1. 直接插入排序算法在含有n 个元素的初始数据正序、反序和数据全部相等时时间复杂度各是多少？

答：直接插入排序算法初始数据正序时，直接按照顺序插入，时间复杂度为O(n)

初始数据反序时，需要将已插入数据后移，时间复杂度为O(n2)

初始数据全部相等时，与正序情况相同，时间复杂度为O(n)

2. 回答以下关于直接插入排序和折半插入排序的问题：

(1) 使用折半插入排序所要进行的关键字比较次数是否与待排序的元素的初始状态有关？

(2) 在一些特殊情况下，折半插入排序比直接插入排序要执行更多的关键字比较，这句话对吗？

答：

1. 无关。因为初始数据是否有序，都需要从数据中部先开始进行判定，然后查找插入位置，其查找次数是固定的。
2. 正确，例如在数据都为正序的情况下，折半插入排序比直接插入排序要执行更多的关键字比较。

4. 在实现快速排序的非递归算法时，可根据基准元素将待排序序列划分为两个子序列。若下一趟首先对较短的子序列进行排序，试证明在此做法下快速排序所需要的栈的深度为O( log­2n ) 。

答：根据快速排序的算法特点可知，其递栈的深度与将数据划分的次数有关。

每次划分都是将数据根据基准数据划分为左右两部分，然后再将这两个部分按照相同的规则处理，设S(n)为对n个记录进行快速排序时平均所需栈的深度，则有

表格

低可信度描述已自动生成

当n=1时，，所需栈空间为常量，由此可推出：S(n)=O(log­2n)。

6. 在堆排序、快速排序和二路归并排序中：

（1）若只从存储空间考虑，则应首先选取哪种排序方法，其次选取哪种排序方法，最

后选取哪种排序方法？

（2）若只从排序结果的稳定性考虑，则应选取哪种排序方法？

（3）若只从最坏情况下的排序时间考虑，则不应选取哪种排序方法？

答：

（1）三种算法的空间复杂度如下：堆排序O(1)、快速排序O(nlog2n)、二路归并O(n)，故按照堆排序、快速排序、二路归并的顺序选择。

（2）三种算法的稳定度如下：堆排序不稳定、快速排序不稳定、二路归并稳定，故选择二路归并。

（3）三种算法的时间复杂度最坏情况如下：堆排序O(nlog2n)、快速排序O(n­2)、二路归并O(nlog2n)，故不应选择快速排序。

8. 基数排序过程中用队列暂存排序的元素，是否可以用栈来代替队列？为什么？

答：不可以，因为基数排序不允许迫害已经排好的队列，如果使用栈来代替，会导致其顺序发生改变，使得整个过程变得不稳定。

11. 设计一个双向冒泡排序的算法，即在排序过程中交替改变扫描方向。

代码如下：

void DBubbleSort(int a[], int n) {

int i = 0, j;

int temp;

bool flag = true;

while (flag) {

for (j = n - 1 - i;j > i;j--) {

if (a[j] < a[j - 1]) //由后向前冒泡小元素

{

flag = true;

temp = a[j];a[j] = a[j - 1];a[j - 1] = temp;

}

}

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

if (a[j] > a[j + 1]) //由前向后冒泡小元素

{

flag = true;

temp = a[j];a[j] = a[j + 1];a[j + 1] = temp;

}

}

if (!flag) return;

i++;

}

}

13. 对于给定的含有n元素的无序数据序列（所有元素的关键字不相同），利用快速排序方法求这个序列中第k（1≤k≤n）小元素的关键字，并分析所设计算法的最好和平均时间复杂度。

int QuickSelect(int a[], int s, int t, int k) {//在a[s..t]序列中找第k小的元素

int i = s, j = k;

int temp;

if (s < t) {

temp = a[s];

while (i != j) {//从两端向中间扫描,直至i==j为止

while (j > i&& a[i] >= temp)

i++;//从右向左扫描,找第1个关键字小于tmp的a[j]

a[i] = a[j];//将a[j]前移到a[i]的位置

while (j < i&& a[i] <= temp)

i++;//从左向右扫描,找第1个关键字大于tmp的a[i]

a[j] = a[i];//将a[i]后移到a[j]的位置

}

a[i] = temp;

if (k - 1 == i) return a[i];

else if(k-1<i) return QuickSelect(a, s, i - 1, k);//在左区间中递归查找

else return QuickSelect(a, i + 1,t, k);//在右区间中递归查找

}

else if (s == t && s == k - 1) //区间内只有一个元素且为R[k-1]

return a[k - 1];

else

return -1; //k错误返回特殊值-1

}

本算法的比较次数为T(n)=T(n/2)+O(n)，最好情况为每次划分的基准均为中位数，比较次数为T(n)=O(n)；最坏情况为每次划分的基准均为最大值或最小值，则处理区间只比上次少一个元素，故比较次数为T(n)=O(n2)。

14. 设n个记录R[0..n-1]的关键字只取3个值：0，1，2。采用基数排序方法将这n个记录排序。并用相关数据进行测试。

代码如下：

#include "sqlist.cpp" //顺序表基本运算算法

#include <malloc.h>

#define Max 3

#define MAXL 10

typedef struct {

int key;

char data;

}RecType;

typedef struct node

{

RecType Rec;

struct node\* next;

} NodeType;

void RadixSort1(RecType R[], int n)

{

NodeType\* head[Max], \* tail[Max], \* p, \* t; //定义各链队的首尾指针

int i, k;

for (i = 0;i < Max;i++) //初始化各链队首、尾指针

head[i] = tail[i] = NULL;

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

{

p = (NodeType\*)malloc(sizeof(NodeType));//创建新节点

p->Rec = R[i];

p->next = NULL;

k = R[i].key; //找第k个链队,k=0,1或2

if (head[k] == NULL) //进行分配,采用前插法建表

{

head[k] = p; tail[k] = p;

}

else

{

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

}

}

p = NULL;

for (i = 0;i < Max;i++) //对于每一个链队进行循环收集

if (head[i] != NULL) //产生以p为首节点指针的单链表

{

if (p == NULL)

{

p = head[i]; t = tail[i];

}

else

{

t->next = head[i]; t = tail[i];

}

}

i = 0;

while (p != NULL) //将排序后的结果放到R[]数组中

{

R[i++] = p->Rec;

p = p->next;

}

}

int main()

{

int i, n = 5;

RecType R[MAXL] = { {1,'A'},{0,'B'},{0,'C'},{2,'D'},{1,'F'} };

printf("排序前:\n ");

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

printf("[%d,%c] ", R[i].key, R[i].data);

printf("\n");

RadixSort1(R, n);

printf("排序后:\n ");

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

printf("[%d,%c] ", R[i].key, R[i].data);

printf("\n");

return 1;

}

结果如下：

文本

描述已自动生成

1. 外排序中两个相对独立的阶段是什么？

答：产生初始归并段和多路归并排序。

2. 给出一组关键字T=（12，2，16，30，8，28，4，10，20，6，18），设内存工作区可容纳4个记录，给出用置换-选择排序算法得到的全部初始归并段。

表格

描述已自动生成

3.设输入的关键字满足k1>k2>…>kn，缓冲区大小为m，用置换-选择排序方法可产生多少个初始归并段？

答：可产生个初始归并段。设记录Ri的关键字为ki（1≤i≤n），先读入m个记录R1、R2、…、Rm，采用败者树选择最小记录Rm，将其输出到归并段1，Rmin=km，在该位置上读入Rm+1，采用败者树选择最小记录Rm-1，将其输出到归并段1，Rmin=km-1，在该位置上读入Rm+2，采用败者树选择最小记录Rm－2，将其输出到到归并段1，Rmin=km-2，…，以此类推，产生归并段1：（Rm，Rm-1， …，R1）。同样产生其他归并段（R2m，R2m-1， …，Rm+1），（R3m，R3m-1， …，R2m+1），…，可产生个初始归并段。