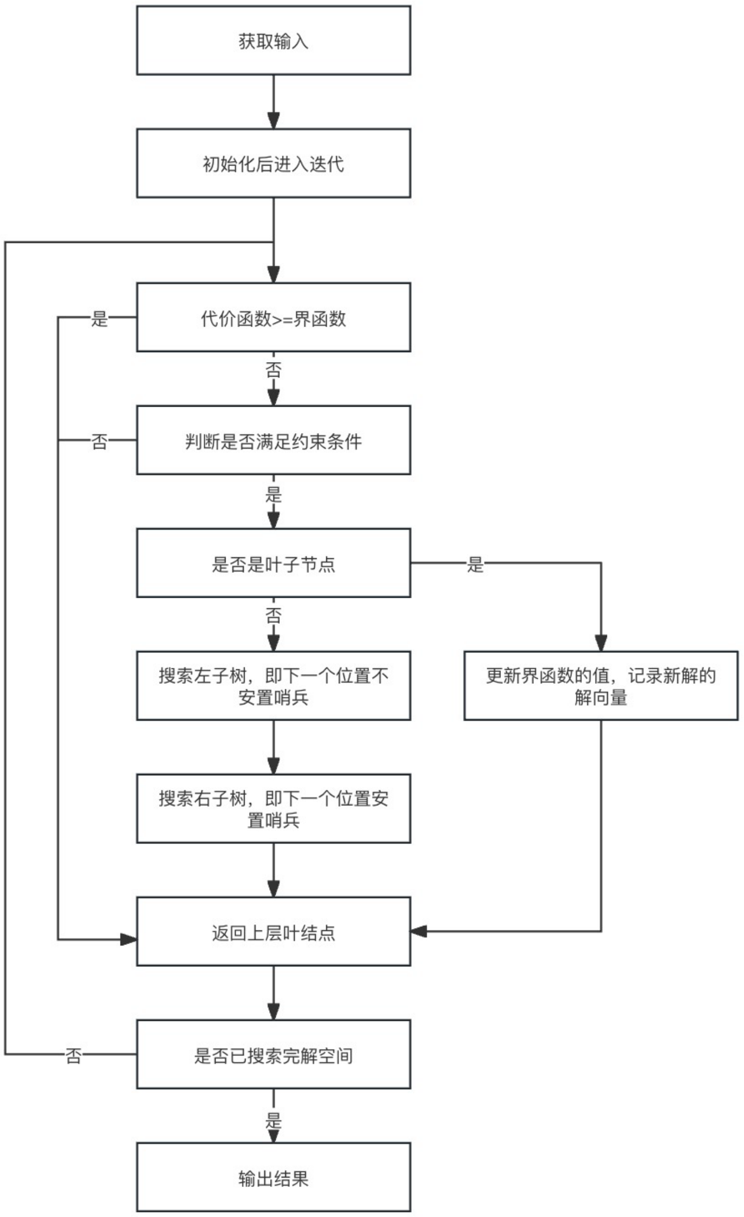
算法设计与分析第 5 章实验作业

林宇浩 21311274

一、实现思想



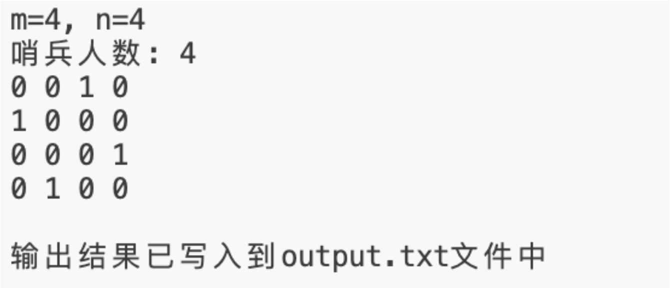
解向量为<x11, x12, x13, … , x21, x22, x23, … , xmn>，xij=1 代表位于 i 行 j 列的陈列室布置哨兵，xij=0 则代表不在此位置的陈列室布置哨兵。

约束条件为所有陈列室都在哨兵的监视之下。

界函数为此结点之前已找到的方案中哨兵数的最小值。

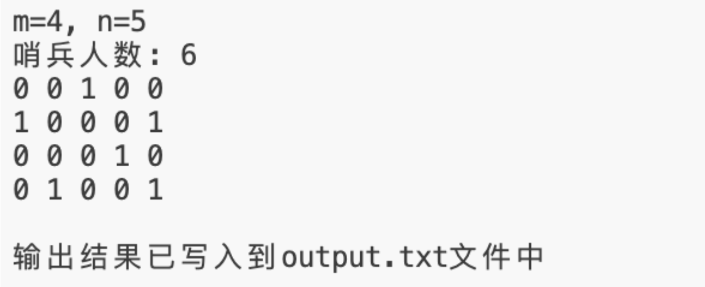
代价函数为到当前结点时，已布置的哨兵数量，加上剩余还未考虑的陈列室数量除以 5。这是当前结点所能达到的哨兵数量的下界。即对于剩下还未考虑的陈列室，即使哨兵之间监视的陈列室没有重复，不会出现一个陈列室被多个哨兵监视的浪费，也还要剩余陈列室数量除以 5 这么多哨兵才够。

二、输出截图



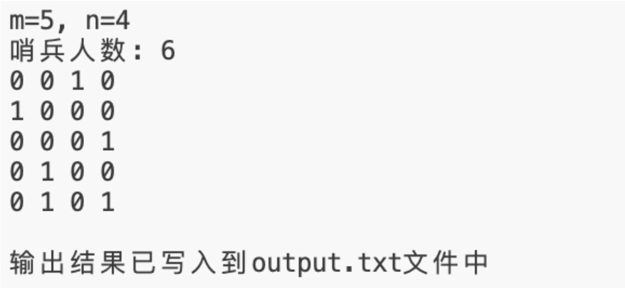
```
m=4, n=4
哨兵人数: 4
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0

输出结果已写入到output.txt文件中
```



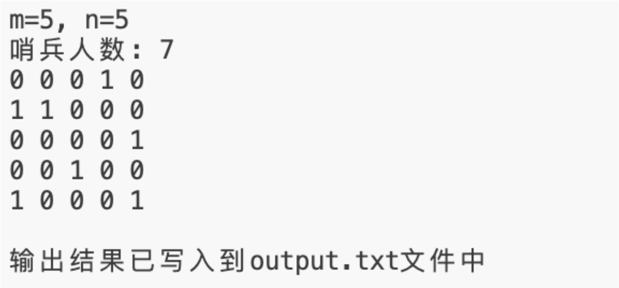
```
m=4, n=5
哨兵人数: 6
0 0 1 0 0
1 0 0 0 1
0 0 0 1 0
0 1 0 0 1

输出结果已写入到output.txt文件中
```



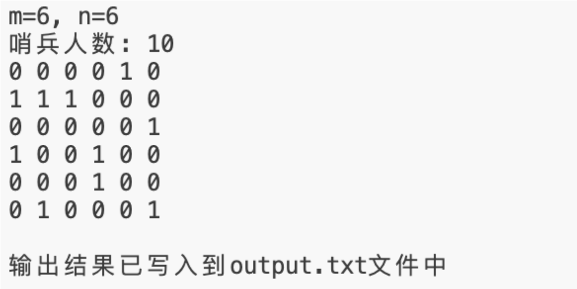
```
m=5, n=4
哨兵人数: 6
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
0 1 0 1

输出结果已写入到output.txt文件中
```



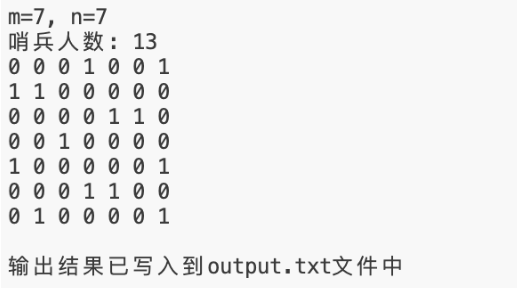
```
m=5, n=5
哨兵人数: 7
0 0 0 1 0
1 1 0 0 0
0 0 0 0 1
0 0 1 0 0
1 0 0 0 1

输出结果已写入到output.txt文件中
```



```
m=6, n=6
哨兵人数: 10
0 0 0 0 1 0
1 1 1 0 0 0
0 0 0 0 0 1
1 0 0 1 0 0
0 0 0 1 0 0
0 1 0 0 0 1

输出结果已写入到output.txt文件中
```



```
m=7, n=7
哨兵人数: 13
0 0 0 1 0 0 1
1 1 0 0 0 0 0
0 0 0 0 1 1 0
0 0 1 0 0 0 0
1 0 0 0 0 0 1
0 0 0 1 1 0 0
0 1 0 0 0 0 1

输出结果已写入到output.txt文件中
```

不同条件下的用时比较：
代码 1：仅在叶结点处判断是否满足约束条件，代价函数为当前结点已布置的哨兵数量。
代码 2：更改约束条件为，已考虑过的陈列室都需要在哨兵的监视之下，边界位置和还未考虑的陈列室可以不在哨兵监视之下，在每个结点都检查约束条件。代价函数仍为当前结点已布置的哨兵数量。
代码 3：在代码 2 的基础上，将代价函数改为已布置的哨兵数量，加上剩余还未考虑的陈列室数量除以 5。这样约束变得更强。

下面是三种代码耗时的比较，单位为秒：

	4*4	5*4	4*5	5*5	5*6
代码 1	0.0716	1.3755	1.0386	24.3967	474.4282
代码 2	0.0095	0.1394	0.1479	0.8146	5.8421
代码 3	0.0014	0.0041	0.0048	0.0050	0.0521

可以看到，从代码 1 到代码 3 约束越来越强，不满足约束的可能性就越大，回溯的机会就越多，裁剪的分支数就越多，从而算法更快耗时更少。

三、源代码

```python
import copy
import math
import time

input_file_path = 'input.txt'

with open(input_file_path, 'r') as file:
    data = file.readline().split()
m = int(data[0])
n = int(data[1])


def index_plus_one(index):  # 下标递增
    if index[1] == n-1:
        return [index[0]+1, 0]
    return [index[0], index[1]+1]


def judge_satisfy_constraint(matrix):  # 判断叶结点是否满足约束条件
    for i in range(m):
        for j in range(n):
            have_guard = False
            if matrix[i][j] == 1:
                have_guard = True
                continue
            if i+1 <= m-1 and matrix[i+1][j] == 1:
                have_guard = True
                continue
            if i-1 >= 0 and matrix[i-1][j] == 1:
                have_guard = True
                continue
            if j+1 <= n-1 and matrix[i][j+1] == 1:
                have_guard = True
                continue
            if j-1 >= 0 and matrix[i][j-1] == 1:
                have_guard = True
                continue
            if have_guard == False:
                return False
    return True


def judge_satisfy_constraint_2(index, matrix):  # 判断非叶结点是否满足约束条件
    matrix = copy.deepcopy(matrix)
    index = copy.deepcopy(index)
    for i in range(n):
        matrix[index[0]][index[1]] = 1
        index = index_plus_one(index)
        if index[0] == m:
            break

    for i in range(index[0]):
        for j in range(n):
            have_guard = False
            if matrix[i][j] == 1:
                have_guard = True
                continue
            if i+1 <= m-1 and matrix[i+1][j] == 1:
                have_guard = True
                continue
            if i-1 >= 0 and matrix[i-1][j] == 1:
                have_guard = True
                continue
            if j+1 <= n-1 and matrix[i][j+1] == 1:
                have_guard = True
                continue
            if j-1 >= 0 and matrix[i][j-1] == 1:
                have_guard = True
                continue
            if have_guard == False:
                return False

    i = index[0]
    for j in range(index[1]):
        have_guard = False
        if matrix[i][j] == 1:
            have_guard = True
            continue
        if i+1 <= m-1 and matrix[i+1][j] == 1:
            have_guard = True
            continue
        if i-1 >= 0 and matrix[i-1][j] == 1:
            have_guard = True
            continue
        if j+1 <= n-1 and matrix[i][j+1] == 1:
            have_guard = True
            continue
        if j-1 >= 0 and matrix[i][j-1] == 1:
            have_guard = True
            continue
        if have_guard == False:
```

```python
            return False
    return True


min_guard_num = m*n  # 界函数初始值
min_guard_matrix = []


def find(index, matrix, guard_num):
    global min_guard_num, min_guard_matrix

    cost_func = guard_num + math.ceil((m*n-(index[0]*n+index[1]))/5)  # 能够达到的卫兵数量的下界
    if cost_func >= min_guard_num:  # 节点的代价超过了界函数
        return
    # if guard_num >= min_guard_num:  # 卫兵数量已经超过现有最优解
    #     return

    if index[0] == m:  # 叶子节点
        if judge_satisfy_constraint(matrix):
            min_guard_num = guard_num
            min_guard_matrix = matrix
            return
        return

    if judge_satisfy_constraint_2(index, matrix) == False:  # 判读是否满足约束条件
        return

    find(index_plus_one(index), matrix, guard_num)  # 下一个位置不安置哨兵

    matrix2 = copy.deepcopy(matrix)
    matrix2[index[0]][index[1]] = 1
    find(index_plus_one(index), matrix2, guard_num+1)  # 下一个位置安置哨兵


init_matrix = [[0 for i in range(n)]for j in range(m)]
start_time = time.time()
find([0, 0], init_matrix, 0)
end_time = time.time()

print(f'\nm={m}, n={n}')
print("哨兵人数:", min_guard_num)
print(f"耗时: {end_time-start_time:.4f}")
for line in min_guard_matrix:
    for num in line:
        print(num, end=' ')
    print('')

output_file_path = 'output.txt'
with open(output_file_path, 'w') as file:
    for line in min_guard_matrix:
        for num in line:
            file.write(str(num))
            file.write(' ')
        file.write('\n')

print(f"\n输出结果已写入到{output_file_path}文件中\n")
```