# 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( )算法进行升序排序，比较次数最少。D

(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。

答：

#include <stdio.h>

#include <stdlib.h>

// 获取数组中的最大值

int getMax(int arr[], int n) {

int max = arr[0];

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

if (arr[i] > max) {

max = arr[i];

}

}

return max;

}

// 基数排序主函数

void radixSort(int arr[], int n) {

int max = getMax(arr, n);

for (int exp = 1; max / exp > 0; exp \*= 10) {

int \*output = (int \*)malloc(n \* sizeof(int));

int count[10] = {0};

// 统计每个桶中的元素数量

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

count[(arr[i] / exp) % 10]++;

}

// 计算每个桶的累积计数

for (int i = 1; i < 10; i++) {

count[i] += count[i - 1];

}

// 将元素放入对应的桶中

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

int digit = (arr[i] / exp) % 10;

output[count[digit] - 1] = arr[i];

count[digit]--;

}

// 将排序后的结果复制回原数组

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

arr[i] = output[i];

}

free(output);

}

}

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。

答：#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 1000

typedef struct {

int key;

int next;

} SLinkList[MAXSIZE];

// 静态链表排序算法

void staticLinkListSort(SLinkList L) {

int i, j, k;

for (i = 0; L[i].next!= -1; i = L[i].next) {

k = i;

for (j = L[i].next; L[j].next!= -1; j = L[j].next) {

if (L[j].key < L[k].key) {

k = j;

}

}

if (k!= i) {

// 交换节点 i 和节点 k 的关键字

int temp = L[i].key;

L[i].key = L[k].key;

L[k].key = temp;

}

}

}}

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。

答：

#include <stdio.h>

#include <string.h>

#define N 20

#define MAX\_LEN 31

// 交换两个字符串

void swap(char a[], char b[]) {

char temp[MAX\_LEN];

strcpy(temp, a);

strcpy(a, b);

strcpy(b, temp);

}

// 起泡排序算法

void bubbleSortDepartmentNames(char name[][MAX\_LEN], int n) {

int i, j;

for (i = 0; i < n - 1; i++) {

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

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

swap(name[j], name[j + 1]);

}

}

}

}

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。

答：

#include <stdio.h>

#include <stdlib.h>

typedef struct RNode {

int key;

struct RNode \*next;

} RNode, \*RList;

typedef RList R[100]; // 假设一个足够大的N，这里设为100

// 获取链表中最大数的位数

int getMaxDigit(RList head) {

RList p = head;

int max = p->key;

while (p) {

if (p->key > max) {

max = p->key;

}

p = p->next;

}

int digit = 0;

while (max > 0) {

max /= 10;

digit++;

}

return digit;

}

// 从链表头部插入节点

void insertHead(RList \*head, int num) {

RList newNode = (RList)malloc(sizeof(RNode));

newNode->key = num;

newNode->next = \*head;

\*head = newNode;

}

// 释放链表内存

void freeList(RList head) {

RList p, temp;

p = head;

while (p) {

temp = p;

p = p->next;

free(temp);

}

}

// 链表基数排序

void radixSort(RList R[], int n) {

int digit = getMaxDigit(R[0]);

int exp = 1;

RList bucket[10] = { NULL };

RList tail[10] = { NULL };

for (int d = 0; d < digit; d++) {

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

RList p = R[i];

while (p) {

int index = (p->key / exp) % 10;

if (!bucket[index]) {

bucket[index] = p;

tail[index] = p;

p = p->next;

tail[index]->next = NULL;

} else {

tail[index]->next = p;

tail[index] = p;

p = p->next;

tail[index]->next = NULL;

}

}

}

int j = 0;

for (int k = 0; k < 10; k++) {

if (bucket[k]) {

RList p = bucket[k];

while (p) {

insertHead(&R[j], p->key);

RList temp = p;

p = p->next;

free(temp);

}

bucket[k] = NULL;

tail[k] = NULL;

j++;

}

}

exp \*= 10;

}

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

freeList(R[i]);

R[i] = NULL;

}

}

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( C )。

(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。

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

（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。

答：

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 100

// 交换两个元素

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// 找到最小元素的索引

int findMinIndex(int arr[], int left, int right) {

int minIndex = left;

for (int i = left + 1; i < right; i++) {

if (arr[i] < arr[minIndex]) {

minIndex = i;

}

}

return minIndex;

}

// 树形选择排序算法

void treeSelectSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

int minIndex = findMinIndex(arr, i, n);

swap(&arr[i], &arr[minIndex]);

}

}