算法中的映射思想

崔贵林

October 28, 2010

Abstract

本文通过两个例子阐述了算法中的映射思想，这种映射在将数据和一维数组下标之间建立起一一对应，在某些情况下会得到比较好的效果

Keywords:算法，映射，一一对应，二项展开，组合数，序数。

1 打印图形

1
5 2
8 6 3
10 9 7 4

书本上主要使用层和数组之间的转化关系来解决问题。也就是最后那句话：综合以上分析，i层第 j 个数据对应的数组元素是a[i-1+j][j]。
这是一种方法，但缺点是占用了不少存储空间，并且将近一般没用上。太浪费了！
能不能换一种方法?
看下面的数：

$S_1:$	1	2	3	4	5	6	7	8	9	10
$S_2:$	$(1,1)$	$(2,2)$	$(3,3)$	$(4,4)$	$(2,1)$	$(3,2)$	$(4,3)$	$(3,1)$	$(4,2)$	$(4,1)$
$S_3:$	$(1,1)$	$(1,2)$	$(1,3)$	$(1,4)$	$(2,1)$	$(2,2)$	$(2,3)$	$(3,1)$	$(3,2)$	$(4,1)$

S_1是自然数列的子集，S_2中的元素是S_1中元素在二维数组下的坐标，S_3中元素是S_1中元素在对角线和垂线组成的斜交坐标系下的坐标。
书本上的方法，说白了，就是寻找S_2和S_3的关系，然后从S_3构建S_2，最后打印输出S_2。
而这里我要讲的方法，是从S_1直接到S_2。
重新排列一下，把S_2放上面，按照S_2的先行后列顺序排列，于是得到:

$S_2':$	$(1,1)$	$(2,1)$	$(2,2)$	$(3,1)$	$(3,2)$	$(3,3)$	$(4,1)$	$(4,2)$	$(4,3)$	$(4,4)$
$S_3':$	$(1,1)$	$(2,1)$	$(3,2)$	$(3,1)$	$(2,2)$	$(1,3)$	$(4,1)$	$(3,2)$	$(2,3)$	$(1,4)$
$S_1':$	1	5	2	8	6	3	10	9	7	4

我们要做的，是直接由S_2'找到S_1'，然后打印输出，占用的空间几乎为零。
但这个关系不是那么好找的。仔细看看。。。
首先我们定义要输出的行数n。那么对于S_2'中的元素(i,j)，S_3'中对应为$(i+1-j,j)$.
其次，S_3'中元素(i,j)对应到S_1'中元素x为$x=n+n-1+n-2+\ldots+n-(i-2)+j=(i-1)\times n-\frac{(i-1)\times(i-2)}{2}+j$
那么 S_2'中元素(i,j)对应到S_1'中元素为$x=(i+1-j-1)\times n-(i+1-j-1)\times(i+1-j-2)\div 2+j=(i-j)*n-(i-j)*(i-j-1)/2+j$
这样把S_2'中下三角矩阵循环完毕，S_1'中的元素也就计算，打印，输出完毕！
具体的源代码请参阅http://code.google.com/p/calgorithm/source/browse/trunk/exercises.cc

2 二项展开和组合数的序数问题

例 从1,2,3,4四个数中

$S_1:$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_2:$	0	1	2	3	4	12	13	14	23	24	34	123	124	134	234	1234

我们知道二项展开和组合数是一一对应的。从上面也可以看出，两位数的个数是C_4^2，三位数的个数是C_4^3，四位数的个数是C_4^4。一位数的个数是C_4^1，这里0作为C_4^0了。

那么我们要求S_2到S_1的映射函数$f: S_2 \to S_1$，该怎么办呢？

对于$\forall x \in S_2, y \in S_2$，定义$d(x,y) = |f(x) - f(y)|$。

比方，求$f(134)$，可以这样，先求$f(123)$，再求$d(123,134)$。而123是首个三位数，所以$f(123)$应该是一位数的个数加上两位数的个数再加一，即$f(123) = C_4^1 + C_4^2 + 1 = 4 + 6 + 1 = 11$

下面求$d(123,134)$。既然首位都是12，那么是不是可以去掉呢？就是求$d(23,34)$了。34是3开头的最小两位数，那么$d(23,34) = d(23,24) + 1 = d(3,4) + 1 = 1 + 1 = 2$。

所以$f(134) = 11 + 2 = 13$

这个例子的数太少，不具有一般性。下面我们用更多的数，使推广到更一般的情形。

假设有$S_n = \{1, 2, 3, \cdots, n\}$，求$f$.

为方便叙述，这里假设$n \geq 8$，现在求一下$f(2458)$。先讲两个概念：紧接前元和紧接后元。相邻的两个数，前者叫后者的紧接前元，后者叫前者的紧接后元。

逻辑上是这样的，从末位开始，依次往前看。最后一位是8，前一位是5，8不是5的紧接后元，那么f(2458)=f(2456)+d(6,8)。

下面求$f(2456)$。现在到了5的位置。5是4的紧接后元，那么继续往前看。4不是2的紧接后元，那么$f(2456) = f(2345) + d(2345, 2456)$.

而$d(345,456) = d(345, 3(n-1)n) + 1 = d(45, (n-1)n) + 1 = C_{n-3}^2 + 1$

下面该2了，2是首位了，但不是最小的首位。那么$f(2345) = f(1234) + d(1234, 2345)$ $d(1234,2345) = d(1234, 1(n-2)(n-1)n) + 1 = d(234, (n-2)(n-1)n) + 1 = C_{n-1}^3 + 1$

$f(1234) = C_n^0 + C_n^1 + C_n^2 + C_n^3$

所以$f(2458) = f(2456) + d(6,8) = f(2345) + d(2345, 2456) + d(6,8)$

$= f(2345) + d(345,456) + d(6,8) = f(1234) + d(1234,2345) + d(345, 3(n-1)n) + 1 + d(6,8)$

$= C_n^0 + C_n^1 + C_n^2 + C_n^3 + d(1234, 1(n-2)(n-1)n) + 1 + d(45, (n-1)n) + 1 + d(6,8)$

$= C_n^0 + C_n^1 + C_n^2 + C_n^3 + d(234, (n-2)(n-1)n) + 1 + C_{n-3}^2 + 1 + d(6,8)$

$= C_n^0 + C_n^1 + C_n^2 + C_n^3 + C_{n-1}^3 + 1 + C_{n-3}^2 + 1 + d(6,8)$

即：

$$f(2458) = \sum_{i=0}^{3} C_n^i + C_{n-1}^3 + 1 + C_{n-3}^2 + 1 + d(6,8)$$

$$f(2458)=\sum_{i=0}^{3}C_n^i+\sum_{i=0}^{3}C_{n-1}^3+1+C_{n-3}^2+1+d(6,8)$$

这是递归的算法。若要递推呢。是这样的。$f(2458)=f(1234)+d(1234,2345)+d(2345,2456)+d(2456,2458)$
$=f(1234)+d(234,(n-2)(n-1)n)+1+d(45,(n-1)n)+1+d(6,8)$

$$f(2458)=\sum_{i=0}^{3}C_n^i+C_{n-1}^3+1+C_{n-3}^2+1+d(6,8)$$

问题提出：给定n，和不超过n长的数字串，求对应的序号。分析：使用一维数组存储该数字串，然后从前往后依次比较。核心算法：

```
int order()
{
        int order=0;
        for(int i=0;i<r;i++)
                order = order + Cnm(n,i
                   );
        for(int j=1;j<r+1;j++)
        {
                for(int k=0;k<a[j]-a[j
                    -1]-1;k++)
                        order = order +
                            Cnm(n-a[j
                           -1]-1-k,r-j)
                           ;
        }
        return order;
}
```

详细算法参见我的google code中order()函数。这样任给一个组合数就可以得到它的序数。这在拓扑中由最简基拓扑生成拓扑的过程中大有用处。

References

[1] 算法设计与分析,吕国英主编,清华大学出版社,2006年