- 哨兵布置问题。一个博物馆由排成m*n个矩阵阵列的陈列室组成，需要在陈列室中设立哨位，每个哨位上的哨兵除了可以监视自己所在陈列室外，还可以监视他上、下、左、右四个陈列室，试基于分支限界法给出一个最佳哨位安排方法，使得所有陈列室都在监视之下，但使用的哨兵最少。
- 数据输入：由文件input.txt给出输入数据。第1行有2个正整数m和n。
- 结果输出：将计算出的哨兵人数及其最佳哨位安排输出到文件output.txt。文件的第1行是哨兵人数，接下来的m行中每行n个数，0表示无哨位，1表示哨位。

1. 实现思想

算法基于BFS（广度优先搜索）和分支界限法：使用广度优先搜索遍历可能的房间布局。每一步决策是在当前房间放置或不放置哨兵，并使用分支定界法优化搜索过程。

使用优先队列（PriorityQueue）按照估计的下界对节点进行排序，以优先探索潜在的最优解。 通过迭代遍历房间，每次决策是放置哨兵或者不放置哨兵。利用分支定界法，剪枝不可能导致最优解的搜索路径，提高算法效率。通过队列和回溯的方式搜索可能的解空间。

剪枝策略:

- 放置的哨兵个数不会超过nm/3+1个(按每个机器人仅辐射左右或上下考虑，堆叠这样的小长条可得)。所以nm/3+2为初始最优值，当放置的个数超过当前最优值时，剪去。
- 估计每个节点的下限为：当前已经放置的哨兵数x+剩余未控制到的房间/5（每个哨兵最多监控五个房间），若节点的下限仍然比当前最优值大，则可进行剪枝。

算法步骤

- 步骤 1：初始化。初始化一个全局变量 `count` 以记录已探索的节点数。将初始上界（`best_count`）设置为正无穷大。创建一个全零的初始布局矩阵。将 `lowest` 设置为一个启发式值，大致为总房间数的三分之一再加上2。创建一个优先队列（`pq`）以存储节点及其下界估计。

- 步骤 2：初始配置。计算当前布局的下界估计。

- 步骤3：主循环。当优先队列非空时，执行以下步骤：从优先队列中出队一个节点。计算当前布局中哨兵的数量。计算当前布局的下界估计。

- 步骤 4：剪枝。如果当前下界大于等于当前最优解数量值，则剪枝当前分支，继续下一轮迭代。

- 步骤 5：检查有效解。如果当前布局是一个有效解（所有房间都被覆盖），且哨兵数量小于当前最优解数量，则更新最优解数量，并记录当前布局为最佳布局。

- 步骤 6：探索节点。如果当前节点不是最后一个房间，将以下节点入队：一个将当前房间标记为未占用的节点；一个将当前房间标记为占用的节点。

- 步骤 7：更新计数和回溯。增加已探索节点的计数。若当前节点不是最后一个房间，更新用于回溯的布局矩阵。

- 步骤 8：返回最佳解。一旦优先队列为空，返回找到的最佳布局。

整个搜索空间为一棵二叉树，每个节点对应一个放置哨兵的策略，每个非叶节点都有两个子节点：分别对应与在某个位置放置哨兵与不放置哨兵的策略。所以该算法的最差情况之下需要探索（2^mn）个节点个数，是指数级增长的算法。但是通过分支界限能够大大减少需要探索的数量。

2. 输出截图

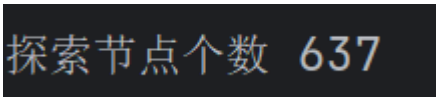
按照题目要求在input.txt文件中设置m，n的值对算法进行测试：

m=n=3，output文件如下图所示：

原始需要探索的节点数为2^3*3=2^9=512,输出算法探索的节点数为：

m=n=4，output文件如下图所示：

原始需要探索的节点数为2^4*4=2^16=65536,输出算法探索的节点数为：



m=n=5，output文件如下图所示：

原始需要探索的节点数为 2^5*5=2^25=33554432,输出算法探索的节点数为：

可以看到分支界限剪枝取得了很好的效果，减少了许多不必要探索的节点。

3. 源代码

```
判断房间布局是否有效的函数
def is_valid(arrangement, m, n):
    num = 0
    for i in range(m):
        for j in range(n):
            if arrangement[i][j] == 0:
                # 检查上方、下方、左边、右边是否有哨兵，若都没有则是一个未监控的房间
                if i > 0 and arrangement[i - 1][j] == 1:
                    continue
                if i < m - 1 and arrangement[i + 1][j] == 1:
                    continue
                if j > 0 and arrangement[i][j - 1] == 1:
                    continue
                if j < n - 1 and arrangement[i][j + 1] == 1:
                    continue
                num += 1
    return num
```

```
# 统计一个布局中哨兵的数量的函数
def count_sentries(arrangement):
    return sum(row.count(1) for row in arrangement)

# 估算哨兵数量下界的函数
def estimate_lower_bound(arrangement, m, n):
    remaining_rooms = is_valid(arrangement, m, n)
    return count_sentries(arrangement) + remaining_rooms / 5
```

```
# 使用BFS和分支定界法的主函数
def bfs_branch_and_bound(m, n):
    global count  # 用于计数已探索的节点数的变量
    count = 1
    best_count = float('inf')
    best_arrangement = None
    lowest = round(m * n / 3) + 2

    arrangement = [[0] * n for _ in range(m)]
    best_arrangement = [row[:] for row in arrangement]
    pq = PriorityQueue()
    pq.put((estimate_lower_bound(arrangement, m, n), arrangement, 0, 0))

    while not pq.empty():
        _, arrangement, row, col = pq.get()
        current_count = count_sentries(arrangement)
        lower_bound = estimate_lower_bound(arrangement, m, n)

        if lower_bound >= min(best_count, lowest):
            # 如果当前节点的下界不比全局最优解和最低估计值小，放弃此节点
            continue

        if is_valid(arrangement, m, n) == 0 and current_count < best_count:
            # 如果当前节点是合法解且哨兵数量更少，更新最优解
            best_count = current_count
            best_arrangement = [row[:] for row in arrangement]

        if row == m - 1 and col == n - 1:
            # 如果已到达最后一个房间，继续下一个节点
            continue
        else:
            next_row = row + 1 if col == n - 1 else row
            next_col = 0 if col == n - 1 else col + 1

            # 将下一个节点加入队列
            pq.put((estimate_lower_bound(arrangement, m, n), arrangement,
next_row, next_col))
            arrangement[row][col] = 1  # 尝试在当前房间放置哨兵
            pq.put((estimate_lower_bound(arrangement, m, n), [row[:] for row in
arrangement],
                    next_row, next_col))
            arrangement[row][col] = 0  # 回溯，尝试不在当前房间放置哨兵

        count += 1

    return best_arrangement
```

```
# 主程序
if __name__ == "__main__":
```

```
    with open("input.txt", "r") as input_file:
        m, n = map(int, input_file.readline().split())

    museum_arrangement = bfs_branch_and_bound(m, n)
    print("探索节点个数", count)
    with open("output.txt", "w") as output_file:
        output_file.write("最少哨兵数量：  " +
str(count_sentries(museum_arrangement)) + "\n")
        for row in museum_arrangement:
            output_file.write(" ".join(map(str, row)) + "\n")
```