### 分治法的应用——循环赛问题

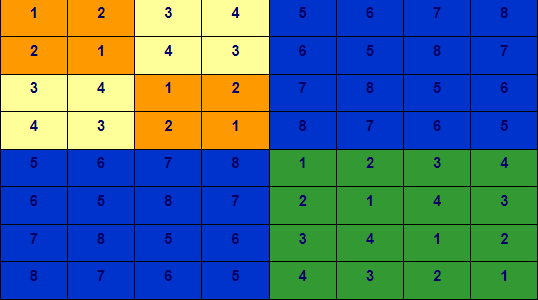
1. **问题描述**

设有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个选手各赛一次
* 每个选手一天只能赛一次
* 当是偶数时，循环赛进行天
* 当是奇数时，循环赛进行天

1. **问题分析**

* 对于



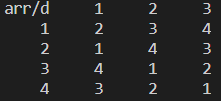
左上角和右下角相等，右上角和左下角相等。

* 对于

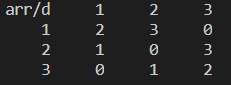
可以将其转化为偶数情形求解，即支队伍的情形，只需将第队伍最 后结果中置为0即可

* 其实，对于所有奇数情况，均可转化为的情形

列出时的日程表：



将其中的4转化为0：

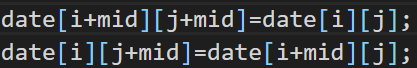


即为时的日程表

* 对于
* 如果为偶数，因为已经递归得到了前支队伍在前天的日程表， 为了保证不重复，我们采用递增的构造方式，让号选手与比大 的选手比赛，即



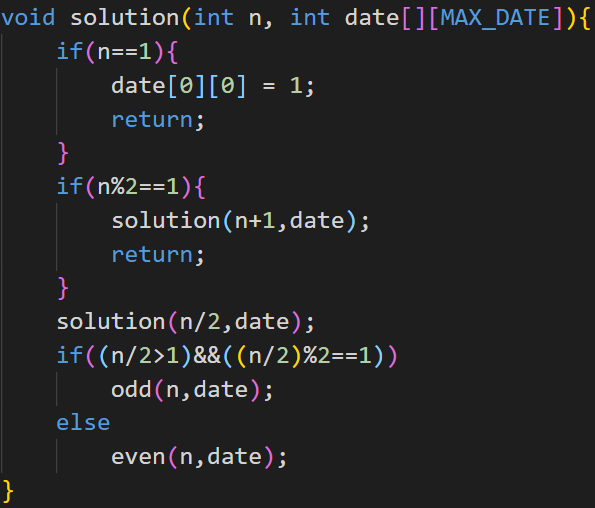
对于剩下的天数，我们复制之前的日程表即可：



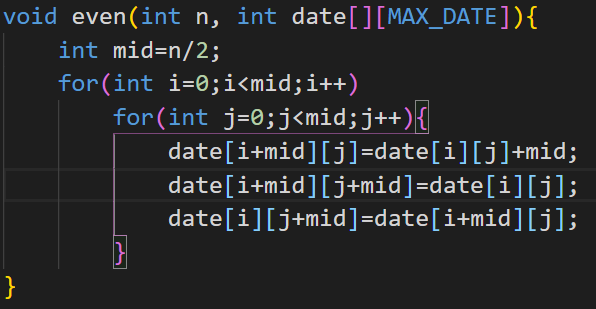
* 如果为奇数，增设一个虚拟的队伍，前天比赛的选手与下一个未参赛 的选手进行比赛

1. **算法实现**

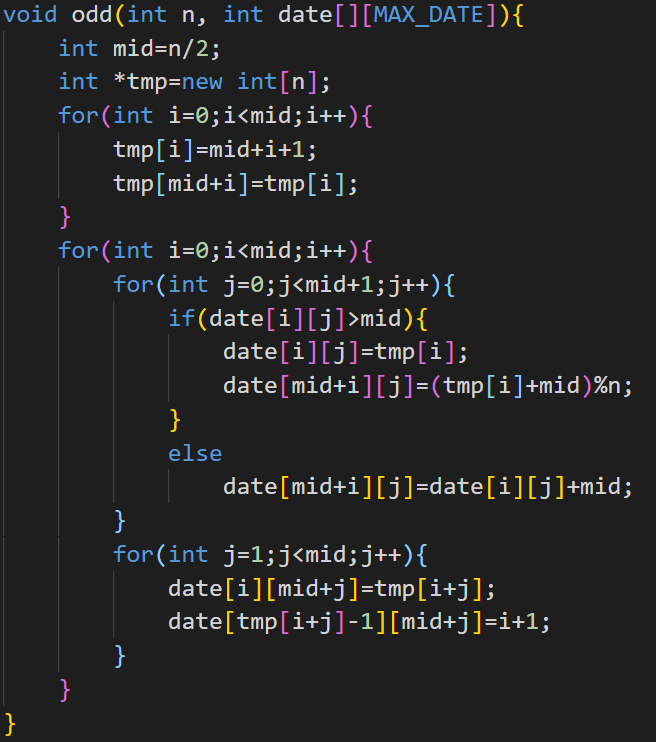
* 分治部分：奇数情形转偶数情形，并针对的奇偶情况进行分类



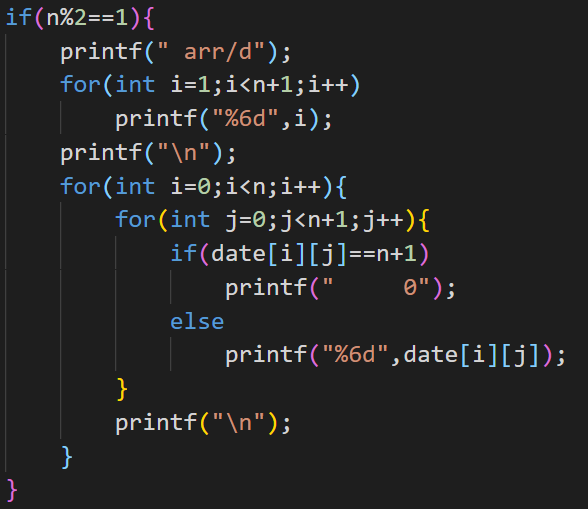
* 为偶数：

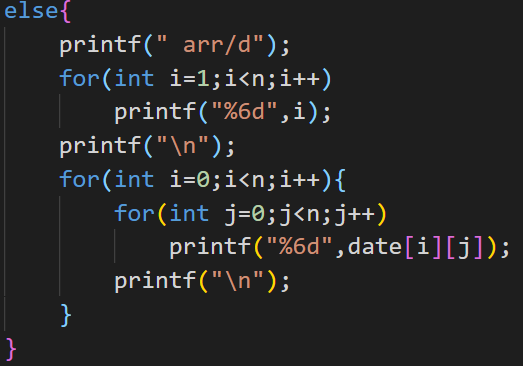


* 为奇数：



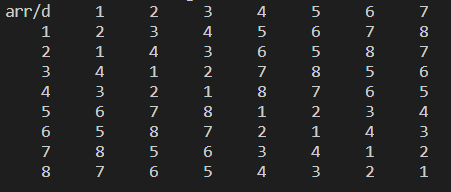
* 输出函数：





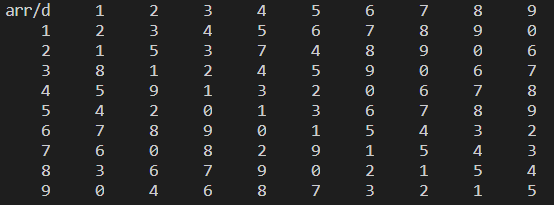
1. **结果测试及分析**

* **当时，结果如下：**



其中，其中一行代表一个选手，每一列代表一天，即第行列表示第个选手第天的对手编号，经检验，结果符合要求

* **当时，结果如下：**



其中，0表示该选手当天没有比赛，一行代表一个选手，每一列代表一天，即第行列表示第个选手第天的对手编号，经检验，结果符合要求

* **算法的时间复杂度为**

1. **实验总结**

网球循环赛的本质是划分子问题，利用递归求解，最后合并问题。

分治法适用的问题有以下特点：

* 该问题的规模缩小到一定的程度就可以容易地解决
* 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质
* 利用该问题分解出的子问题的解可以合并为该问题的解
* 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子子问题。

1. **附录：源代码：**

# include <cstdio>

#define MAX\_DATE 100

using namespace std;

void even(int n, int date[][MAX\_DATE]){

    int mid=n/2;

    for(int i=0;i<mid;i++)

        for(int j=0;j<mid;j++){

            date[i+mid][j]=date[i][j]+mid;

            date[i+mid][j+mid]=date[i][j];

            date[i][j+mid]=date[i+mid][j];

        }

}

void odd(int n, int date[][MAX\_DATE]){

    int mid=n/2;

    int \*tmp=new int[n];

    for(int i=0;i<mid;i++){

        tmp[i]=mid+i+1;

        tmp[mid+i]=tmp[i];

    }

    for(int i=0;i<mid;i++){

        for(int j=0;j<mid+1;j++){

            if(date[i][j]>mid){

                date[i][j]=tmp[i];

                date[mid+i][j]=(tmp[i]+mid)%n;

            }

            else

                date[mid+i][j]=date[i][j]+mid;

        }

        for(int j=1;j<mid;j++){

            date[i][mid+j]=tmp[i+j];

            date[tmp[i+j]-1][mid+j]=i+1;

        }

    }

}

void solution(int n, int date[][MAX\_DATE]){

    if(n==1){

        date[0][0] = 1;

        return;

    }

    if(n%2==1){

        solution(n+1,date);

        return;

    }

    solution(n/2,date);

    if((n/2>1)&&((n/2)%2==1))

        odd(n,date);

    else

        even(n,date);

}

void output(int n,int date[][MAX\_DATE]){

    if(n%2==1){

        printf(" arr/d");

        for(int i=1;i<n+1;i++)

            printf("%6d",i);

        printf("\n");

        for(int i=0;i<n;i++){

            for(int j=0;j<n+1;j++){

                if(date[i][j]==n+1)

                    printf("     0");

                else

                    printf("%6d",date[i][j]);

            }

            printf("\n");

        }

    }

    else{

        printf(" arr/d");

        for(int i=1;i<n;i++)

            printf("%6d",i);

        printf("\n");

        for(int i=0;i<n;i++){

            for(int j=0;j<n;j++)

                printf("%6d",date[i][j]);

            printf("\n");

        }

    }

}

void test(int n){

    int date[MAX\_DATE][MAX\_DATE]={0};

    solution(n, date);

    output(n,date);

}

void tests(){

    for(int i=8;i<20;i++){

        test(i);

    }

}

int main(){

    tests(9);

}