浅谈基于 C 语言的三种常用排序算法分析研究

李成奥 张桂花 周彦汐
四川大学锦城学院计算机与软件学院 四川 成都 611731

【摘 要】在计算机领域，问题的求解离不开算法的设计，算法是问题解决的核心，一个好的算法可以大大提高问题求解的时间和空间效率。在众多算法中，其中基础算法 -- 排序，占有非常重要的地位，在计算机领域有着广泛的应用。本文主要介绍三种不同类型的排序算法，分别是希尔排序、快速排序、堆排序，并对之进行深入的分析。通过算法思想、算法实现代码、算法实例对三种算法进行剖析，结合具体应用场景进行适当的优化和改进。

【关键词】C 语言；排序；算法

日常生活中我们常常会见识到许多地方会用到排序的思想，例如在高考排名中、在打牌整理时等等许多地方都会了解到排序。下面我们就基于 C 语言来分析三种常用的排序算法。

1 插入排序——以希尔排序为例

1.1 概述

将一组无序数进行排序，选择其中一个或几个关键字按照其对应大小分多趟进行排序，每趟排序一次，直到所有数据全部进行排序后为止。例如在生活中人们进行扑克牌斗地主，通常进行摸牌，并将摸出的牌插入到手中已有牌的序列当中，当牌摸完时，此时就得到了一组有序序列。在顺序结构的插入排序中，又以希尔排序最为常用。尤其当序列数量庞大时，希尔排序效果更明显。

1.2 希尔排序

将一组无序序列按照增量 d 进行分组进行排序（增量逐级递减，直到 d=1）。其算法核心代码如下：

```
for(i=d+1;i<=length;++i)
    if(r[i]<r[i-d])
{
    r[0]=r[i];
    for(j=i-d;j>0&&r[0]<r[j];j-=d)
            r[j+d]=r[j];
    r[j+d]=r[0];
}
```

特点：（1）不稳定；（2）只存在于顺序结构，不能用于链式结构；（3）增量是素数和 1，并且最后一个增量值只能是 1；（4）总的比较和移动次数都比直接插入排序要少，n 越大，效果越明显。所以适合初始记录无序、n 较大的情况。

1.3 算法分析

从时间复杂度上看：增量表示所取两数的相隔位数，当增量大于 1 时，元素并非逐个进行移动，而是进行跳跃移动。当增量为 1 时，序列排序已经基本有序。希尔排序法利用增量 d 简化排序的比较总次数，从而减低了算法的时间复杂度。若通过具体分析，则我们认为希尔排序是通过给定的增量来对序列进行排序，在数学上若要求出该算法的具体的时间复杂度还有待考究。现今已经出现了一些局部结论：（1）当增量序列为 $dt[k]=2^{t-k+1}-1$ 时，希尔排序算法的时间复杂度为 $O(n^{3/2})$，t 为排序的趟数，$1 \leqslant k \leqslant t \leqslant [\log_2(n+1)]$。还有人在大量实验的基础上推理出：当 n 在某个特定范围时，希尔排序所需的比较和移动次数约为 $n^{1.3}$，当 n--> ∞时，可减少到 $n(\log_2 n)^2$。[1]

从空间复杂度上看：该排序算法只需占用一个额外空间，所以空间复杂度为 O(1)。

1.4 算法实例

已知待排序记录的关键字序列为 {50，39，66，98，77，14，28，50’，56，05}，通过希尔排序法进行排序（增量选取为 5、3、1）。

过程示意图如下表所示：

表 1 希尔排序过程示意图

初始关键字	50	39	66	98	77	14	28	50’	56	05
比较并排序(增量 d=5)	50					14				
		39					28			
			66					50’		
				98					56	
					77					05
一趟排序结果	14	28	50’	56	05	50	39	66	98	77
比较并排序(增量 d=3)	14			56			39			77
		28			05			66		
			50’			50			98	
二趟排序结果	14	05	50’	39	28	50	56	66	98	77
三趟排序结果	05	14	28	39	50’	50	55	66	77	98

2 交换排序——以快速排序为例

2.1 概述

交换排序与上一种插入排序所不同的就是选择两两交换，直到整个序列满足所需顺序为止。在我们生活中大家应该已经见过冒泡排序的，本节主要介绍以冒泡排序为基础来进行改进的快速排序。

2.2 快速排序

快速排序相对于只对两两交换的冒泡排序而言大大提升了排序速率。在冒泡排序中，一次排序只能交换两个相邻的数据。如果通过两个不相邻的一次交换，就能消除多个不规则顺序，则便会大大加快排序效率。快速排序就是一次能消除多个不规则顺序的一种排序方法。其核心代码如下：

```
while(low<high)
{
    while(low<high&&L.r[high].key>=pivotkey)--high;
    L.r[low]=L.r[high];
    while(low<high&&L.r[low].key<=pivotkey)++low;
    L.r[high]=L.r[low];
}
```

特点：（1）不稳定；（2）更易适用于顺序结构，难适用于链式结构；（3）适用于无序序列，并且数据较大的情况。

2.3 算法分析

从时间复杂度上看：其平均情况下的时间复杂度为 $O(n\log_2 n)$;

从空间复杂度上看：最好情况的空间复杂度为 $O(\log_2 n)$, 最坏情况为 $O(n)$。[2]

2.4 算法实例

已知待排序记录的关键字序列为{50，39，66，98，77，14，28，50’}，请给出快速排序法进行排序的过程。（如图 1 所示）

初始关键字	pivotkey							
	50	39	66	98	77	14	28	50’
	low							high
一次交换	28	39	66	98	77	14	50	50’
	low	→	low				high	
二次交换	28	39	50	98	77	14	66	50’
			low			high	←high	
三次交换	28	39	14	98	77	50	66	50‘
			low →	low		high		
四次交换	28	39	14	50	77	98	66	50’
				low high	←	high		
完成一趟排序	28	39	14	50	77	98	66	50’
第一趟快速排序过程								

初始状态	{50	39	66	98	77	14	28	50’}
一趟排序	{28	39	14}	50	{77	98	66	50’}
二趟排序	{14}	28	{39}	50	{77	98	66	50’}
三趟排序	14	28	39	50	{50’	66}	77	{98}
四趟排序	14	28	39	50	50’	{66}	77	98
快速排序全过程								

图 1 快速排序示意图

3 选择排序——以堆排序为例

3.1 概述

选择排序就是在一组序列中选择最小最小关键字放在第二组序列最后，再从原序列中找出最小的一个关键字放在与第二组序列最后，不断重复此过程，直到原来序列中所有元素选择完毕。这样的过程称为选择排序，本小结将以堆排序为例来进行详细介绍。

3.2 堆排序

堆排序是基于树的形式来进行排序，首先通过筛选法调整堆，然后建立初堆，最终完成堆的排序。其核心代码如下：

```
// 调整堆
for(k=2*s;j<=m;j*=2)
{
if(j<m&&L.r[j].key<L.r[j+1].key) ++j;
if(rc.key>=L.r[j].key) break;
L.r[s]=L.r[j];s=j;
}
L.r[s]=rc;
```

特点：（1）不稳定；（2）只适用于顺序结构；（3）记录数少不建议使用，记录数多时较为高效。

3.3 算法分析

从时间复杂度上看：此种排序算法用时主要体现于建初堆和调整堆的反复筛选上。[3] 由于堆排序利用树的形式进行，设 n 个数据的树深度为 h，每个非终端结点都将与该结点的孩子进行比较。初建堆是比较总次数为 $\sum_{i=h-1}^{1} 2^{i-1}*2(h-1)$.

调整堆需要做 n-1 次筛选，一颗结点为 n 的完全二叉树深度为 $[\log_2 n]+1$，所以比较总次数不超过

2n([$\log_2$n])。

所以，时间复杂度为 O(n$\log_2$n)，且平均性能接近于最坏性能。

空间复杂度：仅需一个记录空间，所以空间复杂度为 O（1）。

3.4 算法实例

已知待排序记录的关键字序列为 {50，39，66，98，77，14，28，50’}，给出用堆排序方法进行排序的过程。（如图 2 所示）

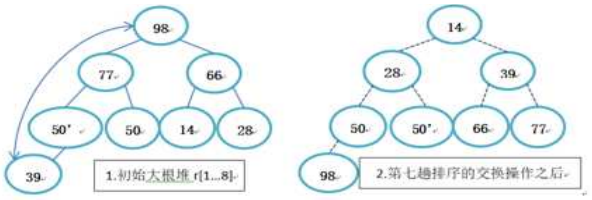

图 2 堆排序过程示意图（初始状态和最终状态）

4 小结

表 2 三种排序的比较

排序方法	时间复杂度			空间复杂度	稳定性
	最好情况	最坏情况	平均情况		
希尔排序			O(n1.3)	O(1)	不稳定
快速排序	O(nlog2n)	O(n2)	O(nlog2n)	O(log2n)	不稳定
堆排序	O(nlog2n)	O(nlog2n)	O(nlog2n)	O(1)	不稳定

通过以上分析可以得出，每种算法都有着自己的应用场景。在生活中、在大数据时代，面对种种繁杂且无序的序列，我们要合理运用高效且正确的算法。本文重点分析了三种不同算法的具体排序方法：插入排序——希尔排序、交换排序——快速排序、选择排序——堆排序。本文详细介绍了三种算法的具体作用，并进行实例分析，对以后相关方面的学习和研究起着重要作用。

【参考文献】

[1] 严蔚敏 , 李冬梅 , 吴伟民 . 数据结构 (C 语言版)[J]. 计算机教育 ,2012,No.168,66.

[2] 章程 .C 语言编程中排序算法的研究 [J]. 计算机产品与流通 ,2019(04):33.

[3] 贾丹 , 张兴 . 排序算法的性能分析 [J]. 电脑知识与技术 ,2015,11(26):75-77.