**第九章 排序**

**一、单项选择题**

1．（1） DC （2）ADF （3）B （4）ACF BDE

2．D

3．A

4．C

5．C

6．C

7. D

8. A

9. C

10. D

11．C

12. C

**二、填空题**

1.稳定、不稳定

2.内部、外部

3．免去查找过程中每一步都要检测整个表是否查找完毕，提高了查找效率。

4．n(n-1)/2

5．题中为操作方便，先增加头结点（最后删除），p指向无序区的前一记录，r指向最小值结点的前驱，一趟排序结束，无序区第一个记录与r所指结点的后继交换指针。

(1)q->link!=NULL (2)r!=p (3)p->link (4)p->link=s (5)p=p->link

6．.(1)i<n-i+1 (2)j<=n-i+1 (3)r[j].key<r[min].key (4)min!=i (5)max==i (6)r[max]<-->r[n-i+1]

7．(1)1 (2)a[i]=t (3)(i=2;i<=n;i+=2) (4)1 (5)flag

**三、应用题**

1． 解答：①直接插入排序

序号　　1　　2　　3　　4　　5　　6　　7　　8　　9　　10　　11　　12

　 关键字　83 40　 63 13 84 35 96 57 39 79 61 15

i = 2 40 83 [63 13 84 35 96 57 39 79 61 15]

i = 3 40 63 83 [13 84 35 96 57 39 79 61 15]

i = 4 13 40 63 83 [84 35 96 57 39 79 61 15]

i = 5 13 40 63 83 84 [35 96 57 39 79 61 15]

i = 6 13 35 40 63 83 84 [96 57 39 79 61 15]

i = 7 13 35 40 63 83 84 96 [57 39 79 61 15]

i = 8 13 35 40 57 63 83 84 96 [39 79 61 15]

i = 9 13 35 39 40 57 63 83 84 96 [79 61 15]

i = 10 13 35 39 40 57 63 79 83 84 96 [61 15]

i = 11 13 35 39 40 57 61 63 79 83 84 96 [15]

i = 12 13 15 35 39 40 57 61 63 79 83 84 96

　②直接选择排序

序号　　1　　2　　3　　4　　5　　6　　7　　8　　9　　10　　11　　12

　 关键字　83 40　 63 13 84 35 96 57 39 79 61 15

i = 1 13 [40 63 83 84 35 96 57 39 79 61 15]

i = 2 13 15 [63 83 84 35 96 57 39 79 61 40]

i = 3 13 15 35 [83 84 63 96 57 39 79 61 40]

i = 4 13 15 35 39 [84 63 96 57 83 79 61 40]

i = 5 13 15 35 39 40 [63 96 57 83 79 61 84]

i = 6 13 15 35 39 40 57 [96 63 83 79 61 84]

i = 7 13 15 35 39 40 57 61 [63 83 79 96 84]

i = 8 13 15 35 39 40 57 61 63 [83 79 96 84]

i = 9 13 15 35 39 40 57 61 63 79 [83 96 84]

i = 10 13 15 35 39 40 57 61 63 79 83 [96 84]

i = 11 13 15 35 39 40 57 61 63 79 83 84 [96]

　③快速排序

　 关键字　　　83 40　 63 13 84 35 96 57 39 79 61 15

第一趟排序后 [15 40 63 13 61 35 79 57 39] 83 [96 84]

第二趟排序后 [13] 15 [63 40 61 35 79 57 39] 83 　 84 [96]

第三趟排序后 13 15 [39 40 61 35 57] 63 [79] 83 84 96

第四趟排序后 13 15 [35] 39 [61 40 57] 63 79 83 84 96

第五趟排序后 13 15 35 39 [57 40] 61 63 79 83 84 96

第六趟排序后 13 15 35 39 40 [57] 61 63 79 83 84 96

第七趟排序后 13 15 35 39 40 57 61 63 79 83 84 96

④堆排序

关键字：　83　40　63　13　84　35　96　57　39　79　61　15

排序成功的序列：96　84　83　79　63　61　57　40　39　35　15　13

　　　　　排序过程如图简答题8－1.1、8－1.2、8－1.3所示。

⑤归并排序

　 关键字　　　83 40　 63 13 84 35 96 57 39 79 61 15

第一趟排序后 [40 83] [13 63] [35 84] [57 96] [39 79] [15 61]

第二趟排序后 [13 40 63 83] [35 57 84 96] [15 39 　 61 79]

第三趟排序后 [13 35 40 57 63 83 84 96] [15 39 61 79]

第四趟排序后 13 15 35 39 40 57 61 63 79 83 84 96

2．(1)是大堆； (2)是大堆；（4）是小堆；

(3)不是堆，调成大堆 100，98，66，85，80，60，40，77，82，10，20

3．(1)①child=child+1; ②child/2

(2)不能，调为大堆：92,86,56,70,33,33,48,65,12,24

**四、算法设计题：**

1．解答：分析：每趟从单链表头部开始，顺序查找当前链值最小的结点。找到后，插入到当前的有序表区的最后。

Void selesort ( lklist L ) /\* 设链表L带头结点 \*/

{ q=L; /\* 指向第一数据前趋 \*/

while ( q-> next !=NULL )

{ pl = q -> next ;

minp =pl; /\* minp指向当前已知的最小数 \*/

while ( pl -> next != NULL )

{ if ( pl -> next -> data < minp -> data )

minp = pl -> next ; /\* 找到了更小数 \*/

pl = pl -> next ; /\* 继续往下找 \*/

}

if ( minp != q -> next) /\* 将最小数交换到第一个位置上 \*/

{ r1 = minp -> next ;

minp -> next = r1 -> next ; /\* 删除最小数 \*/

r2 = q -> next ;

q -> next = r2 -> next ; /\* 删除当前表中第一个数 \*/

r1 -> next = q -> next ;

q -> next = r1 ; /\* 将最小数插入到第一个位置上 \*/

r2 -> next = minp -> next ;

minp -> next = r2 ; /\* 将原第一个数放到最小数原位置上 \*/

}

q = q > next ; /\* 选择下一个最小数 \*/

}

}

2．**void** BubbleSort2(**int** a[],**int** n) //相邻两趟向相反方向起泡的冒泡排序算法

{ change=1;low=0;high=n-1; //冒泡的上下界

**while**(low<high && change)

{ change=0; //设不发生交换

**for**(i=low;i<high;i++) //从上向下起泡

**if**(a[i]>a[i+1]){a[i]<-->a[i+1];change=1;} //有交换，修改标志change

high--; //修改上界

**for**(i=high;i>low;i--) //从下向上起泡

**if**(a[i]<a[i-1]){a[i]<-->a[i-1];change=1;}

low++; //修改下界

}//**while**

}//BubbleSort2

[算法讨论]题目中“向上移”理解为向序列的右端，而“向下移”按向序列的左端来处理。

3．**typedef** **struct**

{ **int** num; **float** score; }RecType;

**void** SelectSort(RecType R[51]，**int** n)

{ **for**(i=1; i<n; i++)

{ //选择第i大的记录，并交换到位

k=i; //假定第i个元素的关键字最大

**for**(j=i+1;j<=n;j++) //找最大元素的下标

**if**(R[j].score>R[k].score) k=j;

**if**(i!=k) R[i] <-->R[k]; //与第i个记录交换

}//**for**

**for**(i=1; i<=n; i++) //输出成绩

{ printf("%d,%f",R[i].num,R[i].score); **if**(i%10==0) printf("\n");}

}//SelectSort

4．解答：分析：此问题分为两个算法，第一个算法 shift 假设 a[ 1] , a[ 2 ],…,a[ k]为堆，增加一个无素a[ k + 1 ],把数组a[ 1 ],a[ 2 ], …,a[ k + 1 ]调整为堆。第二个算法heep 从1开始调用算法sift将整个数组调整为堆。

Void sift (datatype A[ n ] , int K) /\* n > = k + 1 \*/

{ x = A[ K+1] ;

i = K +1 ;

while ( ( i/2 > 0 )&&( A[i/2]>x) ) { A[i]= A[i./2]; i = i/2;} /\*从下往上插入位置 \*/

A[i] = x ;

}

Void heap ( datatype A[ n ] ) ; /\* 从1开始调用算法sift ，将整个数组调整为堆 \*/

{ for ( k = 1 ; k <= n-1; k++ ) sift ( A,k ) ; }