

实 验 报 告

实 验 项 目 名 称 遗传算法探索工件排产

所属课程名称 人工智能基础及应用

实 验 类 型 操作型实验

实 验 日 期 2023.6.10

班级 工商管理21

学号 2021311410

姓名 吴振海

## 实验目的与概述

### 实验目的

遗传算法是一类借鉴生物界遗传机制和自然选择机制的随机搜索算法，适用于复杂和非线性优化问题。本实验中，我们将利用遗传算法探索和解决经典的FSP（流水间车间调度）问题，在此之前，我们将利用遗传算法进行一个简单的单变量实验，求解一个非线性函数在一个区间上的最大值以及对应的自变量的取值。因此该实验可以分为两个部分——单变量实验和工件流水线规划实验，下面是对这两个实验的简介：

（1）单变量实验：

求解目标函数

在区间[0,5]上的最大值以及对应的x的取值；

（2）工件流水线规划实验：

*n*个工件要在*m*台机器上加工，每个工件需要经过*m*道工序，每道工序要求不同的机器，*n*个工件在*m*台机器上的加工顺序相同。工件在机器上的加工时间是给定的，设为

试确定*n*个工件在每台机器上的最优加工顺序，使最大流程时间达到最小。

假设：TIME=[[31,41,25,30],[19,55,3,34],[23,42,27,6],[13,22,14,13],[33,5,57,19]]  #用时矩阵，子列表代表工件，子列表中的值代表对应工序用时

### 实验/算法原理简介

（1）单变量实验：

* 1. 编码：采用二进制数编码，将个体的值编码为二进数制构成的长度为10的列表；
  2. 群体设定：设定一个群体含有的个体数为100个；
  3. 适应度函数：本实验中原则上目标函数f(x)就是适应度函数，但是为了防止出现负值或者0而引发问题，采用函数值减去最小值加上一个极小的数(1e-3)；
  4. 选择：使用Python的numpy.random.choice(a,size,replace,p)函数进行选择，这里是随机选择，a是[0:1:99]，size是100，可选重复值，该位置处对应个体适应度值越大概率越大，该函数得到的是长度为100的将要选取的群体列表中个体的角标，将该角标代入群体列表就可以以一定的随机性选到适应度更大的个体进入新的群体中；
  5. 交叉：交叉概率选为0.8；
  6. 变异：变异概率选为0.003；
  7. 迭代次数：200代

选取的代码是作者提供的源代码：

（2）工件流水线规划实验：

数学理论上的原理：

将定义为工序在机器k上加工完工的时间；

则有：

目标为：确定{j1,j2,…,jn}使得

实验代码原理：

1. 编码：用列表表示，列表长度是工件总数，列表值代表工件编号，索引值是加工顺序；
2. 群体设定：显然调度方法一共有n!种，可以选取2倍n的平方作为群体个数，实现时间复杂度的降低，加快得到最优解；具体应该选多少，应该是存在一个概率关系，这里就不深究了；群体太大了，容易掩盖最优解（交叉过程中最优解被稀释了），增大迭代次数；群体太小了，很快就收敛了，容易陷入局部最优；本例中选为50；
3. 适应度函数：本例中这是一个很严重的问题，适应度函数的优劣基本上决定了我们能不能把最优解选出来，如果我们过多奖励一个群体中的最优解的话容易陷入局部最优；如果奖励不足，又会导致区分度不够，
4. 选择：依据概率选择，概率就是适应度函数值除以该群体中适应度函数值的总和，这个和单变量实验的选择方法相同；
5. 交叉：这个应该是争议最大的地方了，我选取的是，将父辈中奇数和偶数位的DNA进行交叉，即在一个循环中，循环次数为群体总数的一半，循环体是：如果随机概率落入交叉概率，就进行交叉；否则进入下一次循环；如果要交叉，将该次循环对应的两个父辈parent\_1/parent\_2的DNA各随机选取一半，分别放在两个子代child\_1和child\_2对应的位置上，将剩余的位置用另一个父辈的序列按顺序进行补充，得到交叉后的子代DNA；本例选取交叉概率为0.6；
6. 变异：对每一个子代进行的操作，遍历每一位编码，如果随机概率落入变异概率，则将该位置的索引录入选出数组，将该子代复制到一个新的数组，把没有选出的位置按原位填入新数组，把选出数组打乱，填入新数组，得到的就是变异后的数组；本例中选取变异概率为0.1；

### 实验环境

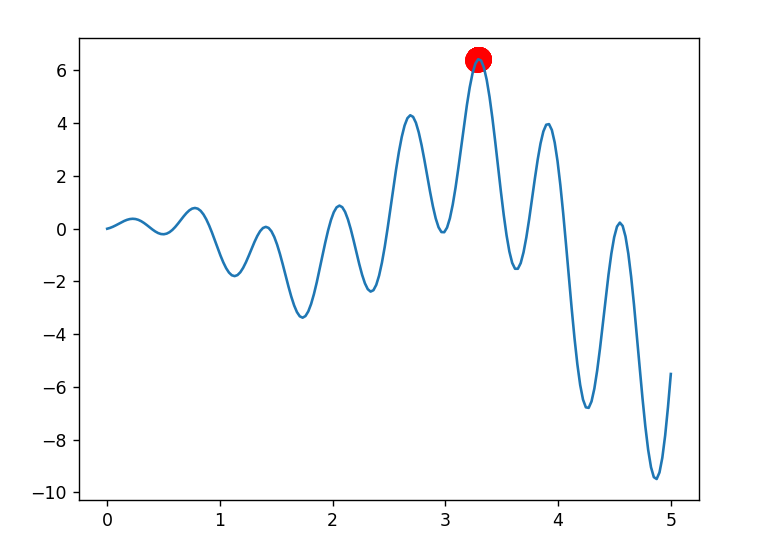
Python 3.10.0

## 实验步骤

应该包括关键实验过程描述及中间结果截图等，程序代码以附件的形式单独提交。

1. **单变量实验**
2. 直接运行：

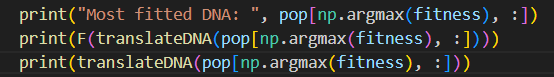
下载源代码之后我简单进行了浏览确保安全性后就直接运行了，反馈出了一个bug：在crossover(parent,pop)函数中，cross\_points的赋值语句出现了报错，发现是numpy库有了更新，将原来的np.bool改成np.bool\_即可，出现了预期的结果；

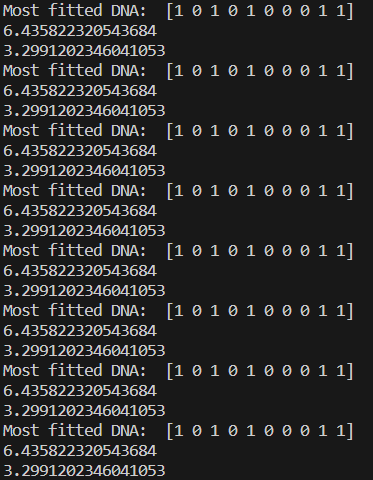




1. 结果的直观显示：

显然，输出结果是一个二进制数组成的列表，这并不能直观地得到目标函数在最高点的取值，因此我将该数组转化成十进制进行输出，得到了结果；



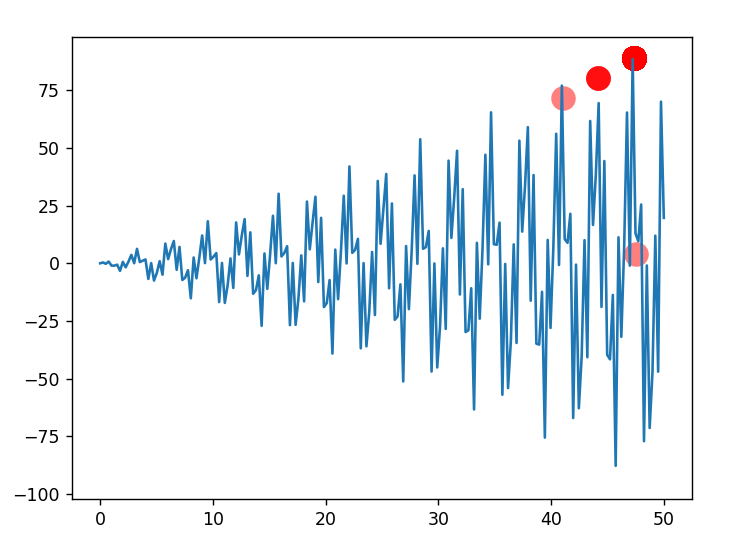
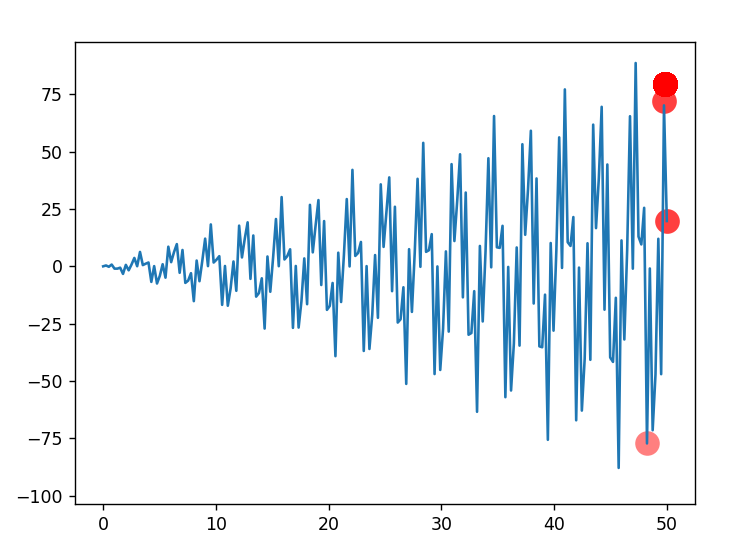


1. 一些思考和探索：
   * 1. 结果截图中可以看出，因变量的值很早就收敛到了6.430530388062344，自变量也是很早的就收敛到了3.2991202346041053，这说明迭代次数过多了，为了探究大概需要迭代多少次就可以得到该解，我加入了先知条件，得到该最优值就跳出循环，发现基本上不超过十次就可以得到最优值，也就是说，基本上10次以后整个程序就是在做着向最高值靠拢的工作；
     2. 迅速得到目标值的可能原因是区间小群体大、交叉概率较高；下面是验证：

当我把搜索区间改成[0,50]之后，出现了200次迭代无法找出最优解的情况，显然是陷入了局部最优解，并且增大迭代次数不能找到更好的解，这时减小交叉概率可以改变困境，改成0.6，解决了问题；

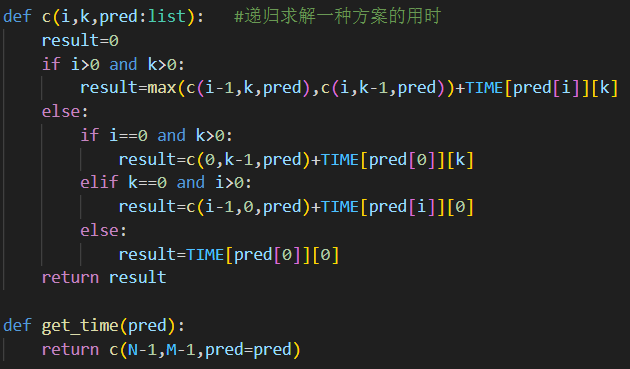
当我把交叉概率改成0.1之后，程序找到最优解的次数明显增加了，这就意味着，如果交叉概率继续减小，可能需要增加迭代次数才能找到最优解；

综上可以看出，交叉概率是一个需要取适当值的量，过高则容易陷入局部最优解，过低则会降低搜索速度；而同等条件下，区间越小越有利于迅速找到最优解，迭代次数越多结果的精确度越高；

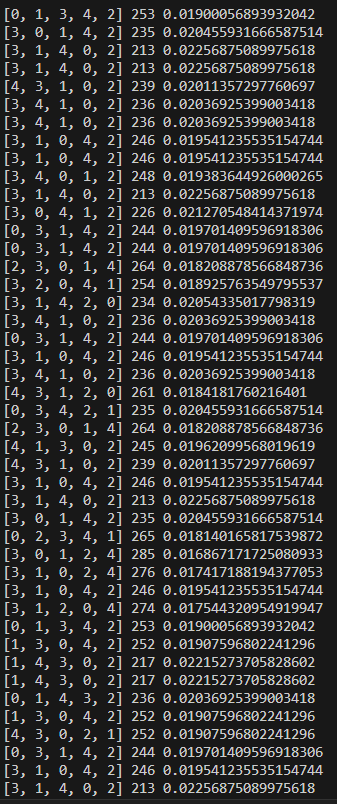
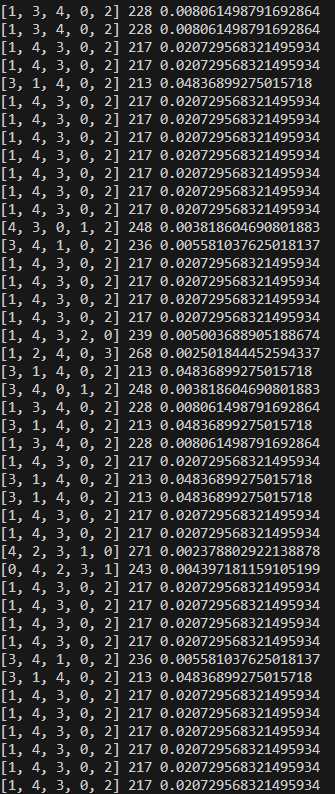


* + 1. 实际问题中，我们可以通过缩小区间范围来加快搜索速度，同时通过计算目标值分布的密度函数可能也有助于我们更快地找到目标值和对应的自变量；此外，由于变异和交叉都是有一定概率的，所以我们可能会陷入局部最优解，这个时候可以通过减小搜索区间、增大群体中的样本个数解决问题，更多的迭代无法奏效；有时我们发现最优值还没有达到明显的收敛状态，这时就应该增大迭代次数以实现收敛，缩小搜索区间也会起作用；另外，陷入局部最优解的问题是需要不断检验的，如修改各项参数，我们还可以通过双倍体、双种群、自适应的改进遗传算法来力图避免陷入局部最优解的困境；总之，不断调整优化和检验是通过遗传算法求解最优解问题的关键，避免局部最优解和加快搜索速度是存在一定的博弈关系，交叉概率是一个权衡两者的重要参数。

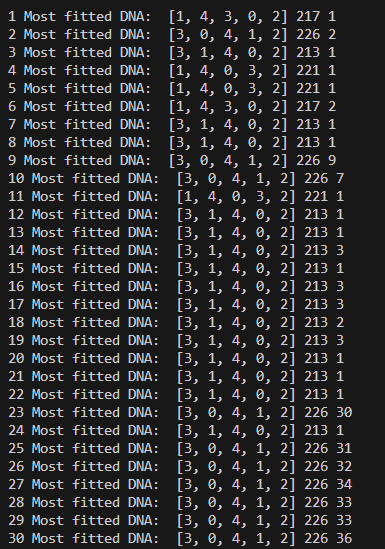
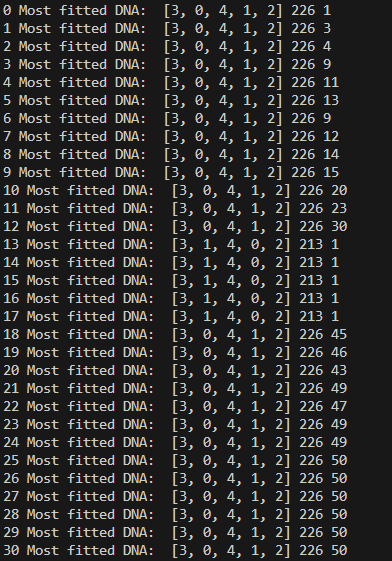
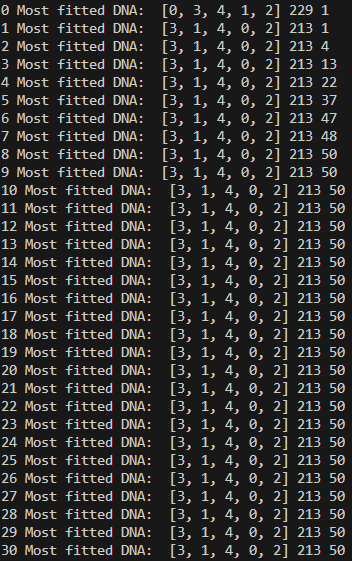
1. **工件流水线规划实验**
2. 关键代码的实现：由于没有找到现成的FSP代码（或者我觉得找到的不好用），所以我参考单变量实验的代码对该实验进行了代码的编写，其中一个重要功能的实现我觉得是值得一提的——计算一种方案用时数学原理的实现：显然这是一个递归，从最后一个元件和最后一道工序开始追溯，具体的数量关系和实验原理中阐述的一样，递归的终结条件是当i和k都为0（数组索引的初始位）时，返回的用时为第一个进入元件的第一道工序的用时；



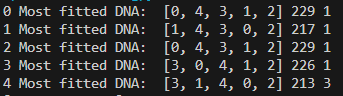
2.关于适应度函数的探索过程：如果采用1/时间作为适应度函数，会发现某些方案的适应度值相差很小，如下左图，最后一列代表该方案被选中的概率，213和235被选中的概率相差只有10%，从数学上来看想让这个概率差显现出来是很难的，被选中的概率基本相同，这就导致最优解有被覆盖掉的概率，经过多次修改和观察，采取1/(方案时间-单个工序所有元件最大用时之和+1e-3)作为试探，最终选取1/(时间-210)作为适应度函数，起到了一定的作用，213的概率要明显更高，如下右图；

3.关于交叉函数的探索过程：实验原理中也已经提到了，此处不赘述，事实上，关于交叉函数的思考是在不断地实践过程中探索出来的，开始使用1，我发现当DNA长度为5的时候这个值似乎太大了，最优解很容易就被淘汰了，而且基本上每一次迭代的收敛值都不一样，随机性很强，如下左图；改成0.1之后，发现初始群体中的最优解很容易占据优势地位然后将后来经过交叉变异产生的最优解挤出去，如下中图；选取了0.6作为交叉概率，还是会存在最优解被覆盖的情况，仔细思考觉得还是适应度函数的问题，选用了很多种，如1/(时间-210)，结果不尽令人满意，总是很难摆脱过早收敛的问题，即最优解213尚未出现就已经收敛了，比较好的一点是如果在开始的几次迭代中出现了最优解就总能收敛到这个解，我想这里是因为适应度函数区分度大的原因；如果开始几次没有出现最优解，次优解就会迅速占领群体，因为它们的适应度函数相对于后面的解也有显著差异，如下右图，所以，如果我们能够尽早出现最优解将会对收敛到该解有很大帮助，适当增大变异概率、适当增大种群规模会有帮助（这也是一个博弈的关系）；不过这种交叉方法是否在数学上是有效的还有待考证（我认为作为一种纯随机的结果是没有问题的）；

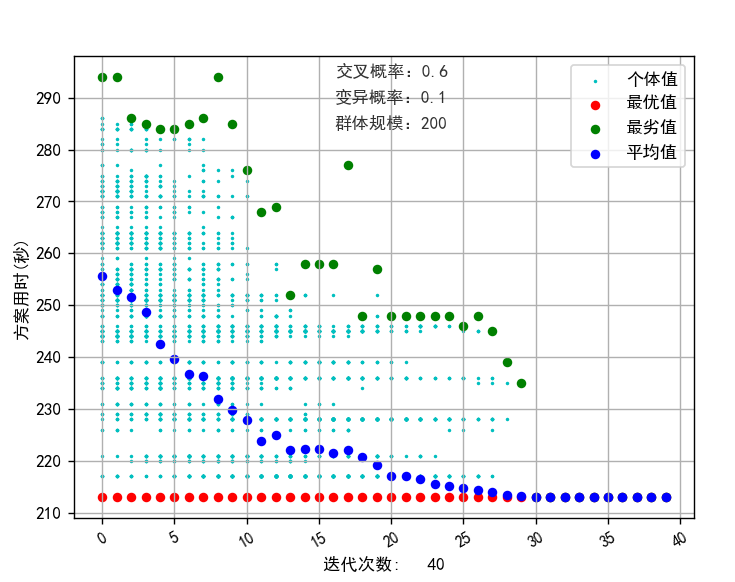
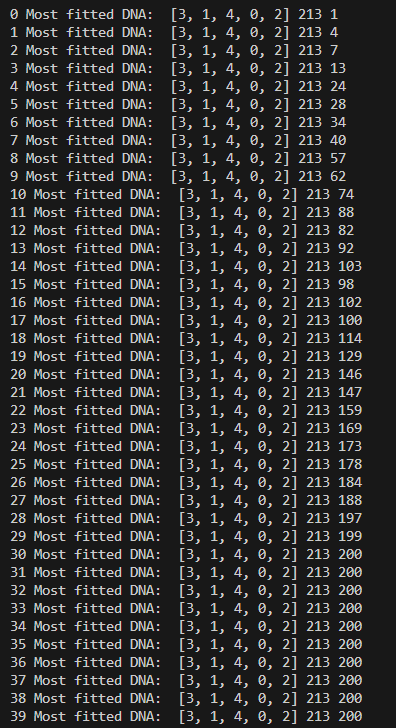
  

4.关于变异函数的交叉过程：变异一方面是产生最优解的重要渠道，另一方面也有可能导致为数不多的最优解因为变异而丢失，所以也存在权衡的关系，在适应度函数值变化较大的时候，尽快产生最优解很重要，因此变异概率要稍微大一点，下图是变异概率为0.2、适应度函数为1/(时间-MIN\_TIME)时的开始几次迭代，可以看到第四次依然能够产生最优解子代，有效避免了过早收敛的情况，并且这个最优解会很快占领群体；



5.关于列表的操作的注意事项：一定要注意.copy()，列表的引用是很容易出现错误的，因此在引用的时候要非常注意；

6.数据可视化处理：将样本点在散点图上表示，描绘出不同代数的个体值、最优值、最劣值和平均值变化趋势，这一块进行的非常顺利；（每一次迭代的结果会以“迭代次数+最优DNA+最优值+最优DNA出现次数”的形式在终端输出，不在散点图上表示）

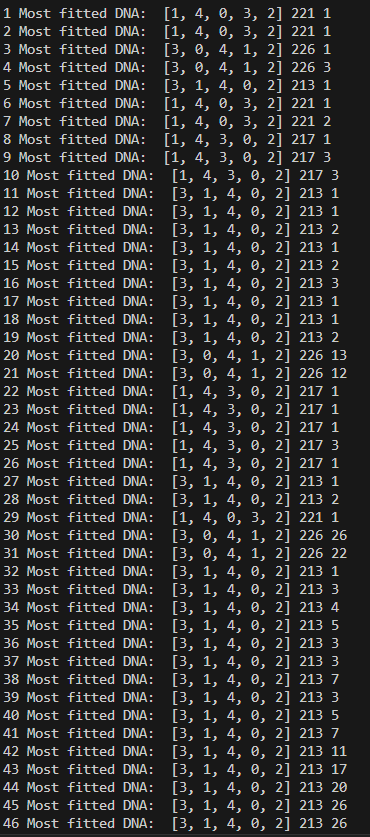
7.其他思考：

用遗传算法解决FSP问题是一种很好的方法，成功的关键在于适应度函数、交叉和变异概率的选取上，事实证明权衡这几个量的关系往往不是一次性可以成功的，经过探索和检验与矫正，我们可以逐渐得到最合适的参数和函数来解决不同的问题（显然不同问题的最优参数不会相同），多次实验取最优值可以很好地达到目标——本身遗传算法就是有一定概率的，如果算法本身不够完善可能会放大概率的效应，所以多次实验取最优值能够取得很好的效果；

在目标函数值变化比较大的情况下，群体规模的增大会起到很好地效果、变异概率适当的增大会大幅加快最优解出现的概率和其被选中的概率，下图是一个比较满意的结果（适应度函数为1/(时间-MIN\_TIME)，交叉概率是0.6，变异概率是0.1，群体规模是50）；

最后就是关于FSP问题本身，本例中我们只选取了5个工件，本身组合数就不多，并且最优解和次优解相差不大，这样收敛的速度是很快的，可能次优解已经占领了绝大部分最优解才刚出现，所以这个算法在组合总数不多的情况下并没有显示出很好地优越性，如果工件数量变多，效果将会更好；

此外，本案例还有一个特点，就是次优解[1, 4, 3, 0, 2] 217和最优解[3, 1, 4, 0, 2] 213在前三位不相同，所以当次优解占多数的时候不太容易通过变异或者交叉得到最优解，从而导致无法产生最优解的情况，我觉得这种问题就可以通过多次实验来解决（当然数学上可能会有其他方法但此处就不深究了）。



**三、实验代码（见附件）**

1.单变量实验 single\_variable.py

2.工件流水线规划实验 FSP.py

3.穷举列表 穷举法.txt