离散优化小组作业 I

杨锟　2313878

1　题目

Exercise 1.1

填格子问题：用 N 个整数 $(0,1,2,\ldots,N-1)$ 去填充一个 $n\times n$ 的格子.

要求：

(1) 格子的每行和每列元素各不相同；

(2) 对格子中任一元素 a_{ij}，如果存在两个不同 元素 a_{is} 和 a_{tj} 相等，那么这个元素 的第 s 列和第 t 行的所有元素都不等于 a_{ij}.

求满足条件的最小值 N，或描述出 N 的一个上界 (用 n 表示).

提示：$n=2$ 时，$N=3$；$n=3$ 时，$N=6$，那么，$n=4,5,6,\ldots$ 时，$N=$? 通过编程实现 N 比较小的结果.

2　对于题目更正的建议

值得指出的是，本题的表述存在一定的问题：

在要求 (2) 中，“如果存在两个不同元素 a_{is} 和 a_{tj} 相等”中的“不同”应该指的是元素在格子中的位置不同，但又要求这两个元素的值是“相同”的就会营造出一种矛盾的感觉，我建议可以把“如果存在两个不同元素 a_{is} 和 a_{tj} 相等”改为“如果存在两个在格子中位置不同的元素 a_{is} 和 a_{tj} 相等”.

在要求 (2) 中，“那么这个元素的第 s 列和第 t 行的所有元素都不等于 a_{ij}.”应改为“那么这个格子的第 s 列和第 t 行的所有元素都不等于 a_{ij}”.

3　解答

3.1　显式构造

首先 $N=n^2$ 显然是可以的，即所有格子上的数都各不相相同 (这显然是一组符合题意的构造)。

例如，$n=4$ 时，$N=n^2=16$. 我们有如下构造：

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{pmatrix}$$

3.2 一种更好的构造

接下来，我们可以考虑一种更好的构造：

为使 N 最小，我们可以考虑以对称矩阵形式构造. 在一个 $n \times n$ 的格子中，对角线元素有 n 个，非对角线元素有 $n^2 - n$ 个，考虑在每个对角线格子用一个不同的数填充，这样剩下的 $n^2 - n$ 个非对角线格子可以以对称的形式用 $\frac{n^n - n}{2}$ 个不同的数填充，用上述方法填充的格子经验证是符合题目要求的，则此时

$$N = n + \frac{n^2 - n}{2} = \frac{n^2 + n}{2} \tag{1}$$

下面我们对较小的 n 进行验证:

①当 $n = 4$ 时，由 (1) 可知，$N = \frac{4^2+4}{2} = 10$. 我们有如下构造：

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 5 & 7 & 8 \\ 3 & 6 & 8 & 9 \end{pmatrix}$$

此时 $N - 1 = 9 \iff N = 10$.

②当 $n = 5$ 时，由 (1) 可知，$N = \frac{5^2+5}{2} = 15$. 我们有如下构造：

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 8 \\ 2 & 6 & 9 & 10 & 11 \\ 3 & 7 & 10 & 12 & 13 \\ 4 & 8 & 11 & 13 & 14 \end{pmatrix}$$

此时 $N - 1 = 14 \iff N = 5$.

①当 $n = 6$ 时，由 (1) 可知，$N = \frac{6^2+6}{2} = 21$. 我们有如下构造：

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \\ 2 & 7 & 11 & 12 & 13 & 14 \\ 3 & 8 & 12 & 15 & 16 & 17 \\ 4 & 9 & 13 & 16 & 18 & 19 \\ 5 & 10 & 14 & 17 & 19 & 20 \end{pmatrix}$$

此时 $N - 1 = 20 \iff N = 21$.

以上是我们通过初步思考和结合对称性原理得到的容易被注意到的解，我们接下来考虑如何通过编程实现 N 的求解.(我使用 $C++$ 语言来求解)

3.3 编程求解

3.3.1 求解准备

准备：

```
#include<stdio.h>
const int maxn=100;//the max value of the input "n"
int a[maxn][maxn];//the matrix for storing the result
int n;//the input "n"
```

检验是否符合要求 (2):

```
bool check_row(int s,int t,int i,int j){//check whether a[i][s] & a[j][t] suit for the
    requirement(2)
    int k;
    for(k=1;k<=n;k++){
        if(a[t][k]==a[i][j])return false;
    }
    for(k=1;k<=n;k++){
        if(a[k][s]==a[i][j])return false;
    }
    return true;
}
```

检验求解出的结果是否符合题目的全体要求：

```
bool check(){
    int i,j,s,t,k;
    for(i=1;i<=n;i++){
        for(s=1;s<=n;s++){
            for(t=1;t<=n;t++){
                for(j=1;j<=n;j++){
                    if(i==t&&s==j)continue;
                    if((i==t&&a[i][s]==a[t][j])||(s==j&&a[i][s]==a[t][j]))return false;//check
                        whether the result suits for requirement(1)
                    if(a[i][s]==a[t][j]&&check_row(s,t,i,j)==false)return false;//check
                        whether the result suits for requirement(2)
                }}}}
    return true;
}
```

3.3.2 求解思路 1

我们考虑从 1 开始枚举 N，对 N 进行检验，直到检验出的 N 满足题目要求为止。

在这里我们考虑将每个元素的取值从 0 取到我们枚举的 N，再进行检验，这样做的好处是可以保证我们找到的 N 是最小的，即 N 是唯一的。

具体实施是通过递归来进行赋值，递归程序：

```
int N=0;
bool dfs(int m){//m represents the element number.
    int y=(m-1)%n+1;//Row number
    int x=(m-y)/n+1;//Column number
    for(int i=0;i<=N;i++){
        a[x][y]=i;
        if(check()==true)return true;
        if(m==n*n)continue;
        if(dfs(m+1)==true){
            return true;
        }
    }
    return false;
}
```

主程序：

```
int main(){
    scanf("%d",&n);//the input "n"
    int i,j,k;
    while(1){//Repeat the loop until you find a viable N
        if(dfs(1)==true)break;
        else N++;
    }
    printf("%d\n",N+1);//print the solution
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            printf("%4d␣",a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

这种算法在 $n = 2, 3$ 的时候很快就得到了书上的答案，但是当 $n = 4$ 的时候普通的计算机已经无法在短时间内检验出结果了。

3.3.3 求解思路 2

我们考虑先对我们最终要求的方格进行基础赋值再进行调整。

先将方格中的值赋成不同的值，即：同 3.1 节的构造。

```
scanf("%d",&n);//the input "n"
int i,j,k;
int N=0;
for(i=1;i<=n;i++){
    for(j=1;j<=n;j++){
        a[i][j]=((i-1)*n+j-1);
    }
}
```

接着我们考虑对方格进行第一次调整，具体思路是使方格中各个位置中的元素尽量小，即对于任意一个格子上的元素 a_{ij}，我们考虑将其从小到大赋值，即令 $a_{ij} = 0, 1, 2, \ldots, N$，对于每次赋值我们进行检查，检查该格子赋值成新的值是否仍然能够保持方格整体依旧符合题目条件，如此操作下去，直到赋的值符合题目要求为止。如果 $a_{ij} = 0, 1, 2, \ldots, N$ 均不符合题目要求，则我们现有的 N 是没法做到达到题目要求的，我们需要将 N 加 1，然后令 $a_{ij} = N + 1$，此时显然有且仅有 $a_{ij} = N + 1$，这样的方格是符合题目要求的。

```
bool flag1=false;//As a flag to mark whether the current N can meet the requirements of
    the question.
for(i=1;i<=n;i++){
    for(j=1;j<=n;j++){
        flag1=false;
        for(k=0;k<=N;k++){
            a[i][j]=k;
            if(check_find(i,j)==true){
                flag1=true;
                break;
            }}
        if(flag1==false){
            N++;
            a[i][j]=N;
        }}
}
```

由此我们结束了对方格的第一次调整，尽管按照这样的顺序操作下来，似乎每个 a_{ij} 都符合题目要求，但实则不然。我们在每次对 a_{ij} 进行检验的时候，检查的是当前状态下 a_{ij} 与其他元素匹配是否满足题目要求，但当我们对与 a_{ij} 不同的元素 (如 a_{ij+1}、a_{ij+2}) 进行赋值的时候，我们检查的是 a_{ij+1}、a_{ij+2}

与其他元素匹配是否满足题目要求，但不能满足 a_{ij} 仍然能与其他元素匹配符合题目要求。

例如，$n=3$ 时我们进行我们第一次调整的操作，得到的结果为

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix}$$

这样的结果显然是不符合题目条件的，因此我们还要进行进一步调整。

接下来到了求解这个问题的最难的部分，就是我们会发现我们每进行一次调整，虽然能够使调整的元素 a_{ij} 与其他元素相匹配，即：位于不同位置的 a_{ij} 与 a_{xy} 有 $a_{ij}=a_{xy}$ 且方格中的第 j 列和第 x 行的所有元素都不等于 a_{iy}。但不能保证的一点是：a_{ij} 出现在 a_{is}、a_{tj} 的检查中要求：第 s 列和第 t 行的所有元素都不等于 a_{ij}。这是我们无法保证的。

也许会出现巧合的情况，即：对于所有的 s 和 t，在 a_{ij} 与 a_{is}、a_{tj} 的检查中能够保证第 s 列和第 t 行的所有元素都不等于 a_{ij}。但绝大多数情况是没法保证的。

目前想到的处理办法就是按顺序依次对方格的每个元素进行调整，再对调整完的方格进行检查，若检查的结果不符合题目要求，则必然存在至少一个 a_{ij} 是造成不符合题目要求的结果的矛盾点，我们仍然按顺序对方格的每个元素进行调整，调整的结果必然和调整前的结果不同，此时我们再对调整后的方格进行检查，若检查的结果符合题目要求，则调整后的方格就是符合题目要求的结果。若不符合，我们仍然需要按照上面的方法进行调整，直到调整后的方格符合题目要求为止。

具体的代码实现如下：

```
1   while(1){//Since we do not know exactly how many operations are needed to obtain the
        final result, we use while(1) for an infinite loop until we find the square that
        meets the requirements of the problem.
2       bool check1=true;//Record whether we have actually adjusted the elements when we
            adjust the squares in sequence. If all the elements meet the requirements of the
             question at this time, then this square is in line with the requirements of the
             question, and we can exit our repetitive operation at this point.
3       for(i=1;i<=n;i++){
4           for(j=1;j<=n;j++){
5               if(check_find(i,j)==false){
6                   check1=false;
7                   bool flag2=false;
8                   for(k=0;k<=N;k++){
9                       a[i][j]=k;
10                      if(check_find(i,j)==true){
11                          flag2=true;
12                          break;
13                      }}
```

```
                    if(flag2==false){
                        N++;
                        a[i][j]=N;
                    }}}}
        if(check1==true)break;
    }
```

在进行上面的操作后，我们得到的方格就是符合题目要求的结果，此时我们可以输出方格中的值，即得到 N 的值。

```
        printf("%d\n",N+1);//export "N"
        for(i=1;i<=n;i++){
            for(j=1;j<=n;j++){
                printf("%4d␣",a[i][j]);\\export the table what we are looking for
            }
            printf("\n");
        }
```

在编程的过程中，我们发现，按照这样的方法求解是没法计算算法的时间复杂度的，因为是通过重复操作直到得到我们想要的结果，我们并不清楚要进行多少次、进行的次数是否与给定的 n 有关，因此我们无法计算出算法的时间复杂度。

用我本地的计算机运行了一下上述代码，当 $n = 12$ 时，该程序用时 0.5869 秒得到最终结果；当 $n = 13$ 时，该程序用时 1.331 秒得到最终结果；当 $n = 14$ 时，该程序用时 99.86 秒得到最终结果；当 $n = 15$ 时，该程序用时 2258 秒得到最终结果；当 $n = 16$ 时，该程序历经一下午还没能得到最终结果。

甚至尽管有输出结果，我们仍然无法保证该结果对应的 N 就是最优解，要判断 N 是否是最优解，对于是否为最优解的验证，目前能想到的没有特别好的办法，只能通过枚举来进行（即求解思路 1），粗略计算一下，在最极端的情况，给定 n，我们对 $n \times n$ 的方格按格子逐个进行枚举需要 n^{n^2} 次，对于每个格子还需要和其他的剩余格子进行比较，这还需要 n^2 次，若找到两个相同的元素还需要进行 $2n$ 次关于行列的比较。由乘法原理，粗略地计算需要 $2n^{n^2+n+1}$ 次，当 $n = 4$ 时，$2n^{n^2+n+1} = 2^{43}$，而计算机通常来说运行的次数是每秒 10^9 次，$2^{43} \div 10^9 \approx 10^3$，即我们在 $n = 4$ 时检验最优解在最极端的情况下就得花费 1000 秒，更不用说检验次数伴随 n 的增长差异非常巨大，在 $n = 5$ 的时候普通的计算机已经无法在短时间内检验出结果了，$5^{31} \div 10^9 \approx 10^{28}$，照这个算法需要的检验时间已经超过了地球的诞生时期。当然关于最优解的检验肯定有更为简单的检验方法，但不管如何，检验方法的时间复杂度也是相当大的，无法在短期内得到结果。

从结果上来看，我们的算法确实是找到了比 (1) 构造的更好的解，即：

$$
\begin{aligned}
n = 10, N = 39, 39 < 55 &= \frac{10^2 + 10}{2} \\
n = 11, N = 44, 44 < 66 &= \frac{11^2 + 11}{2} \\
n = 12, N = 50, 50 < 78 &= \frac{12^2 + 12}{2} \\
n = 13, N = 59, 59 < 91 &= \frac{13^2 + 13}{2} \\
n = 14, N = 69, 69 < 115 &= \frac{14^2 + 14}{2} \\
n = 15, N = 78, 78 < 136 &= \frac{15^2 + 15}{2}
\end{aligned}
$$

4 结果

最后我们列举一下通过编程得到的 $n = 10, 11, 12, 13, 14, 15$ 时的结果：

①$n = 10, N = 39$：

$$
\begin{pmatrix}
0 & 12 & 19 & 20 & 24 & 10 & 2 & 18 & 7 & 14 \\
27 & 29 & 3 & 12 & 35 & 19 & 1 & 8 & 5 & 11 \\
3 & 37 & 23 & 14 & 9 & 0 & 28 & 11 & 19 & 21 \\
29 & 2 & 8 & 31 & 0 & 1 & 13 & 28 & 34 & 5 \\
26 & 8 & 21 & 16 & 12 & 22 & 20 & 0 & 1 & 17 \\
13 & 19 & 33 & 18 & 7 & 38 & 15 & 17 & 30 & 6 \\
21 & 7 & 11 & 10 & 4 & 33 & 9 & 32 & 16 & 18 \\
5 & 11 & 13 & 3 & 6 & 4 & 32 & 36 & 2 & 22 \\
12 & 6 & 31 & 24 & 23 & 14 & 3 & 5 & 15 & 16 \\
17 & 9 & 6 & 15 & 27 & 12 & 23 & 1 & 4 & 25
\end{pmatrix}
$$

②$n = 11, N = 44$：

$$\begin{pmatrix}
4 & 11 & 0 & 18 & 24 & 33 & 30 & 2 & 28 & 8 & 5 \\
16 & 39 & 42 & 32 & 15 & 6 & 5 & 0 & 14 & 20 & 13 \\
29 & 2 & 16 & 8 & 25 & 4 & 23 & 36 & 15 & 7 & 34 \\
40 & 1 & 12 & 10 & 4 & 28 & 34 & 31 & 3 & 16 & 21 \\
6 & 26 & 1 & 34 & 39 & 30 & 14 & 4 & 24 & 11 & 9 \\
26 & 13 & 25 & 36 & 38 & 31 & 0 & 17 & 9 & 10 & 35 \\
8 & 17 & 9 & 0 & 11 & 27 & 3 & 37 & 6 & 21 & 15 \\
9 & 7 & 20 & 1 & 19 & 15 & 11 & 21 & 12 & 28 & 43 \\
41 & 9 & 14 & 6 & 27 & 2 & 18 & 16 & 23 & 5 & 12 \\
10 & 0 & 41 & 24 & 12 & 7 & 8 & 25 & 2 & 1 & 20 \\
27 & 29 & 13 & 31 & 22 & 20 & 26 & 9 & 32 & 19 & 3
\end{pmatrix}$$

③$n = 12, N = 50$：

$$\begin{pmatrix}
38 & 31 & 5 & 27 & 44 & 15 & 23 & 41 & 4 & 6 & 17 & 21 \\
4 & 32 & 26 & 7 & 39 & 9 & 33 & 14 & 8 & 45 & 0 & 28 \\
1 & 4 & 42 & 19 & 16 & 3 & 45 & 0 & 9 & 15 & 22 & 7 \\
36 & 33 & 30 & 2 & 28 & 1 & 38 & 8 & 12 & 26 & 31 & 27 \\
33 & 48 & 18 & 14 & 11 & 25 & 6 & 29 & 46 & 1 & 42 & 8 \\
21 & 11 & 14 & 44 & 24 & 34 & 3 & 2 & 1 & 13 & 4 & 16 \\
19 & 37 & 0 & 34 & 21 & 17 & 9 & 38 & 20 & 11 & 2 & 14 \\
17 & 10 & 1 & 48 & 41 & 22 & 0 & 35 & 29 & 2 & 43 & 23 \\
49 & 28 & 36 & 0 & 3 & 27 & 5 & 17 & 18 & 8 & 12 & 20 \\
29 & 7 & 8 & 3 & 43 & 5 & 10 & 40 & 24 & 47 & 25 & 12 \\
18 & 40 & 28 & 20 & 17 & 4 & 39 & 6 & 16 & 12 & 10 & 31 \\
13 & 35 & 7 & 11 & 15 & 37 & 19 & 5 & 22 & 32 & 23 & 6
\end{pmatrix}$$

④$n = 13, N = 59$：

$$\begin{pmatrix}
3 & 17 & 44 & 14 & 4 & 27 & 55 & 19 & 20 & 26 & 38 & 10 & 6 \\
1 & 27 & 30 & 53 & 10 & 32 & 36 & 22 & 8 & 42 & 37 & 21 & 2 \\
15 & 2 & 21 & 48 & 3 & 10 & 29 & 49 & 14 & 18 & 8 & 11 & 19 \\
17 & 45 & 33 & 21 & 14 & 9 & 44 & 41 & 12 & 31 & 18 & 7 & 20 \\
12 & 28 & 54 & 25 & 7 & 23 & 43 & 18 & 0 & 33 & 15 & 5 & 4 \\
33 & 13 & 27 & 3 & 45 & 16 & 21 & 1 & 24 & 6 & 4 & 40 & 35 \\
2 & 52 & 4 & 19 & 9 & 28 & 8 & 24 & 31 & 32 & 25 & 3 & 47 \\
5 & 4 & 14 & 32 & 51 & 26 & 16 & 3 & 39 & 0 & 30 & 6 & 8 \\
35 & 12 & 22 & 41 & 2 & 5 & 14 & 43 & 42 & 7 & 6 & 34 & 11 \\
40 & 56 & 12 & 8 & 16 & 36 & 13 & 9 & 5 & 1 & 28 & 37 & 23 \\
7 & 18 & 26 & 10 & 39 & 1 & 0 & 58 & 41 & 24 & 3 & 22 & 5 \\
31 & 1 & 6 & 30 & 26 & 38 & 7 & 10 & 16 & 57 & 46 & 15 & 9 \\
13 & 0 & 3 & 11 & 23 & 50 & 6 & 2 & 35 & 20 & 17 & 1 & 46
\end{pmatrix}$$

⑤$n = 14, N = 69$：

$$\begin{pmatrix}
12 & 28 & 26 & 2 & 4 & 54 & 32 & 21 & 46 & 29 & 27 & 30 & 13 & 22 \\
7 & 13 & 23 & 64 & 5 & 61 & 42 & 47 & 21 & 14 & 12 & 1 & 9 & 0 \\
10 & 30 & 21 & 39 & 40 & 32 & 18 & 11 & 58 & 4 & 14 & 50 & 5 & 3 \\
62 & 7 & 18 & 23 & 42 & 3 & 63 & 6 & 2 & 16 & 65 & 12 & 19 & 15 \\
15 & 37 & 1 & 22 & 33 & 27 & 3 & 19 & 4 & 68 & 31 & 32 & 48 & 43 \\
38 & 26 & 14 & 5 & 8 & 13 & 46 & 12 & 48 & 17 & 24 & 9 & 52 & 11 \\
8 & 25 & 22 & 42 & 20 & 7 & 16 & 15 & 11 & 0 & 32 & 53 & 24 & 59 \\
40 & 24 & 6 & 45 & 31 & 5 & 20 & 28 & 7 & 49 & 29 & 22 & 23 & 17 \\
34 & 57 & 36 & 25 & 18 & 28 & 66 & 4 & 20 & 2 & 6 & 10 & 22 & 8 \\
49 & 8 & 5 & 7 & 67 & 47 & 10 & 44 & 3 & 37 & 15 & 0 & 16 & 33 \\
23 & 18 & 39 & 41 & 3 & 22 & 6 & 1 & 17 & 34 & 11 & 51 & 50 & 37 \\
1 & 44 & 30 & 11 & 14 & 38 & 45 & 36 & 6 & 10 & 9 & 35 & 18 & 31 \\
24 & 60 & 10 & 4 & 1 & 34 & 9 & 35 & 26 & 15 & 19 & 29 & 41 & 20 \\
17 & 27 & 2 & 24 & 41 & 29 & 14 & 7 & 13 & 56 & 55 & 16 & 43 & 25
\end{pmatrix}$$

⑥$n = 15, N = 78$：

$$\begin{pmatrix}
65 & 13 & 37 & 36 & 48 & 6 & 26 & 3 & 1 & 39 & 52 & 32 & 19 & 30 & 24 \\
29 & 28 & 58 & 10 & 39 & 3 & 44 & 21 & 22 & 31 & 30 & 11 & 16 & 55 & 38 \\
38 & 60 & 25 & 12 & 14 & 58 & 68 & 4 & 0 & 8 & 56 & 1 & 7 & 22 & 34 \\
33 & 9 & 0 & 2 & 40 & 7 & 8 & 59 & 11 & 10 & 24 & 50 & 47 & 14 & 46 \\
6 & 11 & 13 & 26 & 3 & 4 & 5 & 43 & 77 & 32 & 72 & 46 & 14 & 15 & 17 \\
21 & 38 & 48 & 15 & 70 & 45 & 4 & 18 & 8 & 42 & 23 & 13 & 36 & 25 & 27 \\
19 & 35 & 23 & 49 & 36 & 15 & 12 & 7 & 67 & 3 & 29 & 18 & 41 & 2 & 10 \\
2 & 6 & 31 & 35 & 5 & 18 & 25 & 0 & 46 & 9 & 32 & 47 & 12 & 16 & 42 \\
69 & 64 & 6 & 16 & 71 & 9 & 1 & 36 & 33 & 41 & 20 & 2 & 40 & 21 & 5 \\
64 & 75 & 49 & 30 & 20 & 28 & 27 & 1 & 35 & 38 & 7 & 6 & 13 & 9 & 40 \\
28 & 25 & 45 & 17 & 9 & 66 & 19 & 2 & 31 & 13 & 3 & 57 & 11 & 44 & 8 \\
3 & 1 & 20 & 37 & 24 & 8 & 39 & 74 & 2 & 63 & 4 & 73 & 9 & 54 & 0 \\
5 & 20 & 21 & 41 & 26 & 44 & 35 & 50 & 15 & 14 & 37 & 19 & 0 & 29 & 30 \\
7 & 17 & 76 & 24 & 29 & 12 & 0 & 62 & 28 & 4 & 43 & 20 & 23 & 34 & 53 \\
32 & 8 & 16 & 52 & 50 & 19 & 61 & 11 & 5 & 24 & 22 & 23 & 51 & 27 & 12
\end{pmatrix}$$