**Homework 1**

309553048\_陳柏丞

* **Execution：**
  + Ubuntu 18.04.5 LTS
    - g++ -std=c++17 -o solver main.cpp
    - ./solver sudoku\_9x9.txt out ./MiniSat\_v1.14\_linux
* **Abstract：**
  + Functions：
    - GetInput()
    - CellExactlyOneT()
    - ExactlyOneTfromPool()
    - RowColExactlyOneT()
    - BlockExactlyOneT()
  + Main：
    - Check input arguments
    - GetInput()
    - Transfer input vector to 2D vector
    - Get all clauses
      * CellExactlyOneT()
      * RowColExactlyOneT()
      * BlockExactlyOneT()
    - Encode
    - Run MiniSat
    - Decode
    - Print
* **Implementation：**

將input\_file的內容放進vector

std::vector<int> GetInput(std::string input\_file)

{

std::ifstream Input(input\_file);

if(!Input)

{

std::cout << "Can't read file: " << input\_file << "\n";

exit(1);

}

int tmp;

std::vector<int> vin;

while(Input >> tmp)

vin.push\_back(tmp);

Input.close();

return vin;

}

Exactly one digit appears in each cell.

void CellExactlyOneT(int i, int j, std::vector<std::string> &TMP, int n)

{

int base = i\*n\*n + j\*n;

// At least one true

std::string s = "";

for(int d=1; d<=n; d++)

s = s + std::to\_string(base+d) + ' ';

s = s + "0" + "\n";

TMP.push\_back(s);

// At most one true

s = "";

for(int x=1; x<=n; x++)

{

for(int y=x+1; y<=n; y++)

{

s = s + std::to\_string(-1\*(base+x)) + ' ';

s = s + std::to\_string(-1\*(base+y)) + ' ';

s = s + "0" + "\n";

TMP.push\_back(s);

s = "";

}

}

}

從給定的變數中找出所有each digit exactly appears once的可能

void ExactlyOneTfromPool(std::vector<int> &idx\_pool, std::vector<std::string> &TMP, int n)

{

// At least one true

std::string s = "";

for(int d=1; d<=n; d++)

{

for(auto idx:idx\_pool)

s = s + std::to\_string(idx + d) + ' ';

s = s + "0" + "\n";

TMP.push\_back(s);

s = "";

}

// At most one true

s = "";

for(int d=1; d<=n; d++)

{

for(int x=0; x<idx\_pool.size(); x++)

{

for(int y=x+1; y<idx\_pool.size(); y++)

{

s = s + std::to\_string(-1\*(idx\_pool[x] + d)) + ' ';

s = s + std::to\_string(-1\*(idx\_pool[y] + d)) + ' ';

s = s + "0" + "\n";

TMP.push\_back(s);

s = "";

}

}

}

}

Each digit appears exactly once in each row/column.

void RowColExactlyOneT(int i, int j, std::vector<std::string> &TMP, int n, bool type)

{

std::vector<int> idx\_pool;

// row

if(type)

for(int idx=0; idx<n; idx++)

idx\_pool.push\_back(idx\*n\*n + j\*n);

// col

else

for(int idx=0; idx<n; idx++)

idx\_pool.push\_back(i\*n\*n + idx\*n);

ExactlyOneTfromPool(idx\_pool, TMP, n);

}

Each digit appears exactly once in each block

void BlockExactlyOneT(int i, int j, std::vector<std::string> &TMP, int n)

{

int sqrtN = sqrt(n);

// from upper left corner on each block

int init\_x = i, init\_y = j;

std::vector<int> idx\_pool;

for(int x=0; x<sqrtN; x++)

for(int y=0; y<sqrtN; y++)

idx\_pool.push\_back((init\_x+x)\*n\*n + (init\_y+y)\*n);

ExactlyOneTfromPool(idx\_pool, TMP, n);

}

確認參數數量 → 將input\_file的內容放入vector → 重整成2D vector → 執行CellExactlyOneT(), RowColExactlyOneT(), BlockExactlyOneT() → encode → run MiniSat → decode → print

int main(int argc, char \*argv[])

{

if (argc<4) exit(1);

std::string input\_file = argv[1], output\_file = argv[2], MiniSat = argv[3];

std::vector<int> vin = GetInput(input\_file);

int n = (int) sqrt(vin.size());

int sqrtN = sqrt(n);

// 2D array

std::vector<std::vector<int>> vin2D(n, std::vector<int> (n));

for(int i=0; i<n; i++)

for(int j=0; j<n; j++)

vin2D[i][j] = vin[i\*n+j];

// Clauses array

std::vector<std::string> TMP;

std::string s;

for(int i=0; i<n; i++)

{

for(int j=0; j<n; j++)

{

if(vin2D[i][j] != 0)

{

s = std::to\_string(i\*n\*n + j\*n + vin2D[i][j]) + ' ' + "0" + "\n";

TMP.push\_back(s);

continue;

}

// Cell

CellExactlyOneT(i, j, TMP, n);

}

}

// Row & Col

for(int fix=0; fix<n; fix++)

{

// row (row: [0~8], col: [fix])

RowColExactlyOneT(0, fix, TMP, n, true);

// col (row: [fix], col: [0~8])

RowColExactlyOneT(fix, 0, TMP, n, false);

}

// Block

for(int i=0; i<sqrtN; i++)

for(int j=0; j<sqrtN; j++)

BlockExactlyOneT(i\*sqrtN, j\*sqrtN, TMP, n);

// Encode

std::ofstream Encode("encode");

Encode << 'p' << ' ' << "cnf" << ' ' << n\*n\*n << ' ' << TMP.size() << "\n";

for(int i=0; i<TMP.size(); i++)

Encode << TMP[i];

Encode.close();

// Run MiniSat encode out

std::string str\_run = MiniSat + ' ' + "encode" + ' ' + output\_file;

const char \*run = str\_run.c\_str();

system(run);

// Deldete encode

std::string str\_del = "rm encode";

const char \*del = str\_del.c\_str();

system(del);

// Decode

std::ifstream Decode(output\_file);

std::string ss;

Decode >> ss; // Line 1

if(ss == "UNSAT")

{

std::cout << "NO" << "\n";

return 0;

}

std::vector<std::vector<int>> re(n, std::vector<int>(n));

int i=0, j=0, x;

while(Decode >> x) // Line 2

{

if(x == 0) break;

if(j == n)

{

i++;

j = 0;

}

if(x > 0)

{

x = (x - 1) % n + 1;

re[i][j] = x;

j++;

}

}

Decode.close();

// Print

for(int i=0; i<n; i++)

{

for(int j=0; j<n; j++)

{

std::cout << re[i][j] << ' ';

}

std::cout << "\n";

}

return 0;

}

* **Result：**

1. 9x9：



1. 16x16：

