文章编号:1005－0523(2007)01－0117－03

置换奇偶性的快速算法

周尚超

(华东交通大学 江西 南昌 330013)

摘要:令 $a[1]$, a[2],…, a[n]是 $1,2,\cdots,n$ 的一个置换(排列),对任意 i,j 比较 a[i],a[j]可计算出置换的逆序数,根据逆序数的奇偶性就得到置换的奇偶性.这要进行 $n(n-1)/2$ 次比较,时间复杂度是 $O(n^2)$. 本文给出时间复杂度为 $O(n\log_2 n)$的两种算法:将置换表示为不相交的轮换的积来计算和归并排序的方法来计算.

关 键 词:置换;逆序数; 轮换

中图分类号:O157,TP31 **文献标识码**:A

1 将置换表示为不相交的轮换的积来计算

对于置换 9 7 6 5 4 3 8 2 1 , $a[1]=9$, $a[9]=1$, 可记为(19), $a[2]=7$, $a[7]=8$, $a[8]=2$, 记为(278), $a[3]=6$, $a[6]=3$, 记为(36), $a[4]=5$, $a[5]=4$, 记为(46). 置换 9 7 6 5 4 3 8 2 1 可表示为(19)(278)(36)(45), 是 4 个置换的积. (19),(278)等叫做轮换. (278)可记为(782)和(827),但一般是最小的数记在第 1 位. (278)的长度是 3,(36)的长度是 2. (278)的长度是奇数,是偶置换,(19)的长度是偶数是奇置换. 奇数个奇置换的积是奇置换,偶数个奇置换的积是偶置换,奇置换与偶置换的积是奇置换. 要计算置换的奇偶性,只要计算有多少个长度为偶数的轮换. 长度为偶数的轮换有奇数个就是奇置换,否则为偶置换,置换 9 7 6 5 4 3 8 2 1 有(19),(36),(45)共 3 个长度为偶数的轮换,所以是奇置换.

下面给出算法计算置换 $a[1]$, a[2],…, a[n]的奇偶性的算法

(278)可看做 2 变为 2 用了 3 次,同时 7 变 7 或 8 变 8 都用了 3 次. (278),(782)和(827)都是同一个轮换,为避免重复计算,利用数组 f,对已经使用过的数,用 f 将它标记为 1,表示此数已使用过,下次不能再使用了.

```
for(i=1;i<=n;i++)f[i]=0;
Odd=0;
for(i=1;i<n;i++)
{if(f[i])continue;
f[i]=1;
//计算 i 变为 i 的次数 t
s=a[i];t=1;f[s]=1;
while(s!  =i)
{s=a[s];f[s]=1;t++;}
if(t%2==0)Odd++;
}
```

Odd 是长度为偶数的轮换的个数. Odd 是偶数则为偶置换,否则为奇置换.

上面的算法使用了 2 个数组 a 和 f,空间复杂度为 $2n$,下面给出一个空间复杂度为 n 的算法. 这个算法利用数组 a 本身作为标记. 它占用较少空间,但理解较为困难.

对于(278),当算出 2 变 2 后,7 和 8 便无用了,因此赋值 $a[7]=7$; $a[8]=8$; $a[2]=2$;只有当 $a[i]>i$ 才计算.

```
Odd=0;
for(i=1;i<n;i++)
```

收稿日期:2006－10－02

基金项目:江西省自然科学基金项目(051106);国家自然科学基金项目(10661007)

作者简介:周尚超(1948－),男,云南蒙自人,教授.

```
{if(a[i]>i)
{
//计算 i 变为 i 的次数 t
p=i;s=a[i];t=1;a[p]=p;
while(s! =i)
{p=s;s=a[s];a[p]=p;t++;}
}
if(t%2==0)Odd++;
}
```

2 一种类似于归并排序的算法

上述轮换法只能计算出置换的奇偶性,不能计算出逆序数.

引理 1 设 $a[1], a[2], \cdots, a[m], b[1], b[2], \cdots, b[p]$是 1 个置换中的某 1 段,且 $a[i]<a[i+1], i<m$. $b[j]<b[j+1], j<p$. 如果 $a[i]>b[1]$,则将 b[1]移到 a[i]之前置换的逆序数减少 $m-i+1$.

利用引理 1,对置换进行若干次操作,将减少的数累加就得到置换的逆序数,且置换最终变为自然排列 $1,2,3,\cdots, n$.计算程序如下

```
//将 a[p],a[p+1],…,a[q-1] 与 a[q],
a[q+1],…,a[r]合并为排好序的 1 段
//这两段是已排好序的
void merge(int p,int q,int r)
{int i,j=p;
int startA=p,endA=q-1,startB=q,endB=r;
while(startA<=endA&&startB<=endB)
{
if(a[startA]<=a[startB]){b[j]=a[startA];
j++;startA++;}
else
{
b[j]=a[startB];j++;startB++;
num+=q-startA;
}
}
while(startA<=endA){b[j]=a[startA];
startA++;j++;}
while(startB<=endB){b[j]=a[startB];
startB++;j++;}
for(i=p;i<=r;i++)a[i]=b[i];
}
//计算置换 a[1],a[2],…,a[n]的逆序数 num
//初始状态:n0=n 段 ,每段长 seg=1,最后 1
段长 lastdeg=1
n0=n;seg=1;lastseg=1;
num=0;
while(n0>=2)
{//1 2  3 7  8 9  10  4 5 6 11
13
//将相临的两段合并为排好了序的 1 段,合并
后有 n0=n0/2 段
//段长为 s0=seg*2
m=n0;
n0/=2;s0=seg*2;
if(m%2==0)
{
for(i=1,j=0;i<=n0;i++,j+=s0)
{ if(i<n0)merge(j,j+seg,j+s0-1);
else merge(j,j+seg,n-1);
}
lastseg=seg+lastseg;
seg=s0;
}
else
{
for(i=1,j=0;i<=n0;i++,j+=s0)
merge(j,j+seg,j+s0-1);
j=n-lastseg-s0;
merge(j,n-lastseg,n-1);
seg=s0;lastseg+=s0;
}
}
```

3 计算实例

对 $n=80000$, $n=2000000$,构造置换 $a[i]=n-i+1$, $i<=n$ 进行计算,结果为

$n=80000$,用归并排序法,计算时间 0 秒(<1 秒)运行记录为:

```
start time :1167963535
num=3199960000 //逆序数
end time:1167963535
```

一般算法,对任意 i,j 比较 a[i], a[j],计算时间 62-35=27 秒

```
start time :1167963535
```

num=3199960000 //逆序数
end time :1167963562

n=2000000, 归并排序法,计算时间 1 秒, 轮换法计算奇偶性 0 秒(<1 秒)

4 应用

有 M 行每行有 N 个小正方形的矩形,里面写上 $1,2,\cdots,MN-1$,一个小正方形空着,称为空格,将空格上下左右的数字移动到空格中.问能否经过移动,从一种状态变为另 1 种状态.如果将空格去掉,依次写下第一行,…,第 M 行,这样就得到了 $MN-1$ 个数的置换,两种状态得到 2 个置换.如果 N 是奇数,那么当且仅当 2 个置换的奇偶性相同时,可从一种状态变为另一种状态.

如果 N 是偶数.当第 1 种状态的空格所在行与第 2 种状态的空格所在行的差为偶数时,可互变的条件是两个置换的奇偶性相同,否则可变的条件是奇偶性不相同.[1]

参考文献:

[1]陈景润.组合数学简介[M]. 天津:天津科学技术出版社, 1988.

[2]周尚超.整数运算中的问题和解决办法[J].华东交通大学学报,2005,(4):155～157.

A quick Algorithm for Inversion Number of Permutation

ZHOU Shang-chao

(East China Jiaotong University, Nanchang 330013, China)

Abstract: Let $\{a_1, a_2, \cdots, a_n\}$ be a permutation of the set $\{1, 2, \ldots, n\}$. If i<j and ai>aj then the pair (ai, aj) is called an "inversion" of the permutation. In this paper, by mergesort method a quick algorithm for inversion number of permutation was gained.

Key words: permutation; inversion number; alternation