**《数据结构与算法》作业**

# 习题6 排序

6-1 如果顺序表中的大部分数据元素按关键字值递增有序，则采用( D )算法进行升序排序，比较次数最少。

(A) 快速排序

(B) 归并排序

(C) 选择排序

(D) 插入排序

6-2 若一组记录的关键码为46, 79, 56, 38, 40, 84，则利用快速排序方法且以第一个记录为基准，得到的一次划分结果为( C )。

(A) 38，40，46，56，79，84

(B) 40，38，46，79，56，84

(C) 40，38，46，56，79，84

(D) 40，38，46，84，56，79

6-3 对数据36，12，57，86，9，25进行排序，如果前三趟的排序结果如下：

第1趟：12，36，57，9，25，86

第2趟：12，36，9，25，57，86

第3趟：12，9，25，36，57，86

则采用的排序方法是( B )。

(A) 插入排序

(B) 起泡排序

(C) 归并排序

(D) 快速排序

6-4设int r[9]={0, 25, 28, 13, 33, 56, 47, 19, 40}; 则调用F(r, 1, 8)之后，数组r[ ]中的数据元素存放顺序是( D )。

F(int r[ ], int s, int t)

{

for (int j=2\*s; j≤t; j\*=2)

{

if (r[j+1]<r[j]) ++j;

if (r[s]≤r[j]) break;

int x=r[s];

r[s]=r[j];

r[j]=x;

s = j;

}

}

(A) 0, 13, 19, 25, 28, 33, 40, 47, 56

(B) 56, 47, 40, 33, 28, 25, 19, 13, 0

(C) 0, 25, 28, 13, 33, 40, 47, 19, 56

(D) 0, 13, 28, 19, 33, 56, 47, 25, 40

6-5 在链式基数排序中，对关键字序列369, 367, 167, 239, 237, 138, 230, 139进行第1趟分配和收集后，得到的结果是( C )。

(A) 167, 138, 139, 239, 237, 230, 369, 367

(B) 239, 237, 138, 230, 139, 369, 367, 167

(C) 230, 367, 167, 237, 138, 369, 239, 139

(D) 138, 139, 167, 230, 237, 239, 367, 369

6-6 设int r[7]={5，2，6，4，1，7，3}; 则执行for ( i=0; i<7; i++) r[r[i]-1]=r[i]; 命令之后，数组r[7]中的数据元素存放顺序是( D )。

(A) 5，2，7，4，1，6，3

(B) 3，2，1，4，5，7，6

(C) 1，2，3，4，5，6，7

(D) A、B、C都不对

6-7 设计一种排序算法，对1000个[0, 10000]之间的各不相同的整数进行排序，要求比较次数和移动次数尽可能少。

使用计数排序

void sort(int r[])

{

int a[10000],j=0;

for(int i=0;i<1000;i++)

{

a[r[i]-1]=r[i];

}

for(int i=0;i<10000;i++)

{

if(a[i]>0)

r[j++]=a[i];

}

}

6-8 设顺序表的结点结构为(Type Key; int Next)，其中，Key为关键字，Next为链表指针。试设计静态链表排序算法。

typedef struct

{

Type Key;

int Next;

}Record;

void sort(Record r[],int n)

{

for(int i=1;i<n;i++)

{

int p=r[0].Next;

Recode temp=r[p];

r[p]=r[i];

r[i]=temp;

if(r[i].next!=i)

r[0].next=r[i].next;

for(int j=i+1;j<=n;j++)

{

if(r[j].next==i)

{

r[j].next=p;

break;

}

}

}

}

6-9 假设n个部门名称的基本数据存储在字符数组name[N][31]中，0≤n≤N≤20。试设计一个起泡排序算法，将n个部门名称按字典序重新排列顺序。

int name\_ascii(char \*\*name,int j) //计算每个部门名称对应的ascii码值

{

int sum=0;

for(k=0;k<20;k++)

{

sum=sum+name[j][k];

}

return sum;

}

void sort(char \*\* name, int n)

{

int i=0;

While(n>=0)

{

i= -1;

for(j=0;j<n-1;j++)

{

if(name\_ascii(name,j)>name\_ascii(name,j+1))

{

for(int k=0;k<20;k++)

{

char temp=name[j][k];

name[j][k]=name[j+1][k];

name[j+1][k]=temp;

}

i=j;

}

}

n=i;

}

}

6-10 假设采用链表存储类型：

typedef struct RNode

{

int key; //数据域(也是关键字域)

struct RNode \*next; //指针域

} RNode, \*RList;

typedef RList R[N]; //链表类型, 常变量N≥n

又设R[1..n]是[10, 999]之间的随机整数。试设计一个链表基数排序算法，将R[n]中的数从小到大排序。排序结果仍存放在R[n]中。

void radix\_sort(R[N])

{

Initiate(&R[0]);//初始化R[0],并将所有数据存放在R[0]中

for(int i=1;i<=n;i++)

{

Insert(R[0],R[i]->key);

}

RNode \*p[10],\*q;

int i , j , k;

for(j=1;j<=3;j++)

{

//十个头结点初始化

for(i=0;i<10;i++)

{

Initiate(&p[i]);

}

while(R[0]->next!=NULL)

{

q=PopFront(R[0]); //取首结点并删除该结点

k=GetNum(q , j); //取得第j位的元素值k

Pushback(p[k],q); //将该节点连接到10个链表相应的位置

}

//将10个链表从0-9依次连接到R[0]后面

for(i=0;i<10;i++)

{

RNode \*tail=R[0];

while(tail->next!=NULL)

{

tail=tail->next;

}

tail->next=p[i]->next;

}

}

//整理完了以后数据都存放在R[0]中，再将数据分配到R[1]-R[n]

RNode \* cur=R[0]->next;

int m=1;

While(cur!=NULL)

{

R[m++]->key=cur->key;

cur=cur->next;

}

}

6-11 在下列排序算法中，时间复杂度最好的是( A )。

(A) 堆排序

(B) 插入排序

(C) 起泡排序

(D) 选择排序

6-12 根据建堆算法，将关键字序列5，7，10，8，6，4调整成一个大顶堆，最少的交换次数为( B )。

(A) 1

(B) 2

(C) 3

(D) 4

6-13 给定关键字序列503, 87, 512, 61, 908, 170, 897, 275, 653, 426。

(1)以第一个关键字为枢轴，给出第1趟快速排序后的关键字序列;

(2)给出根据堆排序算法建立的小顶堆序列。

(1) 87,61,170,275,426,503,512,908,897,653

(2)61,87,170,275,426,512,897,503,653,908

6-14 设计基于顺序表存储结构的树形选择排序算法。

#define INF 9999999

void sort(int T[], int n)

{

for (int i = 1; i <= n; i++)

{

printf("%d", T[1]); //输出根节点

int j = 1;

//将底层结点设置成无穷

while (T[2 \* j] == T[1] || T[2 \* j + 1] == T[1])

{

j \*= 2;

if (T[j] != T[1])

j++;

}

T[j] = INF;

//将第i+1小的数据元素调整到根节点

for (int k = j; k > 0; k /= 2)

{

if (k % 2)

j = T[k - 1];

else

j = T[k + 1];

if (j < T[k])

T[k / 2] = j;

else

T[k / 2] = T[k];

}

}

}