表，即把待排序序列分为若干个子序列，每个子序列是有序的，然后再把有序子序列合并为整体有序序列。归并排序法又称合并排序法。

6 5 3 1 8 7 2 4

6.5 数组应用程序举例

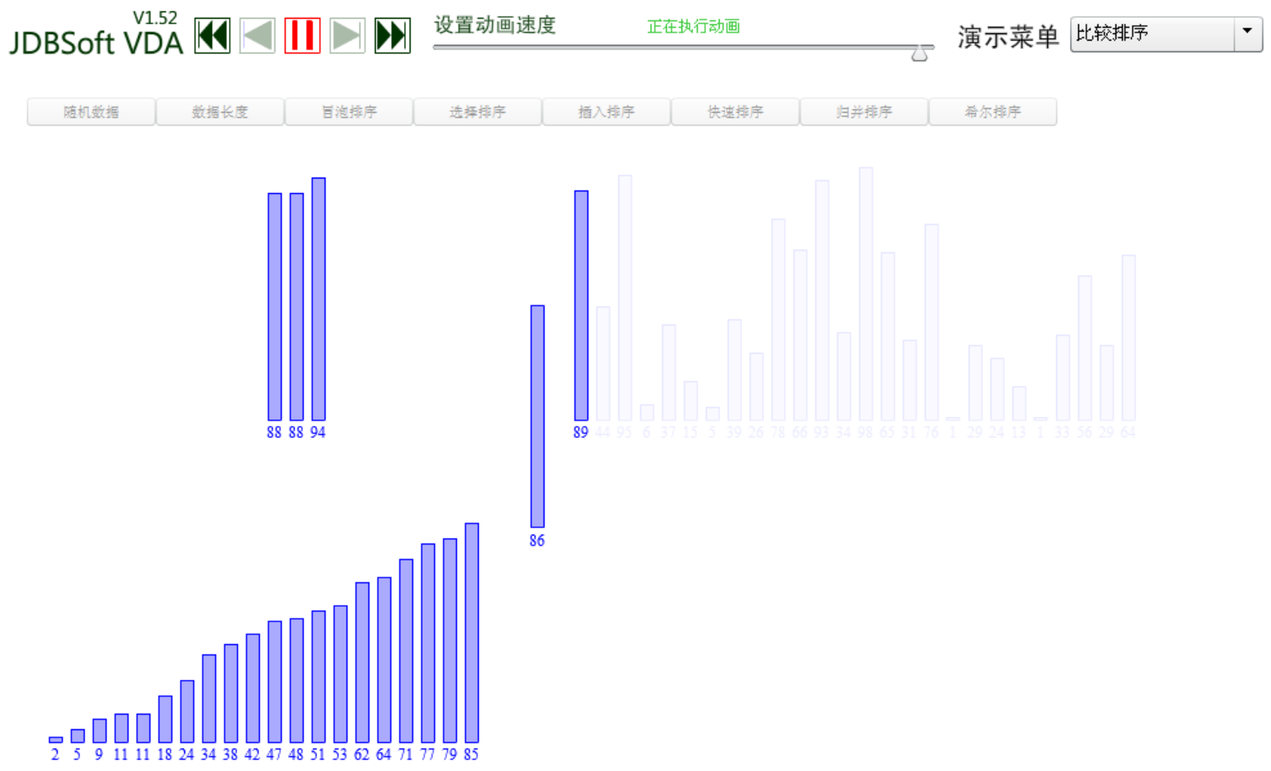
- 归并排序是建立在归并操作上的排序算法，采用分治法（Divide and Conquer）。其工作原理如下：
- （1）申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列；
- （2）设定两个指针，最初位置分别为两个已经排序序列的起始位置；

- （3）比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置；
- （4）重复步骤3直到某一指针达到序列尾；
- （5）将另一序列剩下的所有元素直接复制到合并序列尾。

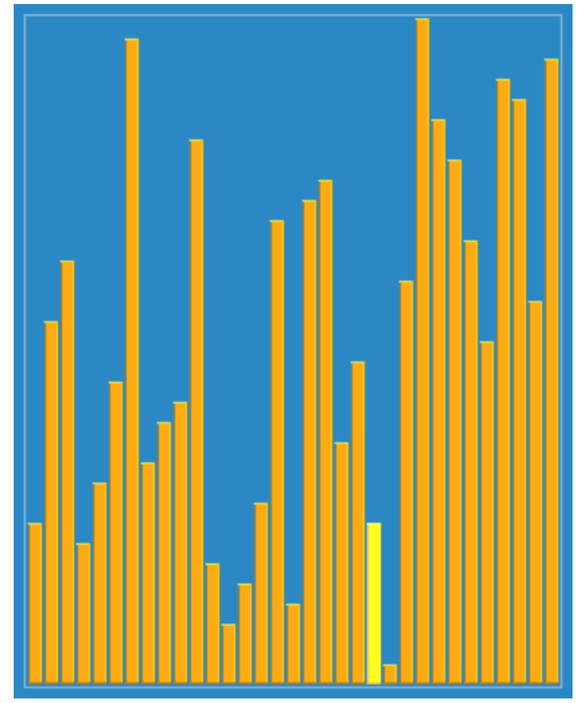


6.5 数组应用程序举例

- 归并排序过程演示



- （7）希尔排序法
- 希尔排序法(shell sort)是针对直接插入排序算法的改进，该方法又称缩小增量排序。

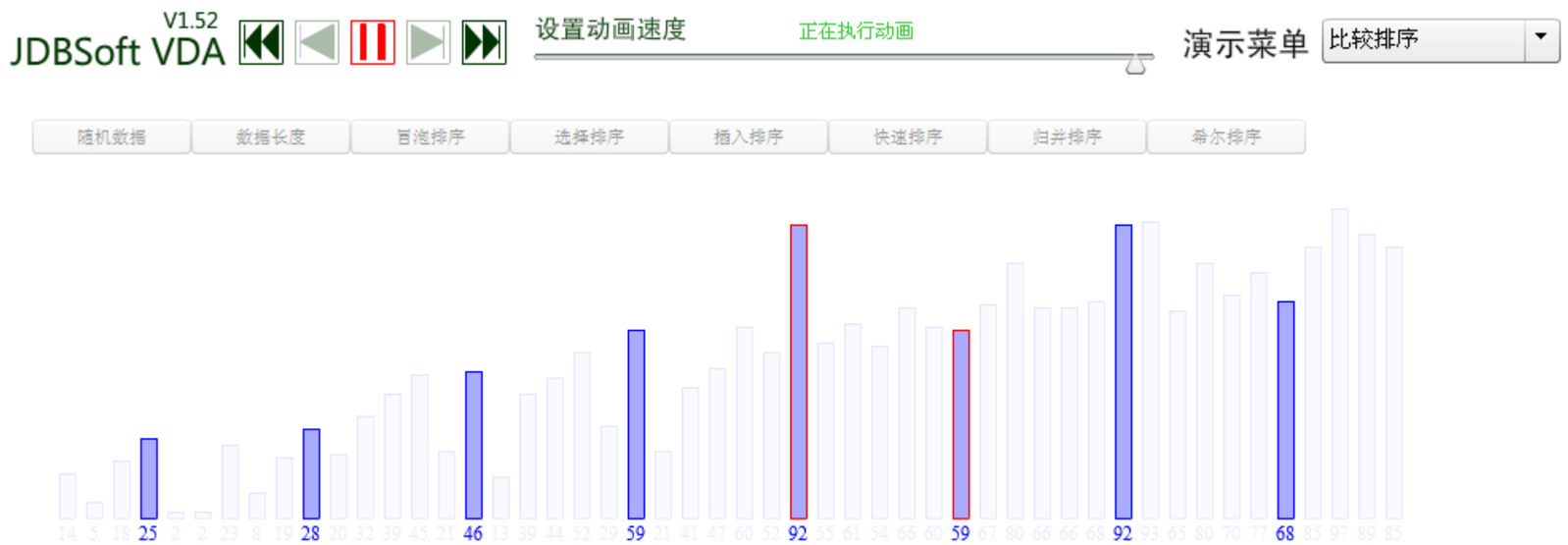


6.5 数组应用程序举例

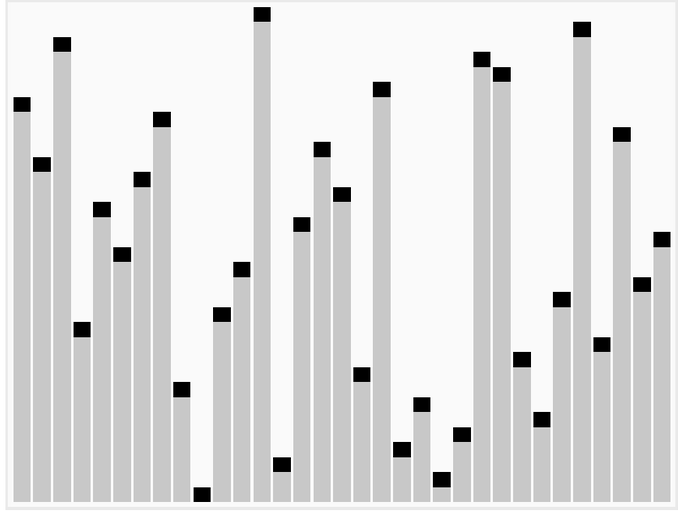
- 通过比较相隔较远距离（称为增量）的数，使得数移动时能跨过多个元素，则进行一次比较就可能消除多个元素交换。
- 希尔排序法先将要排序的一组数按某个增量d分成若干组，每组中记录的下标相差d。对每组中全部元素进行排序，然后再用一个较小的增量，在每组中再进行排序。当增量减到1时，整个要排序的数被分成一组，排序完成。

6.5 数组应用程序举例

- 希尔排序过程演示

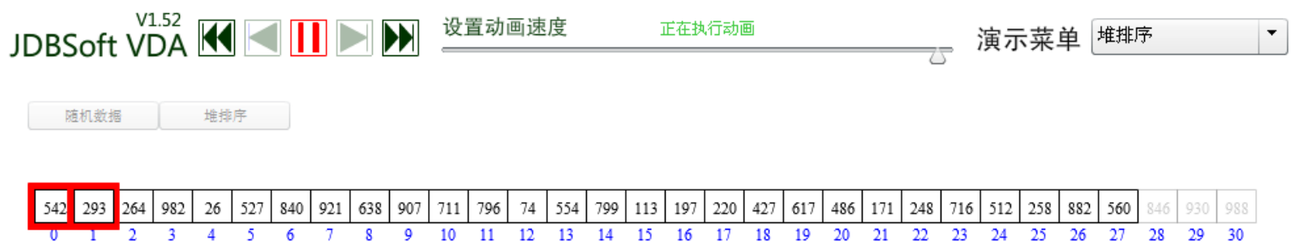


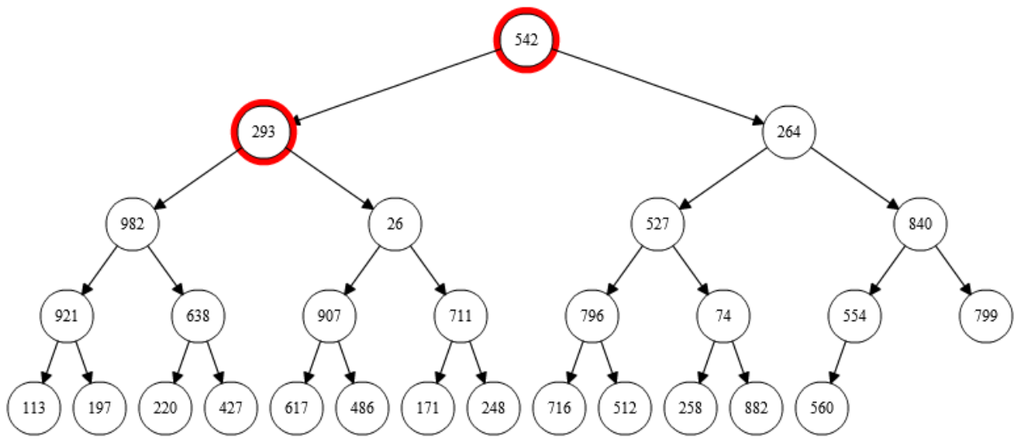
- （8）堆排序法
- 堆排序法(heap sort)是指利用堆积树（堆）这种数据结构所设计的一种排序算法，可以利用数组的特点快速定位指定索引的元素。堆排序要用到“树”的数据结构。



6.5 数组应用程序举例

- 堆排序过程演示





- （9）非比较型排序法
- (A)桶排序法
- 桶排序法（bucket sort）的思想就是把区间[0，1)划分成n个相同大小的子区间，简称桶，然后将n个输入数分布到各个桶中去。因为输入数均匀分布在[0，1)上，所以一般不会有很多数落在一个桶中的情况。为得到结果，先对各个桶中的数进行排序，然后按次序把各桶中的元素列出来即可。

6.5 数组应用程序举例


桶排序过程演示
V1.52
JDBSoft VDA
设置动画速度
正在执行动画
演示菜单
桶排序
随机数据
桶排序
276 882 392 32 90 294 35
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
696
94 175 221 288 425 488 689
10 93 170 212 281 363 406 480 519 562 644 688 767 852 924
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

6.5 数组应用程序举例

- (B)计数排序法
- 计数排序法（counting sort）是一个类似于桶排序的排序算法，其优势是对已知数量范围的数组进行排序。它创建一个长度为这个数据范围的数组C，C中每个元素记录要排序数组中对应记录的出现个数。

6.5 数组应用程序举例

计数排序过程演示

25	28	26	17	16	29	1	21	5	15	4	11																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

0	3	4	5	6	7	7	7	7	7	9	10	11	12	12	15	16	18	18	18	18	20	21	21	21	22	23	24	25	29	30
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

0	0		1	2			9	9	10		12	14	14			16		20		21			26		28	28	28		29
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

6.5 数组应用程序举例

- (C)基数排序法
- 基数排序法（radix sort）属于“分配式排序”（distribution sort），它是通过键值的部份信息，将要排序的元素分配至某些“桶”中，藉以达到排序的作用。基数排序法是属于稳定性的排序，在某些时候，基数排序法的效率高于其它的比较性排序法。

6.5 数组应用程序举例

▶基数排序过程演示

966	344	48	350	200	480	773	859	856	573	147	529	734															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

3	4	5	10	12	13	15	18	21	24
0	1	2	3	4	5	6	7	8	9

			310	651	422	202	522					944			726	776		337	977		58			549	279	429	989	269	649
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29


程序设计