# 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。

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。

void Arrange(Record \*r,int n)

{

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

{

r[0].next;

Record 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)

continue;

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。

#include <iostream>

#include <cstring>

#include <stdlib.h>

#include <time.h>

#define SIZE 10

using namespace std;

char name[25][25]={0};

void Names(char A[][25],int n)

{

srand(time(NULL));

int i,j,k;

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

{

k=2\*(rand()%10+3); //部门字数

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

A[i][j]=rand()%30+176; //汉字区

A[i][j]='\0';

}

}

void Namesort(char a[][25],int n)

{

for(int i=n-1;i>0;i--)

{

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

{

if(strcmp(a[j],a[j+1])>0)

{

char temp[25]={0};

strcpy(temp,a[j]);

strcpy(a[j],a[j+1]);

strcpy(a[j+1],temp);

}

}

}

return ;

}

void Output(char A[][25],int n)

{

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

{

cout<<A[i]<<endl;

}

return ;

}

int main()

{

Names(name,SIZE);

cout<<"最开始的初始化:"<<endl;

Output(name,SIZE);

Namesort(name,SIZE);

cout<<endl;

cout<<"排序之后:"<<endl;

Output(name,SIZE);

return 0;

}

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。

#include <iostream>

using namespace std;

typedef struct RNode{

int key;

int num;

struct RNode \*next;

}RNode,\*RLink;

typedef struct{

RLink head;

RLink tail;

}List;

int main()

{

int n=1000;

List L[10];

RLink p=new RNode;

p->next=NULL;

for(int i=0;i<=9;i++)

{

L[i].head=NULL;

L[i].tail=NULL;

}

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

{

int t;

cin>>t;

RLink q=new RNode;

q->key=t;

q->num=t;

q->next=p->next;

p->next=q;

}

RLink p1=p->next;

for(int i=1;i<=3;i++)

{

while(p1)

{

RLink q=p1;

RLink t=new RNode;

t->next=NULL;

t->key=q->key;

int left=q->num%10;

t->num=q->num/10;

if(L[left].head==NULL)

L[left].head=t;

else

L[left].tail->next=t;

L[left].tail=t;

p1=p1->next;

}

int j=0;

while(L[j].head==NULL)

j++;

p1=L[j].head;

for(int k=j+1;k<=9;k++)

{

if(L[k].head==NULL)

continue;

L[j].tail->next=L[k].head;

j=k;

}

for(int i=0;i<=9;i++)

{

L[i].head=NULL;

L[i].tail=NULL;

}

}

while(p1)

{

cout<<p1->key<<endl;

p1=p1->next;

}

return 0;

}

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;

426 87 275 61 170 503 897 908 653 512

1. 给出根据堆排序算法建立的小顶堆序列。

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。

#include <iostream>

#include <math.h>

#include <algorithm>

#define SIZE 1000

#define inf 2147483647

using namespace std;

int a[SIZE+10] = { 0 };

void Init()

{

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

{

cin >> a[i];

}

return;

}

int getmin(int a, int b)

{

return ((a < b) ? a : b);

}

int getheight(int x)

{

int k = 0;

while (pow(2, k) < x)

k++;

return k+1;

}

void TreeSort()

{

int height = getheight(SIZE);

int FullHeight = height - 1;

int full = pow(2, height - 1);

int b[SIZE\*4] = { 0 };

for (int i = 1; i <= full - 1; i++)

b[i] = inf;

for (int i = full; i <= full + SIZE - 1; i++)

b[i] = a[i- full + 1];

for (int i = full + SIZE; i <= pow(2, height) - 1; i++)

b[i] = inf;

for (int i = height - 1; i >= 1; i--)

for (int j = pow(2, i); j <= pow(2, i + 1) - 1; j += 2)

b[j / 2] = getmin(b[j], b[j + 1]);

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

{

a[i] = b[1];

int j = 1;

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

{

j \*= 2;

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

j++;

if (2 \* j >= 4 \* SIZE)

break;

}

b[j] = inf;

for (int k = j; k >= 1;k/=2 )

{

if (k % 2 == 0)

j = b[k + 1];

else

j = b[k - 1];

if (j < b[k])

b[k / 2] = j;

else

b[k / 2] = b[k];

}

}

return;

}

int main()

{

Init();

TreeSort();

return 0;

}