**Homework 2**

309553048\_陳柏丞

* **Execution：**
  + Python 3.6.9
    - python3 main.py -i <inputfile> -o <outputfile>
* **Abstract：**
  + Functions：
    - getDigit()
    - getGrid()
    - get\_val()
    - display()
    - parse\_grid()
    - solve()
    - main()
  + Main：
    - Read filename for inputFile, outputFile
    - Get n from inputFile, and get DIGITS
    - Value, row&column, Box constraints
    - getGrid()
    - solve it
* **Implementation：**

def getDigit(n):

tmp = ''

for i in range(1, n+1):

tmp += str(i)

return tmp

def getGrid(data):

data2D = []

for ele in data:

data2D.append(ele.split())

grid = ''

for i in range(len(data2D)):

for j in range(len(data2D)):

if data2D[i][j] == '0':

grid+='.'

continue

elif j == sq\_n:

grid+='|'

grid+=data2D[i][j]

return grid

def get\_val(point, r, c):

for v in range(1, n+1):

if point[X[r, c, v]]:

return DIGITS[v-1]

return "X"

def display(point, f):

if point == None:

f.write('0')

print(0)

else:

f.write('1')

print(1)

def parse\_grid(grid):

chars = [c for c in grid if c in DIGITS or c in "0. "]

assert len(chars) == n\*\*2

return And(\*[ X[i//n+1, i%n+1, int(c)]

for i, c in enumerate(chars) if c in DIGITS ])

def solve(grid):

with parse\_grid(grid):

return S.satisfy\_one()

def main(argv):

try:

opts, args = getopt.getopt(argv, "hi:o:", ["ifile=", "ofile="])

except getopt.GetoptError:

print(f'test.py -i <inputfile> -o <outputfile>')

for opt, arg in opts:

if opt == '-h':

print(f'test.py -i <inputfile> -o <outputfile>')

sys.exit(2)

elif opt in ('-i', '--ifile'):

inputFile = arg

elif opt in ('-o', '--ofile'):

outputFile = arg

return inputFile, outputFile

if \_\_name\_\_ == "\_\_main\_\_":

inputFile, outputFile = main(sys.argv[1:])

f = open(inputFile)

data = f.readlines()

f.close

n = len(data)

sq\_n = int(math.sqrt(n))

DIGITS = getDigit(n)

X = exprvars('x', (1, n+1), (1, n+1), (1, n+1))

# Value Vonstraints

V = And(\*[

And(\*[

OneHot(\*[ X[r, c, v]

for v in range(1, n+1) ])

for c in range(1, n+1) ])

for r in range(1, n+1) ])

# Row and Column Constraints

R = And(\*[

And(\*[

OneHot(\*[ X[r, c, v]

for c in range(1, n+1) ])

for v in range(1, n+1) ])

for r in range(1, n+1) ])

C = And(\*[

And(\*[

OneHot(\*[ X[r, c, v]

for r in range(1, n+1) ])

for v in range(1, n+1) ])

for c in range(1, n+1) ])

# Box Constraints

B = And(\*[

And(\*[

OneHot(\*[ X[sq\_n\*br+r, sq\_n\*bc+c, v]

for r in range(1, sq\_n+1) for c in range(1, sq\_n+1) ])

for v in range(1, n+1) ])

for br in range(sq\_n) for bc in range(sq\_n) ])

S = And(V, R, C, B)

grid = getGrid(data)

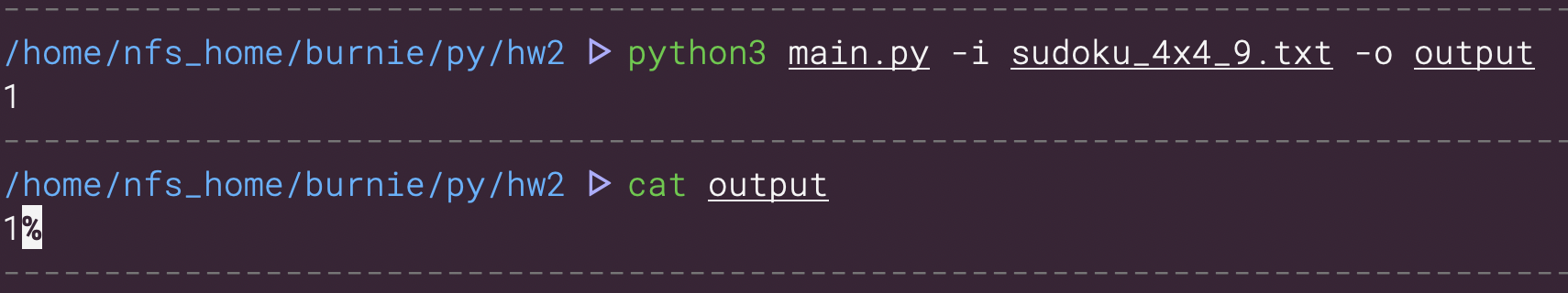
f = open(outputFile, 'w')

display(solve(grid), f)

f.close()

* **Result：**

1. 4x4



1. 9x9

