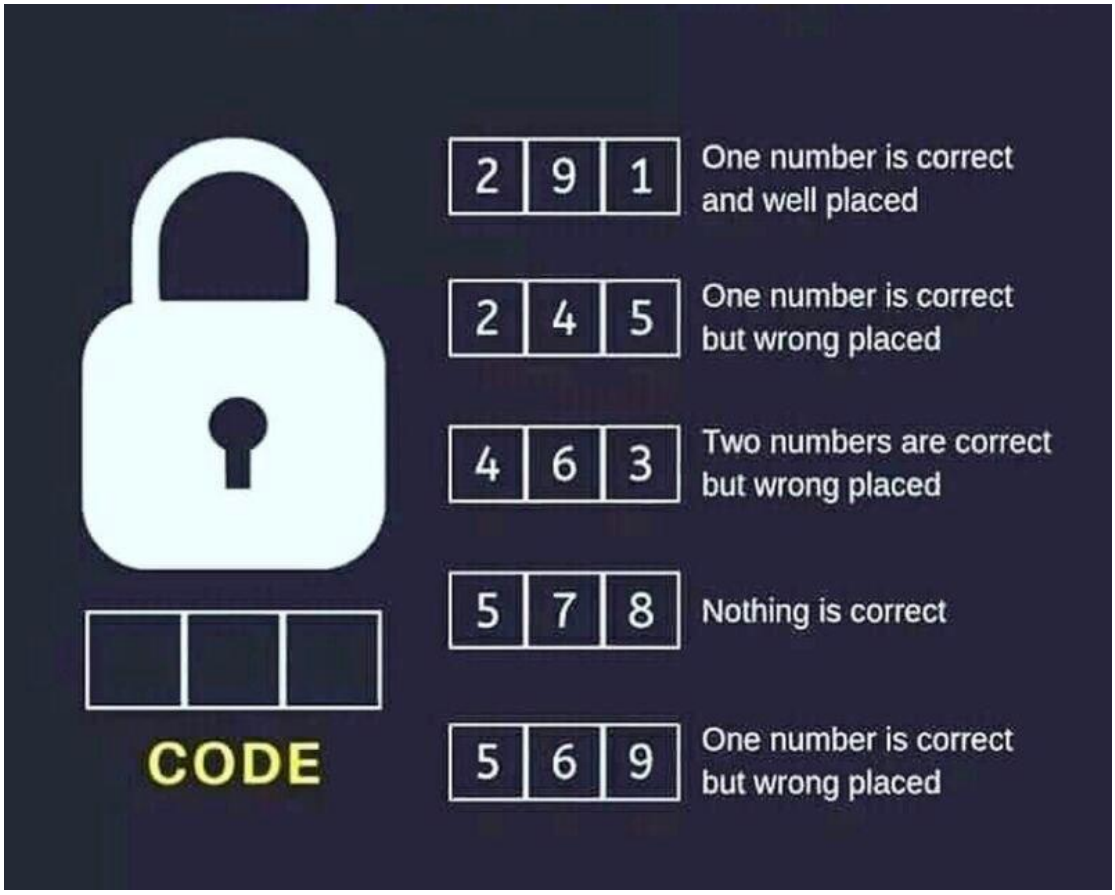
Z3 求解器解题思路

Z3 求解器是一个 SMT 求解器，用于检查逻辑表达式的可满足行，可以找到一组约束中的其中一个可行解。

Z3 支持布尔运算符：And、Or、Not、Implis、If（if-then-else）



我们可以先初始化一个 z3 求解器，再初始化密码向量 code 为三位数字，用于表示**可能的正确密码序列**，在 z3 求解器中添加条件（add）来约束 code 取值。

重点：如何表示数字正确但是位置错误。可以**定义一个函数 Goodvaluebadplace（nums，count）中，来约束 code 的取值。**

```python
def GoodValueBadPlace(nums, count):
    exps = []
    for i in range(len(nums)):
        for j in range(len(code)):
            if i == j:
                # 不满足位置不同的条件
                continue
            # 满足位置不同的条件
            else:
                # 满足数字正确的条件
                # If函数是z3中的一个布尔运算符(if-then-else)
                exps.append(If(nums[i] == code[j], b: 1, c: 0))
    # 如果有count个数字符合条件,则返回true,否则当前code中没有满足该条件的值
    return Sum(exps) == count
```

Add（Goodvaluebadplace（nums，count））：求解器会尝试找到一组满足约束条件的 code 的取值，从而获得符合条件的解。

将图片上的条件输入（add）到求解器中，求解器就可以输出一个符合条件的答案。

```python
# 有一个数字正确且在正确位置
s.add(Or( *args: code[0] == 2, code[1] == 9, code[2] == 1))

# 每个数字的范围0-9
for i in range(3):
    s.add(And( *args: 0 <= code[i], code[i] <= 9))
#有一个数字正确但位置错误
s.add(GoodValueBadPlace( nums: [2, 4, 5], count: 1))
# 两个数字正确但位置错误
s.add(GoodValueBadPlace( nums: [4, 6, 3], count: 2))
# 一个数字正确但位置错误
s.add(GoodValueBadPlace( nums: [5, 6, 9], count: 1))
# 没有数字正确
s.add(And( *args: code[0] != 5, code[1] != 7, code[2] != 8))
```