1. **问题描述**

设有n个运动员要进行网球循环赛。设计一个满足下列条件的比赛日程表：

* 每个选手必须与其他n-1个选手各赛一次；
* 每个选手一天只能赛一次；
* 当n是偶数时，循环赛进行n-1天。
* 当n是奇数时，循环赛进行n天。

1. **问题分析**

本次作业共实现了三种针对该问题的算法。

最容易想到的算法便是回溯法，通过条件判断穷举所有情况。

实际当中通常的处理单循环赛问题采用的是贝格尔编排法，这是一种基于固定轮转编排法的改进算法。改进内容为对于奇数队时其中一队总会和上一轮轮空的队进行比赛，但针对此次作业没有意义，因而实现了基本的固定轮转编排法。

最后一种基于分治的方法我并没有自己独立想到，最终参考了论文*( 扩展循环赛日程表算法研究[J]. 刘超,刘建辉,林森.辽宁工程技术大学学报. 2004 )*，并依照论文内容自己进行了实现。

1. **输入输出格式**

输入：输入n，即运动员个数

输出：首列为选手编号，其后第i列为第i天该运动员的对手，当为-时表示轮空。

1. **回溯法**
2. 算法思想

由于按照原表格式进行遍历时判断条件较为复杂，因而改将行、列均定义为选手，表项则为两名选手交战的天数，以遍历天数的方式进行。通过递归枚举表中所有位置的所有可能结果，再按照原表格式转换进行输出。

1. 测试情况

当n≥15时，程序已难以继续通过递归找到结果。

该方法的解决能力有限，但正确性不难证明，不再过多描述。

1. **固定轮转编排法**
2. 算法思想

采用“逆时针轮转方法”进行编排,先以阿拉伯数字作为代号,代替队名进行编排。把队数按U型走向分成均等两边,如遇单数队,最后一位数字补为0成为偶数。第一轮只要在U形相对队数之间划横线,即为第一轮比赛秩序。第二轮开始固定左上角1数字,其余数字均按逆时针方向移动一个位置,即为第二轮比赛秩序,以后各轮比赛秩序以此类推。

1. 测试情况

该算法相较于回溯处理效率较高，且规律性明显，可以使用双向链表直接进行输出。





其正确性也不难证明，通过轮转，每个运动员与余下n-1人进行比赛。

1. **分治法**
2. 算法思想

依题意，按照n为奇数时，进行n天；n为偶数时，进行n-1天将原问题分解为子问题。

参考n=2^k时的复制实现方法，当运用到任意n时，很容易发现当n为奇数时需要另外的复制方法。

根据论文，对于大小为n的问题，如果：

n=2，该问题为最小子问题，构造

在划分子问题时：

n为奇数，则将其转化为求解n+1；

n为偶数，则转为为求解n/2。并且对子问题进行合并：

如果n/2是偶数，则进行原先的复制方式，即左上角复制至左下角，左上角加n/2后复制至左下角及右上角；

如果n/2是奇数，则需要额外进行构造，这也是通过分治解决该问题的核心。具体方法是，对于左上角的数，若大于等于n/2，则将其替换为mid+i；替换完毕后，将左上角+mid后复制到左下角；再按序在右上角填写左下角数字，同时将右下角补全。

1. 正确性分析

对于数列，可以证明该数列是收敛的，因而该分治方法一定是可以终止的。

对于n为奇数、n为偶数、n/2为偶数的情况的处理方法理解较为容易，故不再讨论。

对于n/2为奇数的处理方法：

该算法很巧妙地替换掉了左上角大于等于mid的数，并且与右上角的数分开。对于任意左上角的数table[i][j]，在替换之后有两种情况（table[i][j]<mid或table[i][j]==mid+i），而同行右侧的数为mid + (i + j) % mid，由于右侧恰好比左侧少一列，所以j>0，因而左上角即使有大于等于mid的数，在右侧的同行也不会再出现。另一方面，由于右上角的数全都大于mid，因而借由右上角生成的右下角的数对于左下角而言也有同样的性质，即左下角任意table[i+mid][j]=(mid + mid + i) % n，右下角同行的数为table[mid + (i + j) % mid][j + mid] = i。通过这种生成方式，既保证了每行出现所有的数，又保证了每行不会有相同的数，同时右上角相当于两行之间移动了一位，因而也保证了每列不会有重复。

1. 测试情况





1. **总结**

通过本次作业，我对利用分治法解决问题有了更深入的认识。分治法的重点在于划分子问题与合并子问题，例如本次作业中当n为奇数时转换为n+1、当n为偶数时转换为n/2的巧妙划分方式，以及n/2为偶数时的普通复制合并、以及n/2为奇数时通过巧妙算法进行合并的思路都让我印象深刻。

但我认为，对于仅仅解决该问题而言，使用分治法其实并不是一种直观而高效的算法，而且由于数列的收敛速度难以确定(在n接近2的幂次时次数较少，期间较多，后附)，因而算法的复杂度也不够稳定，其正确性也并不容易证明。在实际现实中使用的固定轮转编排法以及贝格尔编排法都是更容易理解而且方便实现的。

非常感谢老师的作业设计！

