Lab01 数独 实验报告

项子扬 PB16001768

1. 实验结果

一、前向检验

1.n=4，k=5

平均用时：8.87μs

解矩阵：

8.022779us	11.669497us	6.928764us
1 5 2 3	1 3 2 4	1 2 3 5
4 1 3 2	3 2 5 1	4 1 2 3
3 2 4 5	2 4 1 3	2 3 4 1
2 3 5 1	4 1 3 2	3 5 1 4

2.n=5，k=6

平均用时：38.90μs

解矩阵：

23.338993us	40.478566us	52.877406us
1 2 3 5 4	1 2 3 5 4	5 3 2 1 4
2 3 1 4 5	2 3 1 6 5	2 1 4 5 3
4 1 2 3 6	5 1 4 2 6	1 4 3 2 5
3 5 4 1 2	3 4 5 1 2	3 2 5 4 1
6 4 5 2 1	6 5 2 4 1	6 5 1 3 2

3.n=6，k=7

平均用时：77.43μs

解矩阵：

37.925864us	39.384551us	154.985501us
1 2 3 4 5 6	1 4 7 3 2 5	3 4 5 1 2 6
2 6 7 1 3 4	2 3 1 4 5 6	2 3 4 5 1 7
5 1 4 3 2 7	6 1 2 5 3 4	4 1 2 6 3 5
4 3 2 6 1 5	3 2 5 1 4 7	5 2 1 4 6 3
3 4 1 5 6 2	4 5 6 2 7 1	1 6 3 2 5 4
6 7 5 2 4 1	5 6 3 7 1 2	6 5 7 3 4 1

4.n=7，k=8

平均用时：168.11μs

解矩阵：

187.805960us	246.882786us	69.652307us
5 1 2 3 6 8 4	2 5 4 7 3 6 1	5 6 1 2 4 3 7
1 3 4 2 5 7 6	1 2 8 6 7 3 4	1 3 2 5 7 4 6
2 4 3 6 1 5 7	3 1 5 2 4 7 6	2 1 3 7 6 5 4
3 2 1 4 7 6 5	7 4 1 3 2 5 8	3 4 5 1 8 6 2
6 5 7 1 2 3 8	4 6 3 1 5 8 2	4 2 6 8 1 7 3
8 7 6 5 3 2 1	5 7 2 4 6 1 3	6 7 8 4 3 2 1
4 6 5 7 8 1 2	6 3 7 8 1 2 5	7 5 4 3 2 1 8

5.n=8，k=9

平均用时：331.61μs

解矩阵：

481.731403us	335.498025us	177.595150us
2 3 1 4 5 9 7 6	6 1 2 3 4 5 7 9	8 1 2 4 5 6 7 3
3 1 2 6 7 8 4 5	1 2 4 5 3 6 8 7	1 3 4 2 6 5 8 7
4 7 3 1 2 6 5 8	2 3 1 9 5 7 4 6	2 4 3 1 7 8 5 6
5 2 6 3 1 4 8 9	3 4 5 2 9 8 6 1	3 2 1 5 4 7 6 8
1 4 5 2 8 3 6 7	4 6 8 1 7 2 3 5	4 5 8 6 1 2 3 9
6 5 7 8 3 1 9 2	5 7 6 4 1 3 9 8	5 8 7 3 2 4 9 1
8 9 4 7 6 5 1 3	7 5 3 8 6 4 1 2	7 6 5 9 8 3 1 2
7 6 9 5 4 2 3 1	8 9 7 6 2 1 5 4	6 7 9 8 3 1 2 4

6.n=9，k=10

平均用时：407.34μs

解矩阵：

453.287005us	171.031058us	597.697025us
1 3 4 2 5 6 7 9 8	2 1 3 4 5 7 8 9 6	2 3 1 4 5 6 7 8 9
3 4 2 10 6 1 5 7 9	1 9 2 3 4 5 7 6 8	3 2 6 5 1 7 8 4 10
2 5 3 6 1 7 8 4 10	3 2 10 1 7 4 6 5 9	6 7 5 1 2 8 4 9 3
4 2 1 3 7 5 9 8 6	4 3 1 2 6 9 5 10 7	4 1 2 7 6 5 9 3 8
5 7 6 1 3 8 10 2 4	6 4 7 5 9 1 10 2 3	1 6 3 8 4 9 5 7 2
6 9 5 7 8 2 4 1 3	7 6 4 9 2 10 1 3 5	7 5 8 3 9 1 6 2 4
7 8 10 4 2 9 1 3 5	8 7 5 6 10 3 2 1 4	5 8 4 2 7 10 3 1 6
8 6 7 9 10 4 2 5 1	9 5 6 10 3 2 4 8 1	9 4 10 6 8 3 1 5 7
9 1 8 5 4 10 3 6 7	5 8 9 7 1 6 3 4 10	8 9 7 10 3 4 2 6 1

7.n=10，k=11

平均用时：49370.11μs

解矩阵：

634.528873us	240.683366us	147235.131603us
2 5 4 7 6 8 9 10 1 3	6 1 2 3 4 7 5 8 10 9	2 1 3 4 6 5 8 7 9 10
1 2 3 5 7 6 8 11 4 9	1 3 4 5 2 6 8 10 7 11	3 2 1 10 8 4 5 9 7 11
3 7 1 4 5 9 2 6 8 10	2 5 3 6 8 9 11 4 1 7	4 3 6 1 5 7 9 8 10 2
4 6 7 1 2 3 5 8 9 11	3 6 1 4 5 8 10 9 11 2	5 4 7 3 1 2 6 10 8 9
5 3 6 2 4 1 7 9 10 8	4 8 5 1 3 2 7 6 9 10	6 7 2 5 3 1 10 4 11 8
6 8 10 11 9 2 1 3 5 4	7 9 8 11 10 5 1 2 3 4	7 5 4 2 9 8 11 6 1 3
7 9 2 6 3 4 10 5 11 1	8 11 9 2 7 1 3 5 4 6	8 10 5 9 7 11 1 2 6 4
8 1 5 3 10 11 4 2 6 7	9 10 11 7 1 4 2 3 5 8	9 8 10 7 11 3 2 1 4 6
9 4 8 10 11 5 3 1 2 6	10 4 6 8 11 3 9 1 2 5	1 9 8 11 4 6 7 3 2 5
11 10 9 8 1 7 6 4 3 5	5 2 7 9 6 10 4 11 8 1	11 6 9 8 2 10 4 5 3 7

8.n=9，k=9

平均用时：819.78μs

解矩阵：

987.531143us	381.811339us	1090.003909us
1 3 4 5 2 6 7 8 9	1 2 3 4 5 7 6 8 9	2 1 3 4 5 6 7 8 9
2 1 3 4 5 7 6 9 8	2 1 4 3 6 5 7 9 8	3 2 1 5 4 7 6 9 8
3 2 1 6 4 8 9 5 7	3 4 1 2 7 8 9 5 6	4 5 2 1 6 8 9 3 7
4 5 2 1 3 9 8 7 6	4 3 2 1 8 9 5 6 7	5 3 4 2 1 9 8 7 6
5 4 7 8 9 1 2 6 3	5 6 7 8 9 1 2 3 4	6 4 7 8 9 1 2 5 3
6 7 5 9 8 2 1 3 4	6 5 8 9 1 2 4 7 3	7 6 5 9 8 2 3 1 4
7 8 9 2 6 3 4 1 5	8 7 9 6 2 3 1 4 5	8 7 9 3 2 4 1 6 5
8 9 6 7 1 5 3 4 2	9 8 5 7 4 6 3 1 2	9 8 6 7 3 5 4 2 1
9 6 8 3 7 4 5 2 1	7 9 6 5 3 4 8 2 1	1 9 8 6 7 3 5 4 2

9.n=9，k=9

平均用时：492.67μs

解矩阵：

492.306884us	548.466336us	437.241448us
1 7 2 3 4 5 6 8 9	6 1 2 3 4 5 9 7 8	1 2 3 4 5 6 7 9 8
3 1 4 2 5 6 8 9 7	1 2 3 4 5 6 7 8 9	2 3 1 5 4 7 6 8 9
4 2 3 1 6 9 5 7 8	4 3 1 5 2 7 8 9 6	3 4 2 1 6 8 9 5 7
5 3 1 8 9 2 7 4 6	3 4 5 1 8 9 2 6 7	5 1 4 2 3 9 8 7 6
6 4 5 9 7 8 1 3 2	5 6 4 7 9 8 3 2 1	6 5 7 8 9 1 2 4 3
7 5 9 6 8 1 3 2 4	2 5 8 9 7 1 6 3 4	4 6 8 9 7 2 3 1 5
8 6 7 4 2 3 9 1 5	7 8 9 2 6 3 1 4 5	7 8 9 3 2 5 1 6 4
9 8 6 7 1 4 2 5 3	8 9 7 6 1 2 4 5 3	9 7 5 6 8 3 4 2 1
2 9 8 5 3 7 4 6 1	9 7 6 8 3 4 5 1 2	8 9 6 7 1 4 5 3 2

10.n=9，k=9

平均用时：1053.29μs

```
893.810499us
1 2 3 8 4 5 6 7 9
2 5 1 3 6 4 7 9 8
3 1 5 7 9 6 2 8 4
4 3 9 6 1 2 8 5 7
5 4 6 1 7 8 9 2 3
6 9 7 2 8 1 3 4 5
7 6 8 4 2 9 5 3 1
9 8 2 5 3 7 4 1 6
8 7 4 9 5 3 1 6 2
```

```
1860.920023us
5 3 2 1 4 6 7 9 8
2 5 3 4 8 1 9 6 7
3 2 1 5 6 9 8 7 4
4 1 5 3 9 7 2 8 6
6 7 9 2 1 8 5 4 3
7 8 6 9 2 4 1 3 5
8 6 4 7 5 2 3 1 9
1 9 8 6 7 3 4 5 2
9 4 7 8 3 5 6 2 1
```

```
405.150332us
1 7 3 4 2 5 6 9 8
3 1 4 2 5 8 9 6 7
2 3 1 5 9 4 8 7 6
4 2 5 9 8 6 7 1 3
5 4 6 8 7 9 1 3 2
6 8 9 3 4 7 5 2 1
9 5 7 6 1 3 2 8 4
8 9 2 7 6 1 3 4 5
7 6 8 1 3 2 4 5 9
```

解矩阵：

11.n=9，k=9

平均用时：479.79μs

```
218.803060us
1 9 3 2 4 6 7 5 8
2 3 4 5 6 7 8 1 9
4 1 2 3 5 8 6 9 7
6 2 7 8 1 5 9 3 4
7 6 8 1 2 9 3 4 5
5 4 9 7 8 2 1 6 3
8 5 1 4 9 3 2 7 6
3 8 6 9 7 4 5 2 1
9 7 5 6 3 1 4 8 2
```

```
431.406699us
2 1 8 5 3 4 7 9 6
1 4 5 3 6 7 2 8 9
4 7 1 2 5 6 9 3 8
8 3 2 9 7 1 6 5 4
6 5 3 1 2 9 8 4 7
3 6 9 7 4 8 5 2 1
5 8 6 4 9 2 1 7 3
9 2 7 6 8 3 4 1 5
7 9 4 8 1 5 3 6 2
```

```
789.149702us
3 2 4 5 6 7 8 9 1
4 1 3 6 2 5 7 8 9
5 6 2 1 3 4 9 7 8
1 3 6 9 4 8 2 5 7
6 9 1 7 8 2 3 4 5
7 8 9 2 5 3 6 1 4
2 7 8 4 9 1 5 6 3
8 4 5 3 7 9 1 2 6
9 5 7 8 1 6 4 3 2
```

解矩阵：

12.n=9，k=9

平均用时：26885.79μs

```
7185.492482us
2 1 4 3 5 6 7 9 8
5 4 1 6 7 3 8 2 9
1 6 7 9 2 8 3 4 5
7 5 3 1 4 9 2 8 6
3 9 2 8 1 5 6 7 4
4 2 8 5 6 1 9 3 7
6 8 9 7 3 2 4 5 1
9 7 6 2 8 4 5 1 3
8 3 5 4 9 7 1 6 2
```

```
73190.352827us
1 2 4 3 5 9 6 7 8
2 4 5 7 6 3 8 1 9
4 1 3 8 2 7 9 6 5
3 9 1 2 7 8 5 4 6
8 5 2 6 9 1 4 3 7
7 6 8 9 3 2 1 5 4
9 7 6 5 1 4 3 8 2
6 3 9 4 8 5 7 2 1
5 8 7 1 4 6 2 9 3
```

```
281.526604us
8 4 1 2 3 6 7 5 9
1 6 2 9 8 7 5 4 3
3 1 5 7 6 4 8 9 2
4 5 3 1 2 9 6 7 8
5 3 4 6 1 8 9 2 7
6 8 9 5 7 2 1 3 4
2 9 7 8 4 5 3 1 6
7 2 8 3 9 1 4 6 5
9 7 6 4 5 3 2 8 1
```

解矩阵：

二、模拟退火

1.n=4，k=5

平均用时：22724μs

解矩阵：

25781.199858us
2 5 1 3
1 4 3 2
3 2 4 5
4 3 2 1

16432.839130us
1 3 4 5
3 2 5 4
2 5 1 3
4 1 3 2

25958.795008us
4 1 3 2
5 4 2 1
2 3 4 5
3 5 1 4

2.n=5，k=6
平均用时：133650μs

解矩阵：

155567.516789us
1 2 6 5 4
6 5 1 4 2
5 1 4 3 6
3 4 2 6 1
4 6 5 2 3

97909.993173us
6 2 3 5 4
5 3 1 6 2
2 1 4 3 6
3 6 5 2 1
1 5 6 4 3

147470.709564us
5 4 2 6 1
2 1 6 5 3
4 5 3 2 6
3 6 5 4 2
6 2 1 3 4

3.n=6，k=7
平均用时：243178μs

解矩阵：

238369.158688us
7 4 3 6 5 1
5 6 7 1 4 3
2 1 4 3 6 7
4 3 6 5 1 2
3 5 1 7 2 6
1 7 5 2 3 4

252187.665926us
5 6 7 3 2 1
1 3 2 4 5 7
6 2 1 5 7 4
4 7 5 6 3 2
7 5 6 2 1 3
2 4 3 7 6 5

238975.607835us
3 4 5 1 2 6
2 3 6 5 4 7
5 1 7 2 3 4
4 5 1 7 6 2
1 2 4 6 5 3
6 7 2 3 1 5

4.n=7，k=8
平均用时：375939μs

解矩阵：

372243.446119us
5 6 8 3 7 2 4
8 3 4 2 1 7 6
7 5 2 6 8 3 1
6 4 1 7 5 8 3
3 1 7 4 2 5 8
2 7 3 8 6 4 5
4 8 6 5 3 1 2

378209.840886us
6 5 8 2 3 7 1
1 8 4 6 7 5 3
8 7 5 4 2 1 6
7 4 2 3 5 6 8
2 6 3 7 4 8 5
4 1 6 5 8 3 7
5 3 7 8 1 2 4

377363.073045us
5 6 8 4 1 7 3
3 8 5 1 2 4 6
8 4 3 7 6 1 2
1 3 2 5 8 6 4
6 2 4 8 7 5 1
4 1 7 2 3 8 5
7 5 6 3 4 2 8

5.n=8，k=9
平均用时：545285μs

解矩阵：

540639.021629us	541413.949133us	553800.755016us
4 5 1 2 6 9 7 8	6 8 2 3 1 5 4 7	8 5 2 7 6 9 1 3
7 3 5 6 2 8 1 4	7 2 8 4 3 9 5 6	4 3 1 9 5 6 8 7
2 7 3 8 9 6 4 5	1 7 5 9 2 4 6 3	7 9 3 2 4 8 6 5
8 6 2 1 3 4 5 9	8 3 4 6 9 7 2 5	6 8 9 5 2 7 3 1
1 8 4 9 5 3 6 7	3 1 6 8 5 2 7 9	3 1 8 6 7 2 4 9
3 4 6 5 8 7 9 2	4 6 1 2 7 3 9 8	5 2 7 1 3 4 9 6
6 9 8 4 7 5 2 3	9 4 3 7 8 6 1 2	1 6 5 8 9 3 7 2
5 1 9 7 4 2 3 6	2 9 7 1 6 8 3 4	9 7 6 3 8 1 2 4

6.n=9，k=10

平均用时：804667μs

解矩阵：

831326.909275us	792691.759002us	789982.977122us
1 3 6 2 5 7 8 10 9	2 4 7 5 8 1 9 10 6	8 10 1 3 5 2 6 4 9
7 4 2 10 8 9 6 5 1	1 9 4 10 5 3 8 6 7	4 3 6 5 9 7 1 8 10
6 2 10 5 1 8 9 4 3	9 2 10 7 1 4 6 3 5	6 2 5 8 10 1 4 7 3
2 7 3 9 10 5 1 8 6	10 3 1 2 4 9 7 5 8	10 1 2 7 8 5 9 3 4
3 5 1 8 7 6 10 9 2	8 6 3 1 9 5 4 7 2	1 8 3 9 4 10 7 6 2
4 9 7 6 2 3 5 1 8	6 7 2 9 3 10 5 4 1	3 5 8 4 7 9 10 2 6
9 8 4 1 6 2 7 3 5	7 10 5 3 6 8 2 1 9	9 7 4 10 2 8 3 1 5
8 6 5 7 9 4 3 2 10	4 5 9 6 2 7 10 8 3	2 4 9 6 1 3 5 10 8
5 1 9 3 4 10 2 6 7	5 1 8 4 7 2 3 9 10	5 6 10 2 3 4 8 9 7

7.n=10，k=11

平均用时：1132154μs

解矩阵：

1057453.672099us	1162891.219871us	1176117.864832us
8 9 11 7 2 4 6 5 1 3	6 8 10 5 1 3 2 7 4 9	2 4 3 9 6 8 11 7 5 1
3 6 9 2 5 10 8 11 4 1	4 1 9 3 8 10 6 5 7 11	11 1 4 10 3 6 5 9 7 8
7 11 3 4 1 5 2 6 9 10	7 11 8 10 2 9 5 4 1 3	9 7 1 6 5 4 8 3 10 2
2 8 4 1 7 3 5 10 6 11	1 5 2 11 4 6 10 9 3 7	7 3 10 5 8 2 6 1 11 4
5 10 8 9 4 1 11 7 2 6	11 6 1 4 9 2 7 10 8 5	6 8 7 2 1 11 10 4 3 9
4 7 10 11 9 6 3 1 5 8	8 9 7 6 10 5 11 1 2 4	3 5 2 1 11 7 9 6 4 10
10 5 2 6 3 8 4 9 11 7	10 3 5 2 7 1 4 8 11 6	10 2 5 3 7 1 4 8 6 11
6 1 5 8 10 7 9 2 3 4	3 10 11 7 6 4 8 2 9 1	8 11 9 7 10 3 1 5 2 6
9 4 7 10 6 11 1 3 8 2	9 4 6 1 5 8 3 11 10 2	1 10 6 8 4 5 2 11 9 3
11 2 1 3 8 9 10 4 7 5	5 2 4 8 11 7 1 3 6 10	4 6 8 11 9 10 3 2 1 7

8.n=9，k=9

平均用时：837683μs

	805803.532357us	833218.097055us	874025.961712us
	5 7 4 3 2 9 1 8 6	4 6 3 1 5 7 8 9 2	7 2 4 1 5 9 3 8 6
	7 9 6 1 3 8 2 5 4	2 7 6 8 4 9 5 1 3	2 9 3 8 7 6 1 5 4
	9 2 1 5 4 3 8 6 7	5 4 2 3 6 1 9 8 7	4 1 9 2 6 5 8 3 7
	6 4 7 8 9 2 5 1 3	6 1 9 7 3 2 4 5 8	3 5 8 6 4 7 9 2 1
	8 3 2 6 1 4 9 7 5	1 9 8 2 7 5 3 4 6	8 7 6 5 3 4 2 1 9
	2 8 3 7 5 1 6 4 9	7 5 4 6 8 3 1 2 9	9 3 5 7 1 2 4 6 8
	3 1 5 9 7 6 4 2 8	8 3 7 9 1 4 2 6 5	5 6 1 9 8 3 7 4 2
	1 6 9 4 8 5 7 3 2	9 8 1 5 2 6 7 3 4	6 8 2 4 9 1 5 7 3
解矩阵：	4 5 8 2 6 7 3 9 1	3 2 5 4 9 8 6 7 1	1 4 7 3 2 8 6 9 5

9.n=9，k=9

平均用时：901521μs

	932068.943385us	933687.721356us	838804.868514us
	3 7 2 9 8 4 5 6 1	6 1 4 5 7 8 9 3 2	4 6 3 2 7 5 8 9 1
	7 1 3 5 2 6 8 4 9	9 8 3 7 5 6 2 1 4	6 9 8 3 2 4 5 1 7
	9 8 5 7 6 2 1 3 4	1 7 5 3 2 4 6 8 9	8 2 6 5 3 1 9 7 4
	4 6 1 8 5 3 7 9 2	3 4 7 9 1 2 8 6 5	5 1 9 7 4 8 6 2 3
	8 9 7 2 4 5 3 1 6	8 6 9 4 3 5 7 2 1	2 7 1 8 6 9 3 4 5
	6 5 8 1 7 9 4 2 3	2 5 1 6 8 9 3 4 7	3 5 2 4 1 6 7 8 9
	1 2 4 6 3 7 9 5 8	5 3 8 2 4 7 1 9 6	7 3 4 9 5 2 1 6 8
	5 3 6 4 9 1 2 8 7	7 9 2 8 6 1 4 5 3	9 4 7 1 8 3 2 5 6
解矩阵：	2 4 9 3 1 8 6 7 5	4 2 6 1 9 3 5 7 8	1 8 5 6 9 7 4 3 2

10.n=9，k=9

平均用时：811962μs

	807231.222321us	818615.545520us	810038.465578us
	1 2 4 8 5 3 7 6 9	5 7 2 1 6 3 8 9 4	1 7 6 8 3 5 4 9 2
	6 5 9 1 4 7 8 3 2	6 9 4 7 5 1 2 3 8	7 1 9 2 8 3 5 6 4
	2 7 3 9 1 5 6 8 4	3 2 6 9 8 4 1 5 7	6 5 8 7 9 4 1 2 3
	4 9 8 6 3 1 5 2 7	4 1 5 6 2 9 7 8 3	9 2 3 6 4 8 7 1 5
	3 4 6 5 8 2 9 7 1	9 3 7 2 4 8 5 6 1	2 8 4 5 6 9 3 7 1
	9 8 7 2 6 4 3 1 5	1 8 3 4 9 2 6 7 5	4 9 1 3 2 7 8 5 6
	7 6 1 3 2 9 4 5 8	2 6 8 5 3 7 4 1 9	3 4 5 9 1 2 6 8 7
	5 3 2 7 9 8 1 4 6	8 5 1 3 7 6 9 4 2	5 3 2 1 7 6 9 4 8
解矩阵：	8 1 5 4 7 6 2 9 3	7 4 9 8 1 5 3 2 6	8 6 7 4 5 1 2 3 9

11.n=9，k=9

平均用时：5605569μs

解矩阵：

820880.886532us	7935707.273597us	8060117.599351us
5 9 3 2 4 1 7 8 6	2 1 8 9 4 6 7 3 5	7 3 4 5 6 2 8 9 1
2 5 6 4 9 7 3 1 8	7 4 6 3 9 5 2 8 1	4 9 6 7 2 1 5 3 8
7 8 5 1 2 3 6 4 9	1 7 3 2 6 9 4 5 8	5 6 9 2 1 8 7 4 3
6 2 7 8 1 5 9 3 4	9 3 5 8 7 1 6 2 4	8 1 3 9 4 7 6 5 2
1 6 4 7 3 9 8 2 5	6 8 1 5 3 7 9 4 2	9 2 1 6 7 4 3 8 5
4 1 9 3 8 2 5 6 7	3 9 2 1 5 4 8 7 6	6 8 2 1 3 5 9 7 4
8 7 1 5 6 4 2 9 3	5 6 7 4 2 8 1 9 3	2 5 8 4 9 3 1 6 7
3 4 8 9 7 6 1 5 2	4 2 9 6 8 3 5 1 7	1 7 5 3 8 9 4 2 6
9 3 2 6 5 8 4 7 1	8 5 4 7 1 2 3 6 9	3 4 7 8 5 6 2 1 9

12.n=9，k=9

平均用时：8207649μs

解矩阵：

8277646.131270us	8129912.128691us	8215387.908651us
3 1 4 2 5 6 7 9 8	3 7 6 1 5 9 4 2 8	8 9 6 4 2 3 7 5 1
5 6 8 1 7 3 2 4 9	7 2 5 3 6 4 8 1 9	2 6 4 9 8 7 5 1 3
7 4 1 9 2 8 6 3 5	2 1 3 7 4 8 9 6 5	3 1 7 5 6 9 8 4 2
2 7 3 5 4 9 1 8 6	4 9 8 2 7 1 3 5 6	4 5 2 1 3 8 6 7 9
6 9 2 8 3 1 5 7 4	8 5 4 6 9 7 2 3 1	5 3 8 6 1 4 9 2 7
8 2 5 3 6 4 9 1 7	5 6 9 8 3 2 1 7 4	6 8 9 7 5 2 1 3 4
4 8 7 6 9 2 3 5 1	9 4 2 5 1 3 6 8 7	9 7 1 2 4 5 3 8 6
9 5 6 4 1 7 8 2 3	6 3 1 9 8 5 7 4 2	7 2 3 8 9 1 4 6 5
1 3 9 7 8 5 4 6 2	1 8 7 4 2 6 5 9 3	1 4 5 3 7 6 2 9 8

2.算法思想

一、前向检验

在基础的迭代回溯算法中，加入前向检验的技术，即在移动到新状态时，将未赋值变量的值域中会与当前赋值产生冲突的值从值域中去掉，以避免不必要的错误赋值。

二、模拟退火

在爬山法的搜索算法中，由于传统爬山法存在容易陷入局部最优的问题，故借鉴冶金上金属冷却的规律，在爬山法的内层循环中加入一条新的规则：即使新状态的估值 g(x)并没有原状态好，但存在一个小于 1 的概率，使得程序接受该状态。此概率满足金属冷却规律：$p = e^{-\Delta g/T}$，其中 T 为冷却时间，可见随着金属冷却（时间增加），接受差状态的概率 p 会越来越小，最后趋于最优解。

3.关键代码说明

一、前向检验

```
double run_time;
_LARGE_INTEGER time_start;  //开始时间
_LARGE_INTEGER time_over;   //结束时间
double dqFreq;      //计时器频率
LARGE_INTEGER f;    //计时器频率
QueryPerformanceFrequency(&f);
dqFreq=(double)f.QuadPart;
```

```
QueryPerformanceCounter(&time_start);//计时开始
ret=CSP_BACKTRACKING(a,D);
QueryPerformanceCounter(&time_over);//计时结束
run_time=1000000*(time_over.QuadPart-time_start.QuadPart)/dqFreq;
fpw=fopen("forward_solution.txt","w");//向文件中写入实验结果
fprintf(fpw,"%fus\n",run_time);
for(i=0;i<n;i++){
    for(j=0;j<n;j++) fprintf(fpw,"%d ",a[i][j]);
    fprintf(fpw,"\n");
}
```

此部分用于算法的计时以及使用示范。需要包含头文件 windows.h。

```
int a[10][10];
int D[10][10][11]; //用于存储变量的值域
```

a 数组存储矩阵当前赋值，D 数组用于存储变量的值域。

```
for(i=0;i<n;i++){
    for(j=0;j<n;j++){
        if(!a[i][j]){//是变量
            for(m=0;m<k;m++) D[i][j][m]=1;
        }
    }
}
for(i=0;i<n;i++){//初始值域设定
    for(j=0;j<n;j++){
        if(a[i][j]){
            for(p=0;p<n;p++){
                for(q=0;q<n;q++){
                    if(!a[p][q]){
                        if(p==i||q==j) D[p][q][a[i][j]-1]=0;
                    }
                }
            }
        }
    }
}
```

此部分在从文件读取数组初始赋值后对变量的初始可取值域进行初始化。同一行同一列若已有某个数字，则该数字对应的值域变为不可取。

```c
int CSP_BACKTRACKING(int a[][10],int D[][10][11]){//算法核心部分
	int i,j,m,flag;
	int change[10][10];//用于标识值域是否在fowardchecking中修改过
	memset(change,0,sizeof(change));
	if(completecheck(a)) return 1;//若赋值已完全则返回
	flag=0;
	for(i=0;i<n;i++){
		for(j=0;j<n;j++){
			if(!a[i][j]){
				flag=1;
				break;
			}
		}
		if(flag) break;
	}
	for(m=0;m<k;m++){
		if(D[i][j][m]){
			a[i][j]=m+1;//赋值
			fowardchecking(a,D,change,i,j,m+1);
			if(Dcheck(a,D))
				if(CSP_BACKTRACKING(a,D)) return 1;
			a[i][j]=0;
			deforward(D,change);
			memset(change,0,sizeof(change));
		}
	}
	return 0;//当前赋值无解
}
```

此部分为算法核心部分，采用的是基础的迭代回溯算法框架，其中 change 数组用来标记在前向检验中改变过的值域，以便在撤回赋值时也将错误的前向检验去除的值域复原。
completecheck 函数用于检查当前赋值是否已完全，是迭代的最底层。
之后按逐行逐列的顺序找到一下个待赋值变量，根据其当前可取值域，按顺序赋值，进行前向检验；Dcheck 函数用于检验剩余变量的值域是否都非空，若有变量值域已空，说明此路不通，需撤回；反之则进一步迭代。若迭代底层传上失败信号，也撤回当前赋值。撤回时还需要根据 change 数组记录的信息将前向检验中改变的值域复原。
最后，若当前变量所有赋值都不通，则返回 0 告知迭代上层失败信息。

其中用到的重要函数:

```
void fowardchecking(int a[][10],int D[][10][11],int change[][10],int p,int q,int r){
    int i,j,m;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(!a[i][j]){//变量
                if((i==p||j==q)&&D[i][j][r-1]==1){
                    D[i][j][r-1]=0;//与刚刚赋值在同一行或同一列，缩小相应值域
                    change[i][j]=r;//记录更改
                }
            }
        }
    }
}
```

前向检验函数，change 数组用于记录，p、q、r 分别代表当前赋值的横坐标、纵坐标和所赋的值。循环找到未赋值变量，若该变量与当前赋值的变量在同一行或同一列且该值在值域中可取，则将该值设为不可取，同时用 change 数组记录下改变的位置和改变的值。

```
void deforward(int D[][10][11],int change[][10]){//用于在回溯时将之前前向检验的值域变回
    int i,j,m;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(change[i][j]) D[i][j][change[i][j]-1]=1;
        }
    }
}
```

前向检验撤回函数，若当前赋值被撤回，则根据 change 数组记录的信息，将对应的值域变化复原。

二、模拟退火

```
srand((unsigned)time(NULL));
double run_time;
_LARGE_INTEGER time_start;  //开始时间
_LARGE_INTEGER time_over;   //结束时间
double dqFreq;      //计时器频率
LARGE_INTEGER f;    //计时器频率
QueryPerformanceFrequency(&f);
dqFreq=(double)f.QuadPart;
```

```
QueryPerformanceCounter(&time_start);//计时开始
Simulated_Annealing(a,fixed);
QueryPerformanceCounter(&time_over);//计时结束
run_time=1000000*(time_over.QuadPart-time_start.QuadPart)/dqFreq;
fpw=fopen("annealing_solution.txt","w");//向文件中写入实验结果
fprintf(fpw,"%fus\n",run_time);
for(i=0;i<n;i++){
    for(j=0;j<n;j++) fprintf(fpw,"%d ",a[i][j]);
    fprintf(fpw,"\n");
}
```

此部分用于算法的计时以及使用示范。需要包含头文件 windows.h。

```
int a[10][10];
int fixed[10][10];//用于标识是否为初始格子
```

a 数组用存储矩阵当前赋值，fixed 数组用来标识矩阵中的某一个元素是常量还是变量，0 为变量，1 为常量。

```
for(i=0;i<n;i++)
    for(j=0;j<n;j++) fscanf(fpr,"%d",&a[i][j]);//读取初始矩阵内容
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        if(a[i][j]) fixed[i][j]=1;//是固定值
```

此部分用于从文件中读取矩阵初始状态，并记录 fixed 数组的值。

```
void Simulated_Annealing(int a[][10],int fixed[][10]){//算法核心部分
    double T=5.0;//初始化T为5
    int i,j,p,q,r,delta,temp;
    int oldv,newv;
    for(i=0;i<n;i++){//赋初值，此处为对所有0填入1
        for(j=0;j<n;j++)
        if(!a[i][j]) a[i][j]=1;
    }
    while(T>0){//循环直到温度足够低
        oldv=value(a);
        if(oldv==2*n*n*n-2*n*n) return;//已经互不相同，返回
        p=rand()%n;
        q=rand()%n;
        r=1+rand()%k;//先产生一个随机方向
        while(fixed[p][q]==1||a[p][q]==r){//该方向必须合法
            p=rand()%n;
            q=rand()%n;
            r=1+rand()%k;//若不合法则重新生成随机方向
        }
        temp=a[p][q];
        a[p][q]=r;
        newv=value(a);
        delta=newv-oldv;
        if(delta<=0&&((float)(rand()%1000)/1000)>=exp(delta/T)){
            a[p][q]=temp;
        }
        T-=0.000005;//温度下降
    }
    printf("timeout! \n");
    return;
}
```

此部分为算法的核心部分，采用爬山法的框架，在内层循环中加入一条概率判断，即有概率采取较差的状态作为新状态。T 为循环运行时间，value 函数用于计算当前赋值的好坏，oldv、newv 分别是旧状态、新状态的 value。在循环中，每次循环中，首先计算当前赋值的 value 作为旧状态的 value，若该 value 已是最佳 value（后在 value 函数中有解释），直接返回；再随机生成一个随机的移动方向（若移动方向不合法则重新生成，直到合法为止），直接赋值成为新状态，计算新状态的 value，与旧状态相比，若更好则采纳；若不如旧状态则变回原来的状态，但有概率 p 仍能接受较差的新状态。循环的最后 T 进行自减（温度下降）。

其中用到的 value 函数:

```
int value(int a[][10]){//用于评估当前填法的优劣
    int v=0;
    int i,j,p,q;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            for(p=0;p<n;p++){
                if(p!=i&&(a[p][j]!=a[i][j])) v++;
            }
            for(q=0;q<n;q++){
                if(q!=j&&(a[i][q]!=a[i][j])) v++;
            }
        }
    }
    return v;
}
```

在这个函数中，我采用了矩阵中处于同行同列的取值互不相同的元素数作为 value 计算。在最佳情况下，所有同行同列的元素都互不相同，那么每个元素对 value 的贡献为 2*（n-1），共有 n*n 个元素，故最佳 value 值应为 2*n*n*n-2*n*n。

4.如何提升速度

一、前向检验

采取更好的赋值次序。

二、模拟退火

在计算测试样例时，对后面的样例有时会出现超时的情况，这可能是由于随机性导致陷入局部最佳，也有可能是冷却时间不够长。为了提升算法速度，可能的改进有：在对矩阵赋初值时根据矩阵初始特征进行赋初值、采用更好的 value 函数、采用更好的温度 T 冷却函数等等。