# 《算法分析与设计》
# 课后作业

作业编号 作业 6

学　　号 2020112257

姓　　名 刘岩

专　　业 软件工程

学　　院 计算机与人工智能学院

二 0 二二年三月

# 第 7 章 回溯法

**总体要求：**

（1） 采用回溯法实现下述题目要求。

（2） 写出求解过程中的目标函数，限界函数（如果有的话）以及约束函数。

（3） 任选一种语言 C/C++编程实现题目要求。

（4） 画出样例输入时的解空间树以及搜索空间树，定义每个结点对应变量的含义以及搜索空间树上结点对应变量的值。

（5） 分析算法的时间复杂度。

（6） 分别以 Word 文档和 pdf 方式提交，并打包压缩后以.rar 文件提交。

题目 1：数独游戏是在 9*9 的方格中填放 1~9 的数字，要求每一行、每一列以及 3*3 的方格中的数字均不能相同，如下图所示。

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 3 | 6 | 2 | 8 | 5 | 7 | 9 |
| 5 | 7 | 2 | 1 | 3 | 9 | 4 | 6 | 8 |
| 9 | 8 | 6 | 7 | 5 | 4 | 2 | 3 | 1 |
| 3 | 9 | 1 | 5 | 4 | 2 | 7 | 8 | 6 |
| 4 | 6 | 8 | 9 | 1 | 7 | 3 | 5 | 2 |
| 7 | 2 | 5 | 8 | 6 | 3 | 9 | 1 | 4 |
| 2 | 3 | 7 | 4 | 8 | 1 | 6 | 9 | 5 |
| 6 | 1 | 9 | 2 | 7 | 5 | 8 | 4 | 3 |
| 8 | 5 | 4 | 3 | 9 | 6 | 1 | 2 | 7 |

现方格中某些数字为空缺（用 0 表示），希望你编写程序能够将空缺的数字补齐。

输入要求：

输入包含 9 行，每一行包含 9 个数字，对应每个方格中的数字，0 表示该方格的数字为空。

输出要求：

输出为 9 行，每行 9 个数字，也就是补齐后的 9*9 的方格中的数字。

样例输入：

103000509

002109400

000704000

300502006

060000050

700803004

000401000

009205800

804000107

样例输出：

143628579

572139468

986754231

391542786

468917352

725863914

237481695

619275843

854396127

（2）写出求解过程中的目标函数，限界函数（如果有的话）以及约束函数。

目标函数：count==81，即所有格都被填充；

约束函数：differ（），保证每一行、每一列以及 3*3 的方格中的数字均不能相同

（3）任选一种语言 C/C++编程实现题目要求。

```
#include<iostream>
using namespace std;
int map[9][9];
int differ(int count)
{
    int row = count / 9;
    int col = count % 9;
    int tempR=row/3*3;
    int tempC=col/3*3;
    for(int j=0;j<9;j++)
    {
```

```
        if(map[row][j]==map[row][col]&&j!=col)
        return 0;
    }
    for (int j=0;j<9;j++)
    {
        if(map[row][col]==map[j][col]&&j!=row)
        return 0;
    }
    for(int j=tempR;j<tempR+3;j++)
    {
        for(int k=tempC;k<tempC+3;k++)
        {
            if(map[row][col]==map[j][k]&&row !=j&&col!=k)
            return 0;
        }
    }
    return 1;
}
void Block_padding(int count)
{
    int row = count / 9;
    int col = count % 9;
    if(count==81)
    {
        for(int i=0;i<9;i++)
        {
            for (int j = 0; j < 9; j++)
              cout<<map[i][j]<<" ";
            cout<<endl;
        }
        return;
    }
  if(map[row][col]==0)
  {
    for(int i=1;i<=9;i++)
    {
        map[row][col]=i;
        if(differ(count))
         Block_padding(count+1);
         map[row][col]=0;
    }
  }
```

```
  else{
    Block_padding(count + 1);
  }
}
int main()
{
    for(int i=0;i<9;i++)
        for(int j=0;j<9;j++)
            cin>>map[i][j];
    Block_padding(0);
    return 0;
}
```

（4）画出样例输入时的解空间树以及搜索空间树，定义每个结点对应变量的含义以及搜索空间树上结点对应变量的值。

6-1解空间树.vsdx

6-1搜索空间树.vsdx

（5）分析算法的时间复杂度。

Differ 函数与数组中空缺数量有关，由解空间树可知，算法时间复杂度为 O（9^n）

题目 2： 一个人站在一个由黑白方格构成的矩形区域中，他从某个黑色方格出发，向上下左右四个方向移动，每次只能从一个黑色方格移动到另一个黑色方格，问他最多能够走多少个黑色方格。

输入要求：

输入包含多组数据。每组数据的第一行包含两个整数 l 和 w，表示矩形区域的长度和宽度，也就是水平和垂直方向上方格的数量。其后的 w 行，每一行有 l 个字符，由“.”，”#”和“@”符号构成，分别表示黑色方格，白色方格和人起始所在的黑色方格。如果 l 和 w 为 0 则表示输入结束。

输出要求：

对于每一组测试数据，输出一个整数，表示他最多能走的黑色方格的数量，占一行。

样例输入：

```
11  6
..#..#..#..
..#..#..#..
```

```
..#..#..###
..#..#..#@.
..#..#..#..
..#..#..#..
7 7
..#.#..
..#.#..
###.###
...@...
###.###
..#.#..
..#.#..
0 0
```

样例输出：

```
6
13
```

（2）写出求解过程中的目标函数，限界函数（如果有的话）以及约束函数。

目标函数：way++，即找出所能走的最多黑格

约束函数：

```
if (map[x - 1][y] == '.' && visit[x - 1][y] == 0 && x - 1 >=0)
if (map[x + 1][y] == '.' && visit[x + 1][y] == 0 && x + 1 < n)
if (map[x][y - 1] == '.' && visit[x][y - 1] == 0 && y - 1 >=0)
if (map[x][y + 1] == '.' && visit[x][y + 1] == 0 && y + 1 < m)
```

（3）任选一种语言 C/C++编程实现题目要求。

```
#include<iostream>
using namespace std;
#define max_n 99
char map[max_n][max_n]={0};
bool visit[max_n][max_n]={0};
int way=0;
int m=0,n=0;
int x,y;//人的起始坐标
void Find_maxway(int x,int y)
{
    way++;
    visit[x][y] = 1;
    if (map[x - 1][y] == '.' && visit[x - 1][y] == 0 && x - 1 >=
0)
```

```
        Find_maxway(x - 1, y);
    if (map[x + 1][y] == '.' && visit[x + 1][y] == 0 && x + 1 < n)
        Find_maxway(x + 1, y);
    if (map[x][y - 1] == '.' && visit[x][y - 1] == 0 && y - 1 >=
0)
        Find_maxway(x, y - 1);
    if (map[x][y + 1] == '.' && visit[x][y + 1] == 0 && y + 1 < m)
        Find_maxway(x, y + 1);
    return;
}
int main()
{
    cin>>m>>n;
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)
        {
            cin>>map[i][j];
            if(map[i][j] == '@')
                x=i;y=j;
        }
    Find_maxway(x,y);
    cout << way + 1 << endl;
    system("pause");
    return 0;
}
```

（4）画出样例输入时的解空间树以及搜索空间树，定义每个结点对应变量的含义以及搜索空间树上结点对应变量的值。

样例 1：

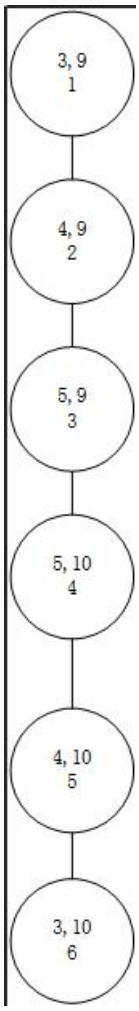
解空间树：

节点定义：当前位置坐标和已经走过的格子数

6-2解空间树.vsdx

搜索空间树：

节点的含义：当前位置坐标和已经走过的格子数

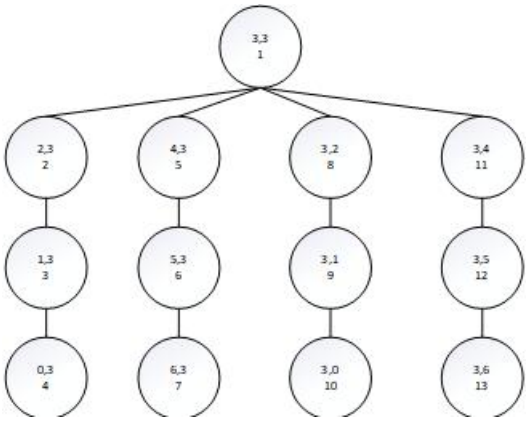



样例 2：

解空间树：

6-2解空间树2.vsd
x

搜索空间树：



（5）分析算法的时间复杂度。

L 行 w 列，考虑最大情况，即全部均为黑格，所以时间复杂度为 O（lw）

评分标准：（总分 100 分，每题按 100 分评分）

1. 采用回溯法实现。（10 分）
2. 算法求解过程中的目标函数，限界函数以及约束函数：（20 分）

（1） 目标函数描述准确。（10 分）

（2） 限界函数以及约束条件描述准确。（10 分）

3. C/C++程序要求（30 分）

（1） 程序编译、运行正确无误。（5 分）

（2） 程序书写规范，变量、函数定义符合规范。（5 分）

（3） 程序与算法求解过程一致，且符合题目要求（20 分）

4. 样例输入时的解空间树以及搜索空间树,结点代表的变量的含义以及搜索空间树上结点的值(30 分)。

（1） 结点对应的变量定义准确。（5 分）

（2） 解空间树结构正确。（10 分）

（3） 搜索空间树正确，对应结点的值正确。（15 分）

5. 算法复杂度分析要求（10 分）

（1） 算法复杂度分析方法及过程正确。（5 分）

（2） 算法复杂度分析结果正确。（5 分）

提交截止时间：2022 年 6 月 11 日 20：00，每延后一天扣 20 分。

提交方式：电子稿，命名规则：学号_姓名_作业 6.rar

邮件发送给：hyhuang@home.swjtu.edu.cn