**算法分析与设计实验报告**

**第 一 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 李平凡 | 学号 | 201907040102 | | 班级 | 计科1905 |
| 时间 | 3.21 | 地点 | 软件大楼 | | | |
| 实验名称 | 金币阵列 | | | | | |
| 实验目的 | 通过搜索资料，要求掌握一些基本问题和典型例题的处理方式 | | | | | |
| 实验原理 | 题目给了两种操作，变换行，变化列  在这两种操作中，第一种操作会把某一行的元素全部取反，第二个操作会将两列值交换因此，首先利用两条规则进行变换，并且有这两个变换的相应函数，算法的具体思路如下：  (1）将矩阵中的每一列作为第1列，并利用第一个规则行变换将第1列中的相关元素与目标矩阵中第1列的元素进行配对，如果不相同，则利用每1个规则进行翻转；  (2）从第2列开始，将处理后的列与目标列进行比较，如果相同，则转下一列；如果不相同，看是否可以通过列的交换完成，如果不可以，则无法做到，如果可以，则继续扫描，直至所有的列描述完成为止。 | | | | | |
| 实验步骤 | ① 先依次找原矩阵每一列，对每一列进行处理  ② 将该列变成目标矩阵的第一列  ③ 判断当这一列为目标矩阵第一列的时候能否将原矩阵变换成目标矩阵若能记下操作数并将最小的操作数给ans，若不能输出-1 | | | | | |
| 关键代码 | void changel(int i,int j) //列变化 交换第i列和第j列  {  int s;  for(int k=0;k<n;k++)  {  s=a[k][i];  a[k][i]=a[k][j];  a[k][j]=s;  }  need++;  }  void changeh(int h) //行变换  {  for(int i=0;i<m;i++)  {  if(a[h][i]==1)  {  a[h][i]=0;  }  else  {  a[h][i]=1;  }  }  need++;  }      这是主函数中最关键的一个步骤，用来判断是否可以通过变换，变换成为目标矩阵，它最外层for循环就是找每一列通过变换作为目标矩阵第一列，看看有没有这种可能或者说变换次数更少，然后内部第一个循环使用a数组临时储存原目标矩阵防止其中数据改变，第二个循环是将原矩阵某一种情况变换后的第一列变换为目标矩阵的第一列，接下来的for循环是判断原矩阵接下来的每一列能否找到与目标矩阵相同的列，但有一点要注意就是在这个for循环里的一个for循环有一个same的判断，意思是说，如果我在原矩阵找与目标矩阵第j列相等的列，那么我必须得保证这个找到的列，与目标矩阵的相对应的列不能相同，否则得找下一个符合条件的列。最后判断出是否能够变换成相应的目标矩阵且ans记录最少步数。但上述算法必须要避免的bug就是当一个原矩阵除了第j列其他列都和目标矩阵相同时的情况，这里因为是一种对于上述算法单独出来的一种特例，所以只需要将这种特例的特征找到，然后加以判断即可，这种特例的特征就是find为1但第j列不等。 | | | | | |
| 测试结果 | 矩阵数据规模增加时间开销          解决100数据规模的8行4列矩阵消耗时间 | | | | | |
| 实验心得 | 对这个问题随机矩阵生成的是4行3列的矩阵所以解决问题可能快了一点但是当行列变多看最后一张截图这个问题的时间消耗就很高了，拿同样100数据规模的不同矩阵，与相同的处理1000个相同规模4行3列的矩阵时间开销相比时间开销差不多，可知算法对处理某一个问题而言很重要，对于n\*m矩阵，该算法中最复杂的部分就是找后面是否有相等的列用到了三层列数并且最后的列变换用到了一次行数故时间复杂度为O(m^3\*n) | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**