Computer System Networking and Telecommunications
计算机系统网络和电信，2019,1(2)
ISSN:2661-3719 (Print); 2661-3727 (Online)

A Brief Analysis of the Sorting Mode and Comparison in the Data Structure

Chengxia ZHANG Zhengde BAO Yawen TANG
School of Computer and Software, Jincheng College, Sichuan University, Chengdu,611731

Abstract

In order to find the target information more quickly, it is more beneficial to improve the working efficiency, to know the sorting algorithm and to get the characteristics of the sorting algorithm. In this paper, based on the related knowledge of space complexity, time complexity, stability and so on, the characteristics and application of different sorting algorithms are compared by using the example method, and some help is provided for optimizing the query mode.

Key Words

Data Structure, Sorting, Time Complexity, Spatial Complexity, Algorithm

DOI:10.18686/jsjxt.v1i2.712

浅析数据结构中排序方式及比较

张承霞 鲍正德 唐娅雯
四川大学锦城学院计算机与软件学院，四川，成都，611731

摘 要

如果按某种特定的顺序关系来排序存储则能更加快速地查找出目标信息，因此选择正确的排序方式更有利于提高工作效率，了解排序算法以及获取其特点对于选择排序方式有着重要作用。本文将基于空间复杂度、时间复杂度、稳定性等相关知识，运用举例的方法为不同排序算法的特点及适用情况做了比较，为优化查询方式提供部分帮助。

关键词

数据结构；排序；时间复杂度；空间复杂度；算法

1.引言

跟着信息化时代发展的脚步，人们可以通过越来越多的数据获得有价值的信息，那依照哪一种次序关系来排序或存储信息可以在一定程度上提高工作效力呢。所以了解排序的算法和特点对于选择排序方式有着重要作用，本文对部分排序方式进行了介绍及比较。

2.排序简介

如一个四十个学生的班级要对学生按照成绩进行排序，他们的成绩就可以作为他们的关键字，按照成绩的高低就可以对他们进行排序，这个让数据中的关键字按照次序，来对某组数据做排列组合的过程称为排序。在计算机运用中，排序算法是特别有用的，如用户数据的存储等，若要排序的数据量小，则可以在内存中排序，这类排序分为内部排序，相反的，要借用存储器来执行排序的数据量较大、不能在内存中执行排序的方式分为外部排序。

3.算法介绍

可以将排序算法分为插入排序（包含了直接插入、折半插入、希尔排序等方式）、交换排序（包含了冒泡、快速排序）、选择排序（包含了简单选择、堆、树形选择排序的排序方法）、归并排序等[1]。

3.1 直接插入排序

直接插入排序是每次从要排序的表中拿一个数据出来，将这个数据的关键字和已经排好序的表里面的元

素逐一来对比，再按照顺序插入到合适的位置，直至记录全都插到有序表中，就获取一个排好序的序列了。

几个同学要按成绩从低到高排，先找个同学站到一边，再找个同学与他比较，如果成绩比他高就站他后面，反之则站他前面，下一个同学分别与他们两个同学相比较，选择合适的位置站，此后每个同学都与这个排好队的同学一一比较，选择合适的位置，直到所有同学都站完。

具体实现：

```
void InsertSort(int elem[],int len){
int i,j,a,b;
for(i=2;i<=len;++i){
    if(elem[i]<elem[i-1]){
        elem[0]=elem[i];//第一个位置作为哨岗
        elem[i]=elem[i-1];
        for(j=i-2;elem[0]<elem[j];--j){
            elem[j+1]=elem[j];
        }
        elem[j+1]=elem[0];
    }
}
}
```

特点：①平均移动 n(n-1)/2 次。时间复杂度为 $O(n^2)$。②空间复杂度为 O(1)。③是稳定排序。

3.2 折半插入排序

折半插入排序则不是让待排序数据的关键字一一与有序表中的元素一一来对比，而是采取半数的方法选择元素与关键字来对比。

如有 6 个同学已经按照从矮到高的顺序站好队了，要排进这个队伍的同学先与第 3 个同学比，如果他比第三个同学高，那他再和第 5 个同学比，如果他比第 5 个同学矮，接着与第 4 个同学比，最后找出合适的位置。用折半插入排序对比三次就可以找到对应位置，用直接插入排序最少要对比 3 次。

具体实现：

```
void BInsertSort(int elem[],int len){
int i,j,low,high,m;
for(i=2;i<len;++i){
    elem[0]=elem[i];
    low=1;high=i-1;
    while(low<=high){
        m=(low+high)/2;
        if(elem[0]<elem[m])
        high=m-1;
        else
        low=m+1;
    }
    for(j=i-1;j>=high+1;--j)
    elem[j+1]=elem[j];
    elem[high+1]=elem[0];
}
}
```

特点：①与直接插入对比，消减了对比的次数，但挪动次数没有改变，时间复杂度为 $O(n^2)$。②空间复杂度为 O(1)。③是稳定排序

3.3 希尔排序

希尔排序是直接插入的另一种改良方式，不是让相邻的数据相比较，而是用增量，增量要小于数据数，间隔为增量的倍数的数据都放在一组，然后对这组数据执行插入排序，接着用一个小于第一个增量的增量，继续执行分组和插入排序，结尾的增量必为 1。

如有数列{21，25，48，26，16，8}进行希尔排序

第一趟（增量选 3）：21 与 26 不交换，25 与 16 交换，48 与 8 交换。

21，16，8，26，25，48

第二趟（增量选 2）：21 与 8 与 25 要交换，16 与 26 与 48 不交换。

8，16，21，26，25，48

第三趟（增量选 1）：对 8，16，21，26，25，48 进行直接插入排序

8，16，21，25，26，48

特点：①在增量不同时，数据的挪动也不同，所以这种方式的时间复杂度目前还未完全清楚，直到现在只得出在 $O(n^{1.25})$~$O(1.6n^{1.25})$范围之间。 ②空间复杂度为 O(1)。③是不稳定排序。

3.4 冒泡排序

冒泡排序是用第一个数据的关键字和第二个关键字相对比，假如不是正确次序就互换位置，是就不互换，接着将第二个和第三个关键字相对比，让两两相邻的关

键字来对比，对比一趟能明确最后一个关键字的位置。第二趟对除那个关键字以外的进行排序，直到在某趟中没有产生互换为止。

如有数列{6，4，10，8，3}进行冒泡排序

第一趟：6 与 4 交换　4，6，10，8，3

6 与 10 不换 4，6，10，8，3

10 与 8 交换 4，6，8，10，3

10 与 3 交换 4，6，8，3，10

第二趟：4，6，3，8，10

第三趟：4，3，6，8，10

第四趟：3，4，6，8，10

具体实现：

```
void BubbleSort(int elem[],int n){
    int j,flag,m,t;
    flag=1;m=n-1;
    while((m>0)&&(flag==1)){
        flag=0;
        for(j=0;j<=m;j++)
        if(elem[j]>elem[j+1]){//如果该关键字大于后面相邻关键字，就交换
            flag=1;
            t=elem[j];
            elem[j]=elem[j+1];
            elem[j+1]=t;
        }
        --m;
    }
}
```

特点：①最好的情况是要排序的记录原本的顺序满足要求，只需要比较一趟，总共 n-1 次，时间复杂度是 O(n)；最坏的情况是要排序的记录原本的顺序与要求的顺序刚好相反，总共要比较 n(n-1)/2 次，此时时间复杂度为 $O(n^2)$。②空间复杂度为 O(1)。③是稳定排序。

3.5 快速排序

快速排序是从待排序的数据中随便取一个作为中间值（一般选择第一个记录的关键字），将比这个中间值大的数据放一边，比中间值小的数据放另一边，然后再对这两组数据重复以上步骤，直到剩下比中间值大或小的数据数目至多有一个为止[2]。

如有数列{28，7，40，2，63，9，58，15，48，20}进行快速排序

第一趟：9，7，20，2，15，28，58，63，48，40

第二趟：2，7，9，20，15，28，58，63，48，40

第三趟：2，7，9，20，15，28，58，63，48，40

第四趟：2，7，9，15，20，28，58，63，48，40

第五趟：2，7，9，15，20，28，48，40，58，63

第六趟：2，7，9，15，20，28，40，48，58，63

特点：①当经快速排序得到的二叉排序树（若孩子不是空的，左孩子上数据都小于根，右孩子上数据都大于根）为一棵平衡二叉树，时间复杂度最少为 $O(n\log_2 n)$，当生成的二叉排序树是棵只有一个子树的二叉排序树时，时间复杂度最多为 $O(n^2)$[3]。②在不同的情况下需要的辅助空间数量不同，空间复杂度最差是 O(n)，最佳是 $O(\log_2 n)$。③是不稳定排序。

3.6 简单选择排序

简单排序是在要排序的数据中来对比，找出最值数据与发端（或末端）的数据来互换，接着在除发端（或末端）的数据中来互换，选出最值数据，再互换位置，直到只剩一个数据为止。

如有数列{11，6，18，12，15，9，10，22}进行简单选择排序

第一趟扫描 8 次，6 最小，与 11 交换：6，11，18，12，15，9，10，22

第二趟扫描 7 次，9 最小，与 11 交换：6，9，18，12，15，11，10，22

第三趟扫描 6 次，10 最小，与 18 交换：6，9，10，12，15，11，18，22

第四趟扫描 5 次，11 最小，与 12 交换：6，9，10，11，15，12，18，22

…

具体实现：

```
void SelectSort(int elem[],int n){
int i,j,k,t;
for(i=0;i<n;++i){
    k=i;
    for(j=i+1;j<=n;++j)
    if(elem[j]<elem[k])
    k=j;
    if(k!=i){
```

```
            t=elem[i];
            elem[i]=elem[k];
            elem[k]=t;
        }
    }
    }
```

特点：①最好的情况是初态顺序满足要求，不移动；最坏的状况是初态顺序刚好相反，需要挪动 3(n-1)次，但在这两种情形的比较次数都相同，时间复杂度是 $O(n^2)$。②空间复杂度为 O(1)。③是不稳定排序。

3.7 堆排序

堆排序是将需要排序的记录看作顺序二叉树，再将其转换为堆，首先把所有记录创建为大根堆或小根堆，然后把堆顶这个最大或最小的记录拿出来，接着调节堆，再把堆顶中的记录拿出来，直到所有记录被取完为止。建立堆：将全部数据依照次序存储方法建为堆（看做顺序二叉树），假定这颗二叉树有 n 个数据元素，末端分支结点的序号为[n/2]，从[n/2]选为双亲，与其孩子相对比，最大的与双亲互换，直到根结点选为双亲来对比调度完成，着重关注经由互换的结点要与子结点对比，再调度。调整堆：把栈顶中的数据拿出后，把末端数据放入堆顶，从[n/2]选为父结点，与子结点相对比，最大的与父结点互换，再调度，直到根结点选为父结点来对比调度完成。

特点：①堆排序中对比的频率是开始创建堆加上调节堆的对比频率，创建时的时间复杂度为 O(n)，n-1 次调整重新建时的比较次数不超过 2n，所以时间复杂度为 $O(n\log_2 n)$。②空间复杂度为 O(1)。③是不稳定排序。

3.8 归并排序

合并排序是把已排好序的线性表聚合在一起。将需要排序的记录看作 n 个只含有 1 个元素并且已经排好序的表，再把这些表两两合并，合并成 n/2 个含有 2 个元素并且排好序的表，接着将这些表继续两两合并，若有单数则留到下轮进行合并，直到合并成 1 个表。

如有数列{3，6，18，12，15，9，10，22}使用归并排序

第一趟后：[3，6]，[12，18]，[9，15]，[10，22]

第二趟后：[3，6，12，18]，[9，10，15，22]

第三趟后：[3，6，9，10，12，15，18，22]

特点：①要合并[log2n]趟，而且每趟的对比频率都小于 n，挪动频率全为 n，所以时间复杂度为 $O(n\log_2 n)$。②空间复杂度为 O(n)。③是稳定排序。

4.算法比较

4.1 计算时间

评价排序算法的性能之一——计算时间，计算时间由对比次数和挪动次数决定，通过以上排序方法的分析与介绍，我们可以了解到直接插入排序最少的挪动频率是 0，比较 n 次，最多移动 n^2 次，比较 n^2 次；折半插入排序要对比的频率是 $n\log_2 n$，最少的挪动频率是 0，最多挪动频率是 n^2；希尔排序要对比的频率是 $n\log_2 n$，挪动的频率 $n\log_2 n$；冒泡排序最少挪动频率是 0，对比频率是 n，最多挪动频率是 n^2，对比频率是 n^2；快速排序最少挪动频率是 $n\log_2 n$，对比频率是 $n\log_2 n$，最多挪动频率是 n^2，最多对比频率是 n^2；简单选择排序的对比频率是 n^2，起码的挪动频率是 0，最多挪动频率是 n^2；堆排序的挪动频率是 $n\log_2 n$，对比频率是 $n\log_2 n$；归并排序的挪动频率是 $n\log_2 n$，对比频率是 $n\log_2 n$。

4.2 存储空间

存储空间指的是在排序的过程中要使用到的辅助空间的数量，如两个记录交换需要一个辅助空间来进行存储。通过以上排序方法的分析，我们可以发现直接插入、折半插入、希尔、冒泡、简单选择以及堆排序都要一个存储空间来互换。快速排序在不同的情况下使用到的辅助空间数量不同，最少时空间复杂度是 $O(\log_2 n)$，最多时空间复杂度是 O(n)[4]。有多少条数据要排序，归并排序就要多少个空间来暂时存储，因而空间复杂度是 O(n)[5]。

4.3 稳定性

排序方式是不是稳固的，这依照数据中的关键字在排序先后的位置来确定的。比如十个同学按照身高进行排列，小明和小红一样高，在排列前小明在小红前面，在排列后小明仍然站小红前面，那这样排序的方式就是稳定的，否则只要一组关键字不满足这个要求就是不稳定的排序。直接插入的方式、折半插入的方式、冒泡的方式、合并的排序方式都是排列之前和之后关键字一样位置不会发生变化的稳固排序方法，而希尔、简单选择、

堆、快速排序方式都是关键字一样位置会发生变化的不稳固排序方法。

5.结束语

通过对以上不同的排序方法的比较和分析，我们可以得出①在需要排序的数据数 n 比较大的时候，优先考虑使用先进的排列的方式，若记录是无序的，则选用快速排序最好，此时使用的平均计算时间最少。②在需要排序的数据数 n 比较多的时候，稳定性最好的排列方式是归并。③在需要排序的数据数 n 比较多的时候，如果关键字有序，不规定稳定性，那么使用堆排序最好。④当数据数目 n 较小时，若要排序的数据有序，那用直接插入或冒泡的方式更好。

参考文献

[1]严蔚敏.数据结构 C 语言版[M].清华大学出版社，2007.
[2]项丽萍.数据结构中各种排序算法的比较与分析[J].太原城市职业技术学院学报,2008(12):154-155..
[3]乔欣.数据结构中典型排序算法性能分析[J].电脑知识与技术,2003(35):17-20.
[4]陆勤佳. 紫外—可见光吸收光谱检测系统的设计与实现[D].杭州电子科技大学,2017.
[5]姜忠华,徐文丽,刘家文,朱豪杰.智能归并排序[J].电子设计工程,2011,19(21):53-55.

作者简介

第一作者：张承霞（1997-），女，汉，四川省成都市，本科，四川大学锦城学院，研究方向：C++编程。
第二作者（通讯作者）：鲍正德（1989-），男，汉，黑龙江哈尔滨，研究生，四川大学锦城学院，研究方向：电子商务。
第三作者：唐娅雯（1999-），女，汉，四川省资阳市，本科，四川大学锦城学院，研究方向：信息管理、J2EE