全排列生成算法的研究和实现

1清华大学，计算机科学与技术

摘要 本文使用四种算法实现全排列，通过实现字典序法、递增进制法、递减进制法、邻位对换法这四种算法，更好地理解这四种全排列算法的核心思想和相通之处。文中对四种算法生成不同全排列的运行时间进行对比，验证其算法效率，并对其复杂度进行一定程度的分析。

关键词全排列 字典序法 递增进制法 递减进制法 邻位对换法 算法复杂度

## 1. 引言

全排列算法就是对于给定的字符集，用有效的方法将所有可能的全排列无重复无遗漏地枚举出来。任何n个字符集的排列都可以与1～n的排列一一对应。n个字符的全体排列之间存在一个确定的线性顺序关系。除最后一个排列外，所有的排列中都有一个后继；除第一个排列外，都有一个前驱。每个排列的后继都可以从它的前驱经过最少的变化而得到，全排列的生成算法就是从第一个排列开始逐个生成所有的排列的方法。

本文中，除字典序法外，其他三种经典的全排列生成算法都遵循着“原排列”→“原中介数”→“新中介数”→“新排列”的过程。本文简单介绍并初步实现了四种经典的全排列生成算法，其中依据中介数的不同会得到递增进位制数和递减进位制数。排列和中介数的一一对应性不难证明，在本文中我们不做讨论。四种经典全排列生成算法分别为字典序法，递增进位制法，递减进位制法和邻位对换法。

n个字符集的全排列一共有n!种，所以全排列算法的复杂度至少为O(n!)。如果要对全排列进行输出，那么输出的时间至少要O(n\*n!)，因为每一个排列都有n个数据。所以实际上，全排列算法对大型的数据是无法处理的，而一般情况下也不会要求我们去遍历一个大型数据的全排列。

本文的整体结构为，第二部分将简单介绍四种经典全排列算法的实现原理，并分别分析算法的时间复杂度。第三部分描述算法实现和结果分析。第四部分综合上述内容给出最终结论。

## 2．经典全排列算法的实现过程与复杂度分析

### 2.1 字典序法

全排列的字典序法描述如下：

1. 从排列的右端开始，找出第一个比右边数字小的数字的序号j（j从左端开始计算），即 j=max{i|<pi+1}。
2. 在的右边的数字中，找出所有比大的数中最小的数字，即 k=max{i|>}（右边的数从右至左是递增的，因此k是所有大于的数字中序号最大者）。
3. 对换，。
4. 再将.........倒转得到下一个排列。

### 2.2 递增进位制法

全排列生成的递增进位制法遵循着“原排列”→“原中介数”→“新中介数”→“新排列”的过程。

映射过程 将原排列按照从n~2的顺序，依次查看其右侧比其小的数字的个数。这个个数就是中介数的一位。

还原过程 我们设新中介数的位置号从左向右依次是n~2。画n个空格。对于每一个在位置x的中介数y，从右向左数y个未被占用的空格。在第y+1个未占用的空格中填上数字x。重复这个过程直到中介数中所有的位都被数完。

### 2.3 递减进位制法

递增进位制法的缺点是由原中介数得到新中介数时进位频繁，对递增进位数法进行改进得到递减进位制数法

映射过程 递减进位制的映射方法刚好和递增进位制相反，即按照n~2的顺序，依次查看其右侧比其小数字的个数。但是，生成中介数的顺序不再是从左向右，而是从右向左。

还原过程 递减进位制中介数的还原方法也刚好和递增进位制中介数相反，从中介数的最低位向最高位进行依次计数填空。

### 2.4 邻位对换法

递减进位制数法启发我们，能不能设计一种算法，下一个排列总是上一个排列某相邻两位对换得到的。递减进位制数字的换位是单向的，从右向左，逐个换位，直到最左端。邻位对换法的换位是双向的，通过保存数字的“方向性”来快速得到下一个排列。

映射过程 设定b2~bn为我们要求的中介数。2的方向一定是向左，i（i>2时）的方向规则如下：如果i为奇数，其方向性决定于bi-1的奇偶性，奇向右、偶向左；如果i为偶数，其方向性决定于bi-1+ bi-2的奇偶性，同样是奇向右、偶向左。当得到方向性后，bi的值就是背向i的方向直到排列边界这个区间里比i小的数字的个数。如此就能得到邻位对换法的中介数。

还原过程 还原过程与映射过程相反。

### 2.5 四种算法的复杂度如下：

|  |  |  |
| --- | --- | --- |
| 排列算法 | 时间复杂度 | 空间复杂度 |
| 字典序法 | O(n\*n!) | O(n) |
| 递增序列 | O(n^2\*n!) | O(n) |
| 递减序列 | O(n\*n!) | O(n) |
| 邻位对换 | O(n\*n!) | O(n) |

表1

四种经典算法的实现过程本文不加描述，根据实验数据得到四种算法的复杂度如图1：

图1 四种经典算法时间复杂度

可以看到字典序法和递增递减进位制法拥有较好的时间性能，因为邻位对换法计算中介数和中介数到全排列的过程更为复杂。

## 3. 算法实现及结果分析

### 3.1 实验环境

本实验的环境是Windows 8 64位 i5，4 GB内存。

### 3.2算法实现

四种经典算法的实现过程我们不加描述，运行结果如图2

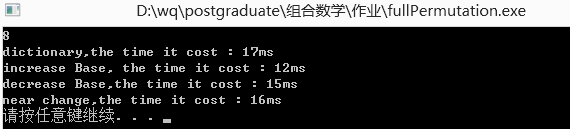


图2 四种经典算法运行结果

### 3.3 各算法的时间复杂度分析

由上图容易看出，随着长度的增加，递增进位制法运行时间明显比递减进位制法和字典序法增长的慢得多，这是因为本文在计算全排列时，是将递增的十进制数转换成中介数，然后根据中介数得到对应的排列，所以递增进制法效果最好。

## 4．总结

本文简单介绍了字典序法、递增进位制法、递减进位制法、邻位对换法四种经典全排列算法。理论上字典序法比较稳定，递增和递减以及邻位对换法，都因为有中介数的映射和还原，导致算法更加复杂且计算全排列的时间更久。本文在计算全排列时，是将递增的十进制数转换成中介数，然后根据中介数得到对应的排列，所以递增进制法效果最好。

## 参考文献

[1]王晓东编著．算法设计与分析[M]．北京：清华大学出版社，2002

[2]卢开澄，卢华明编著.组合数学[M]. 北京：清华大学出版社，2006

**附录：**

**实验数据**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 阶乘 | 字典序 | 递增进制 | 递减进制 | 邻位对换 |
| 6 | 0 | 1 | 0 | 1 |
| 7 | 2 | 1 | 2 | 2 |
| 8 | 18 | 12 | 14 | 16 |
| 9 | 162 | 120 | 146 | 167 |
| 10 | 1727 | 1318 | 1642 | 1827 |
| 11 | 20604 | 15780 | 20088 | 22170 |