第 7 卷第 1 期
1996 年 1 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 7, No. 1
Jan. 1996

堆的路径二分搜索算法*

王晓东　傅清祥　范　庆　王梅集

（福州大学计算机系　福州　350002）

摘要　本文提出堆的路径二分搜索算法.当用堆来实现优先队列时,此算法可用较少的比较次数完成插入及删除最大元素等操作.

关键词　堆,算法,计算复杂性.

60 年代由 Williams 和 Floyd 提出的堆整序算法以其精致的数据结构和巧妙的算法设计思想受到普遍的关注.近年来,我国学者对堆整序算法又作了改进[1,2],使其在计算时间复杂性的数量级和主项系数方面已达到最优,从而使其更加实用和高效.受这些工作的启发,我们着眼于充分挖掘堆的内在性质与潜力,提出堆的路径二分搜索算法,使得能用较少的比较次数搜索元素在堆中的正确位置.

1　路径二分搜索

用建堆算法 buildheap[1]将待整序的 n 个元素 $A[1],\cdots,A[n]$建成一个堆.此时由堆的性质知,从堆顶 $A[1]$到堆中任一结点 $A[k]$确定了一条唯一的路径,且在此路径上的诸元素 $A[1],A[i_1],\cdots,A[k]$已从大到小整好序.因此,如果我们能迅速确定这个路径上的中位结点,那么高效的二分搜索方法就可用到这一路径上.

若对数组 A 的每一下标 $i(1\leqslant i\leqslant n)$作一标记 $m(i)$使得

$$m(i)=\begin{cases}0, & i=2j,j=1,2,\ldots,n/2;\\ 1, & i=2j-1,j=1,2,\ldots,n/2,\end{cases}$$

则堆中任一从根结点 $A[1]$到结点 $A[k]$的路径可用标记序列$\{m(1),\cdots,m(k)\}$唯一表示.我们称这种路径表示方法为根节点 $A[1]$到堆中任一结点 $A[k]$的路径标记.

对此我们有:

定理 1. 对任一正整数 $k(1\leqslant k\leqslant n)$,若 k 的二进制表示为:

* 本研究得到福建省自然科学基金资助和国家教委留学回国人员资助费支持.作者**王晓东**,1957 年生,副教授,主要研究领域为算法与数据结构,计算几何学,计算机图形学.**傅清祥**,1941 年生,教授,主要研究领域为计算几何.**范庆**,1971 年生,硕士研究生,主要研究领域为算法与数据结构,计算几何学,计算机图形学.**王梅集**,1970 年生,主要研究领域为算法与数据结构,计算几何学.

本文通讯联系人:王晓东,福州 350002,福州大学计算机系

本文 1994－10－17 收到,1994－12－12 定稿

$$k=\sum_{i=0}^{\lfloor \log k \rfloor} b_i(k)2^i, \quad b_i(k)\in\{0,1\}.$$

则序列$\{b_{\lfloor \log k \rfloor}(k),\cdots,b_0(k)\}$为堆中从根 $A[1]$到结点 $A[k]$的路径标记. 反之亦然.

证明:对路长$\lfloor \log k \rfloor$用数学归纳法证明定理中的正命题部分.

当路长为 0 和 1 时定理显然成立.

设定理当路长为 $h-1$ 时成立,则当路长为 $h=\lfloor \log k \rfloor$时,$A[k]$的父结点为 $A[\lfloor k/2 \rfloor]$.

此时由于
$$k=\sum_{i=0}^{\lfloor \log k \rfloor} b_i(k)2^i,$$

故
$$k/2=\sum_{i=1}^{\lfloor \log k \rfloor} b_i(k)2^{i-1}+b_0(k)/2=\sum_{i=0}^{\lfloor \log k \rfloor -1} b_{i+1}(k)2^i+b_0(k)/2$$

因此
$$\lfloor k/2 \rfloor=\sum_{i=0}^{\lfloor \log k \rfloor -1} b_{i+1}(k)2^i$$

由归纳假设知$\{b_{\lfloor \log k \rfloor}(k),\cdots,b_1(k)\}$为堆中从根 $A[1]$到结点 $A[\lfloor k/2 \rfloor]$的路径标记. 另一方面,当 k 为偶数时 $b_0(k)=0$,当 k 为奇数时 $b_0(k)=1$. 因此,$\{b_{\lfloor \log k \rfloor}(k),\cdots,b_1(k),b_0(k)\}$为堆中从根 $A[1]$到结点 $A[k]$的路径标记.

定理中的逆命题部分可类似地证明. □

由定理 1 易知,在从根 $A[1]$到结点 $A[k]$的路径上依路长从大到小排列的$\lfloor \log k \rfloor+1$ 个元素中,第 i 个元素$(0\leqslant i\leqslant \lfloor \log k \rfloor)$在数组 A 中的下标的二进制表示为:

$$\{b_{\lfloor \log k \rfloor}(k),\cdots,b_{\lfloor \log k \rfloor -i}(k)\}.$$

因此,用对整数 k 的移位运算可在 $O(1)$时间内确定从根 $A[1]$到结点 $A[k]$路径上第 i 个元素的下标. 由此,若已知 $A[k]\leqslant key<A[j]$,其中 $A[j]$是从 $A[1]$到 $A[k]$路径上的一个元素,且 $A[j]$到 $A[k]$的路长为 L,那么,key 在 $A[j]$到 $A[k]$路上的合理位置可由如下 *binary*算法确定:

```
binary (k,L,key)
int L,k;
char key;
{ register int low,high,mid;
/* * * 设 A[k] ≤ key < A[k>>L] * * * /
low=0;
high=L;
while (low <= high) {
mid=(low+high)/2;
if (key<A[k>>mid]) high=mid-1;
else low=mid+1; }
return(mid);
/* * * A[k>>(mid-1)] ≤ key < A[k>>mid] * * * /
}
```

算法 *binary* 的正确性由定理 1 保证. 由于用了二分搜索算法,所以我们有:

定理 2. 在最坏情况下,算法 *binary* 只用 $\log L$ 次比较.

2 删除堆顶元素

若用堆来实现优先队列,则删除最大元操作相当于删除堆顶元素后,重建 $A[1]$

~$A[n-1]$为一个堆.事实上,若当前堆的高度为 h,则将堆顶元素取走后,为了将这个当前堆重新堆化,使空缺结点左、右儿子中大者上升一层.每上升一层只需作一次比较.令这个过程重复进行到 $f(h)$层上某一结点处.

如果当前堆的那个未整序的叶结点元素不小于该空缺结点的父结点元素,则由定理 2 知, *binary* 最多用 $\log(f(h))$次比较为其在当前堆内找到合理位置.否则,递归地重新堆化以这个空缺结点为堆顶的当前子堆.由于重建堆需要的比较次数关于 h 至少是线性的,因此,从平衡的观点看,$h-f(h)$应与 $\log(f(h))$大致相等.因此,我们取 $f(h)=h-\log h$.由此得到删除堆顶元素后的重建堆算法如下:

```
void rebuildheap (i,j)
int i,j;
{int k,l,count,m,t;
if (h==1) {
            A[i]=A[j+1];
            heapify(i,j);
            }
else {
l=h-log2(h);
h=h-l;
count=1;
while (count<=l) {
if (A[2*i]>A[2*i+1]) k=2*i;
else k=2*i+1;
A[i]=A[k];
i=k;
count=count+1; }
k=i/2;
if (A[j+1]>=A[k]) {
m=binary(k,l,A[j+1]);
for (t=0; t<m,t++) {
A[i]=A[i/2];
i=i/2; }
A[i]=A[j+1] ; }
else rebuildheap (i,j);
}
```

其中 *heapify*(i,j)将 $A[i]$~$A[j]$建成堆[1];$\log2(k)$对正整数 k 计算以 2 为底的对数.

在删除堆顶元素后,再调用 *rebuildheap*$(1,n-1)$即可将 $A[1]$~$A[n-1]$重新调整为一个堆.

3　复杂性分析

设在结点 i 到结点 j 范围内,以 i 为根的当前子堆高度为 h,则有:

定理 3. 在最坏情况下,*rebuildheap* 只需作 $T(h)\leqslant h+\alpha_3(h)+1$ 次比较.其中 $\alpha_3(n)$为三阶阿克曼函数 $A[n,3]$的逆.[3]

证明：*rebuildheap* 的正确性可通过对递归深度用数学归纳法得到证明. 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 $h-\log h$ 层的某结点处，共作 $h-\log h$ 次比较. 然后，将未整序叶结点 $A[j+1]$ 与当前空缺结点的父结点作一次比较，再用 $\log(h-\log h)$ 次比较找到 $A[j+1]$ 的合理位置或在以这个空缺结点为堆顶，高度为 $\log h$ 的当前子堆上递归地为 $A[j+1]$ 寻找合理位置. 因此，*rebuildheap* 所需的比较次数 $T(h)$ 由下列递归方程来确定：

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

由于 $T(\log h)>\log(h-\log h)$，推得 $T(h)\leqslant h-\log h+1+T(\log h)$.

设最大的递归深度为 $k\max$，即

$\underbrace{\log\cdots\cdots\log}_{k\max\text{ 次}}h=1$，则 $k\max=\alpha_3(h)$.

从而，$T(h)\leqslant h-\log h+1+\log h-\log\log h+1+\cdots=h+kmax+1=h+\alpha_3(h)+1$.

因此，在有 n 个元素的优先队列中删除最大元所需的比较次数小于 $\log n+\alpha_3(n)+1$. □

参考文献

1 顾训穆，诸宇章. 堆整序的最优算法. 软件学报，1994，**5**(1)：33～36.

2 顾训穆，诸宇章. 堆整序的改进算法及其复杂性分析. 计算机学报，1990，**13**(4)：289～292.

3 A V Aho, J E Hopcroft, J D Ullman. Data structures and algorithms. Addison-Wesley, Reading, Mass, 1983. 238～239.

A BINARY SEARCH ALGORITHM ON THE PATH OF A HEAP

Wang Xiaodong Fu Qingxiang Fan Qing Wang Meiji

(*Department of Computer Science Fuzhou University Fuzhou* 350002)

Abstract This paper presents a binary search algorithm on the path of a heap. If a heap is used to realize priority queue, $\log n+\alpha_3(n)+1$ comparisons are sufficient to replace the maximum element in the heap by the algorithm.

Key words Heap, algorithm, time complexity.