循环赛日程表分治法解决策略

2014211314 班　　2014211529　袁振宇

循环赛日程表算法是一个经典的计算机算法，它是分治算法的一个典型应用，但经典的循环赛日程表算法只能解决 2^n 个运动员的赛程排列问题，对于非 2^n 个运动员的赛程排列问题并不能很好地解决。因此，针对经典的循环赛日程表算法进行了相应的扩展，使其能够完成非 2^n 个运动员的赛程安排是值得考虑和实现的一个问题，并且应以相应的程序予以实现。

一、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。

二、初步分析

我们先考虑此问题的特殊情形，即 n=2^k，这时，循环赛一共进行 n-1 天。按此要求可将比赛日程表设计成有 n 行和 n-1 列的表。在表中第 i 行和第 j 列处填入第 i 个选手在第 j 天所遇到的对手。

按分治策略，可以将所有选手对分为两半，n 个选手的比赛日程表就可以通过为 n/2 个选手设计的比赛日程表来决定。递归地用这种一分为二的策略对选手进行分割，指导只剩下两个选手时，比赛日程表的制定就变得很简单了。这时只要让这两个选手进行比赛就可以了。下图展示出有 8 个选手参赛时的比赛日程：

1	2	3	4	5	6	7	8
2	1	4	3	6	5	8	7
3	4	1	2	7	8	5	6
4	3	2	1	8	7	6	5
5	6	7	8	1	2	3	4
6	5	8	7	2	1	4	3
7	8	5	6	3	4	1	2
8	7	6	5	4	3	2	1

图 1 n=8 时的比赛日程

一般情况下，算法可描述如下：

```
void Table(int k, int **a)
{
    int n = 1;
    for(int i = 1; i <= k; i++)
        n *= 2;
    for(int i = 1; i <= n; i++)
        a[1][i] = i;
    int m = 1;
    for(int s = 1; s <= k; s++)
    {
        n /= 2;
        for(int t = 1; t <= n; t++)
            for(int i = m+1; i <= 2*m; i++)
                for(int j = m+1; j <= 2*m; j++)
                {
                    a[i][j+(t-1)*m*2] = a[i-m][j+(t-1)*m*2-m];
                    a[i][j+(t-1)*m*2-m] = a[i-m][j+(t-1)*m*2];
                }
    m *= 2;
    }
}
```

三、进阶分析

现在考虑 n 为任意整数时的情形，这时增加了两个新问题，第一是参赛队员有奇偶之分，且当 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；第二是每次对问题进行分治时，产生的子问题可能与原问题不尽相同（当 n=6 时，产生子问题为 n/2=3，此时应按处理奇数的方法处理）。

对于第一个新问题，若 n 为奇数时，可产生一个虚拟选手，即参赛选手数增加为 n+1，此时 n+1 为偶数，这时就可以方便的使用一分为二的分割策略解决问题，但要注意，在实际安排比赛日程时虚拟选手并不必考虑在内，与虚拟选手比赛的选手该轮轮空（每天有且只有一名选手轮空）。

对于第二个新问题，当问题规模为 n（n 为偶数，n/2 为奇数时），可以利用辅助数组 b 来存储 n/2+1 到 n 的数来实现元素填充，算法参考下文源码中的 CopyOdd()函数，这样便解决了产生的两个新问题，从而可以用分治法解决有任意个参赛选手的循环赛日程表安排问题。

四、源码

```cpp
#include <iostream>
#include <cstdlib>
using namespace std;
```

```cpp
const int maxn = 1000;
int a[maxn][maxn],b[maxn];
//a为循环赛日程表数组，b为辅助数组，在n/2为奇数时辅助填充日程表

bool IsOdd(int n)                    //判断参赛选手数n是否为奇数
{
     return n&1; //奇数返回1
}

/*
    算法copy将左上角递归计算出的小块中的所有数字按其
相对位置抄到右下角，将左上角小块中所有数字加n/2，后
按其相对位置抄到左下角和右上角，就完成了比赛日程表。
*/
void Copy(int n)                      //若n为偶数则直接拷贝数组
{
    int m=n/2;
    for(int i=1; i<=m; i++)
        for(int j=1; j<=m; j++)
        {
            a[i][j+m] = a[i][j]+m;  //通过找规律，拷贝相应位置的元素
            a[i+m][j] = a[i][j+m];
            a[i+m][j+m] = a[i][j];  //将左上角相应位置的数赋给右下角
        }
}
```

```cpp
void CopyOdd(int n)                     //处理n/2为奇数的情况
{
    int m=n/2;
    for(int i=1; i<=m; i++)
    {
        b[i]=m+i;
        b[m+i]=b[i];
    }
    for(int i=1; i<=m; i++)
    {
        for(int j=1; j<=m+1 ;j++)
        {
            if(a[i][j]>m)
            {
                a[i][j] = b[i];
                a[m+i][j] = (b[i]+m)%n;
            }
            else
                a[m+i][j] = a[i][j]+m;
        }
        for(int k=2; k<=m;k++)
        {
            a[i][m+k]=b[i+k-1];
            a[b[i+k-1]][m+k]=i;
        }
    }
}
```

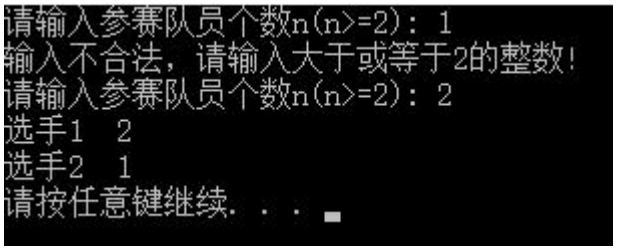
```
void MakeCopy(int n)                //根据情况进行元素拷贝
{
     if(n/2>=2 && IsOdd(n/2))
        CopyOdd(n);
     else
        Copy(n);
}

void Tournament(int n)
{
    if(n==1)
        a[1][1]=1;
    else if(IsOdd(n))
        Tournament(n+1);
    else
    {
        Tournament(n/2);
        MakeCopy(n);
    }
}
```

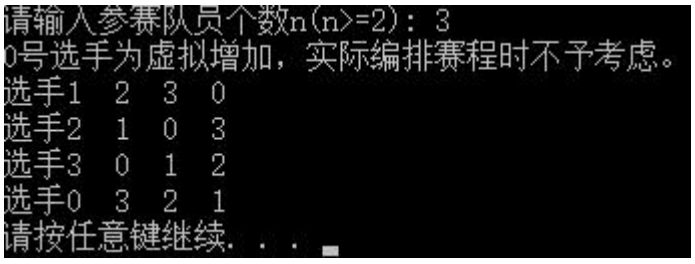
```
int main()
{
    int n;
    bool b = 0;                 //作为判断参赛选手是否为奇数的标记
    cout << "请输入参赛队员个数n(n>=2): ";
    cin  >> n;
    while(n < 2)
    {
        cout << "输入不合法，请输入大于或等于2的整数!" << endl;
        cout << "请输入参赛队员个数n(n>=2): ";
        cin  >> n;
    }

    Tournament(n);
    if(IsOdd(n))  //n为奇数
    {
        n++;
        cout << "0号选手为虚拟增加，实际编排赛程时不予考虑。" << endl;
        b = 1;
    }
```

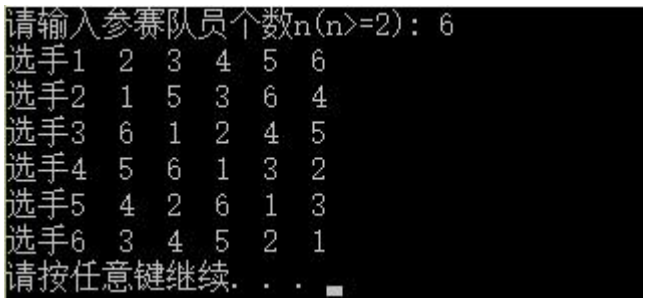
```
    for(int i=1; i<=n; i++)
    {
        for(int j=1; j<=n; j++)
        {

            if(j == 1)
                cout << "选手";
            if(a[i][j]==n && b)
                a[i][j] = 0;
            cout<<a[i][j]<< "   ";
        }
        cout<<endl;
    }

    system("pause");
    return 0;
}
```

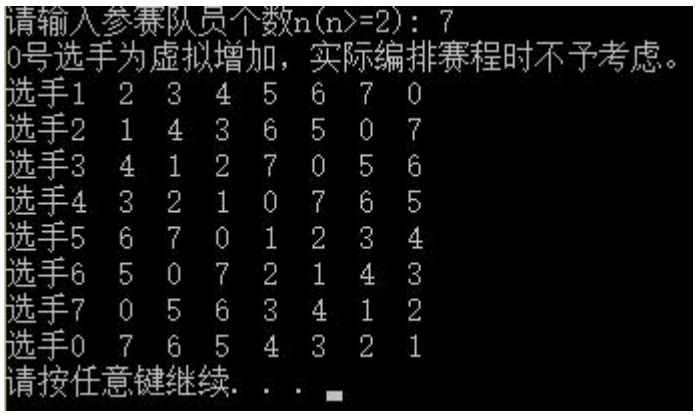
五、测试示例

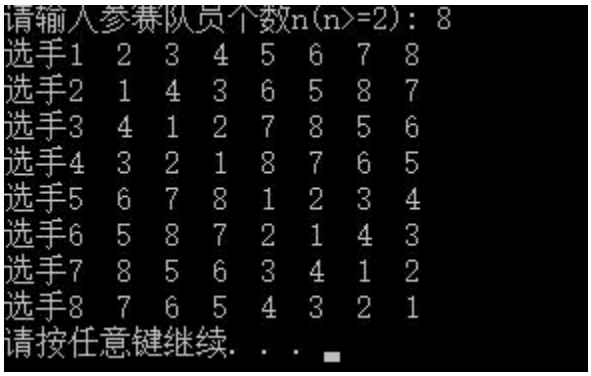
为了检验算法的正确性，选取比较有代表性的几个数据进行测试，选取的数据为：1，2，3，5，6，7，8。











六、时间复杂度分析

该算法将规模为 n 的问题分成规模为 n/2 的问题解决，并用额外的 O(n^2)时间对日程表进行填充，所以时间复杂度为 T(n)=2T(n/2)+O(n^2)。

七、心得体会

通过循环赛日程表问题，我更进一步的认识并理解了分治策略，将一个难以直接解决的大问题，分割成一些规模较小的相同问题，以便各个击破，分而治之。