堆整序的最优算法*

顾训穰 诸宇章

(上海科学技术大学,上海 201800)

摘要 本文给出堆整序的一个新算法,其实用价值比以前的算法效率提高一倍,其理论意义是在复杂性的数量级和主项系数方面已具有最优性能.

关键词 堆,比较整序,算法,计算杂性.

[1]中 Heapsort 算法是六十年代由 Williams, Floyd 提出的,其时间复杂性为

$$2n\log_2 n+O(n)$$

因为比较整序的时间复杂性下界是 $n\log_2 n+O(n)$,[1,2]所以,该算法还不是最优的.

Floyd 对此曾提出过当极大化堆的堆底元素较小时,要设法避免作不必要的元素比较,以提高算法的效率,但未给出具体的实现方法.[3]

1 定 义

定义 1. 设二叉树 T 是一个堆,[4]以 T 中任一结点为根的子树是 T 的一个子堆(subheap).

定义 2. T 中任一结点处所附元素如果被取走,则该结点称为空缺结点.

2 算 法

2.1 设计思想

重新堆化过程用递归来实现.

设当前堆的高度为 h,当把堆顶元素取走后,为了将这个当前堆重新堆化,只需注意到能移入该空缺结点,以充当新堆顶元素的必是其左右儿子中的大者.于是只需一次比较,就可使大者上升一层.令这个过程重复进行到第 $\lfloor(1/2)h\rfloor$ 层上的某一结点(即为空缺结点)处.

如果当前堆外的那个未整序的叶结点元素比该空缺结点的父结点元素不小,则将其移

* 本文 1990 年 11 月 20 日收到,1991 年 7 月 29 日定稿

作者顾训穰,50 岁,副教授,主要研究领域为算法设计,计算复杂性.诸宇章,30 岁,讲师,主要研究领域为算法设计,计算复杂性.

本文通讯联系人:顾训穰,上海 201800,上海科学技术大学计算机科学系

入这个空缺结点，并通过逐次与其父结点比较，使其上升到当前堆的某一合理位置.

否则，递归地重新堆化以这个空缺结点为堆顶的当前子堆.

这一递归过程至多可一直进行到 h=1. 此时，将当前堆外的那个未整序的叶结点元素移入当前这个空缺结点，并至多作两次比较，就可把高度为 1 的这个当前子堆重新堆化.

2.2 算法描述

2.2.1 建立初始堆[2,5]

```
procedure HEAPFIFY(i,j); /* 将数组元素 A[i]—A[j]建成堆 */
  if  i不是一片叶且若i的儿子含有一个比i大的元素
    then begin 令 k 是 i 的带有较大元素的儿子;
               交换 A[i]和 A[k];
               HEAPFIFY(k,j)
         end;
procedure BUILDHEAP; /* 将数组元素 A[1]—A[n]建成堆 */
  for i← ⌊n/2⌋ step −1 until 1 do HEAPFIFY(i,n);
```

2.2.2 重新堆化

```
procedure REBUILDHEAP(i,j);
  if h=1
    then /* 将高度为1的当前子堆重新堆化 */
      begin A[i]←A[j+1];
            HEAPFIFY(i,j)
      end
    else
      begin
        count←1;
        while count≤ ⌊(1/2)h⌋ do /* 左右儿子作一次比较,大者上升一层的过程只进行到当前子堆的第
            ⌊(1/2)h⌋ 层的某结点处 */
          begin 令 k 是 i 的带有较大元素的儿子;
                A[i]←A[k];
                i←k;
                count←count+1;
          end;
        if A[j+1]≥A[ ⌊i/2⌋ ]
          then /* A[j+1]在当前子堆中上升到某一合理位置 */
            begin A[i]←A[j+1];
                  while A[i]>A[ ⌊i/2⌋ ] do
                    begin 交换 A[i]与 A[ ⌊i/2⌋ ];
                          i← ⌊i/2⌋ ;
                    end
            end
          else /* 递归地重新堆化当前子堆 */
            begin h←h− ⌊1/2)h⌋ ;
                  REBUILDHEAP(i,j)
            end
      end;
```

2.2.3 堆整序

输入：待整序的元素 A[i] (1≤i≤n) 的数组.

输出：已整序的数组 A

```
procedure OPTIMAL−HEAPSORT;
  begin
    BUILDHEAP;
    for i←n step −1 until 3 do
```

```
        begin temp←A[1];
              h← ⌊log2(j−1)⌋ ;
              i←1;
              REBUILDHEAP(i,j−1);
              A[j]←temp
        end;
    将A[1]与A[2]交换
end;
```

3 复杂性分析

设在结点 i 到结点 j 范围内,以 i 为根的当前堆的高度为 h. 则有:

引理. 递归过程 REBUILDHEAP 的执行,需作比较次数 $T(h)\leqslant h+\log_2 h+1$

证明:重新堆化过程是正确的,这可对递归深度施归纳证明.

在将当前堆重新堆化时,从堆顶空缺结点开始,其左右儿子通过一次比较,使大者上升一层的过程进行到第(1/2)h 层的某结点处,共作(1/2)h 次比较. 然后,把当前堆外的那个未整序的叶结点元素 A[j+1]和当前空缺结点的父结点元素作一次比较,在最坏情况下,或者 A[j+1]上升至当前堆的堆顶,共作(1/2)h 次比较;或者在以这个空缺结点为堆顶,高度为(1−(1/2))h=(1/2)h 的当前子堆上递归地为 A[j+1]寻找合理位置,因而有下列递归方程:

$$\begin{cases} T(h)=(1/2)h+1+\begin{cases}1/2h\\ T((1/2)h)\end{cases}\\ T(1)\leqslant 2\end{cases}$$

设递归最大深度为 Kmax,则有$(1/2^{Kmax})h=1$,即 $Kmax=\log_2 h$.

显然,当递归深度达到最大时,所作的元素比较次数最多. 所以

$$\begin{aligned} T(h)&\leqslant((1/2)h+1)+((1/2^2)h+1)+\cdots+((1/2^{Kmax})h+1)+2\\ &=h((1/2^1)+(1/2^2)+\cdots+(1/2^{Kmax}))+Kmax+2\\ &\leqslant h+\log_2 h+1 \end{aligned}$$

证毕

定理. OPTIMAL−HEAPSORT 算法的最坏情况比较次数

$$T(n)=n\log_2 n+n\log_2\log_2 n+O(n)$$

证明:算法的正确性可通过对 for 循环的执行次数施归纳证明.

算法的耗费包括两部分:

(1)调用 BUILDHEAP,建立初始堆,其耗费为 O(n).[4,5]

(2)for 循环的耗费 T′(n). 因为含有 j−1个元素的堆的高度 $h=\lfloor\log_2(j-1)\rfloor$,所以:

$$\begin{aligned} T'(n)&\leqslant\sum_{3\leqslant j\leqslant n}(\lfloor\log_2(j-1)\rfloor+\log_2\lfloor\log_2(j-1)\rfloor+1)\\ &\approx n\log_2 n+n\log_2\log_2 n+n \end{aligned}$$

故 OPTIMAL−HEAPSORT 算法的总耗费 $T(n)=n\log_2 n+n\log_2\log_2 n+O(n)$.

参考文献

1 Horowitz E, Sahni S. Fundamentals of computer algorithms. Computer Science Press, Inc., 1978:61−70.

2 Alfred V Aho, John E Hopcroft, Jeffrey D Ullman. The design and analysis of computer algorithms. Addison—Wesley, Reading, Mass., 1975:86—92.

3 Donald E Knuth. The art of computer programming. Vol. 3, Sorting and Searching, Addison—Wesley Publishing Company, Inc., 1973:145—149,158.

4 朱洪等. 计算机算法:设计与分析引论. 上海:复旦大学出版社,1985:78—85.

5 曹新谱. 算法设计与分析. 长沙:湖南科学技术出版社,1984:70—75.

OPTIMAL ALGORITHM OF HEAPSORT

Gu Xunrang and Zhu Yuzhang

(*Shanghai University of Science and Technology, Shanghai* 201800)

Abstract A new heapsort algorithm is given in this paper. Its practical value is that the efficiency of it is two times as high as that of the former algorithm. Also, its theoretical significance lies in the order and the main term coefficient of the complexity being of optimal performance.

Key words Heap, sorting by comparision, algorithm, computational complexity.