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 →

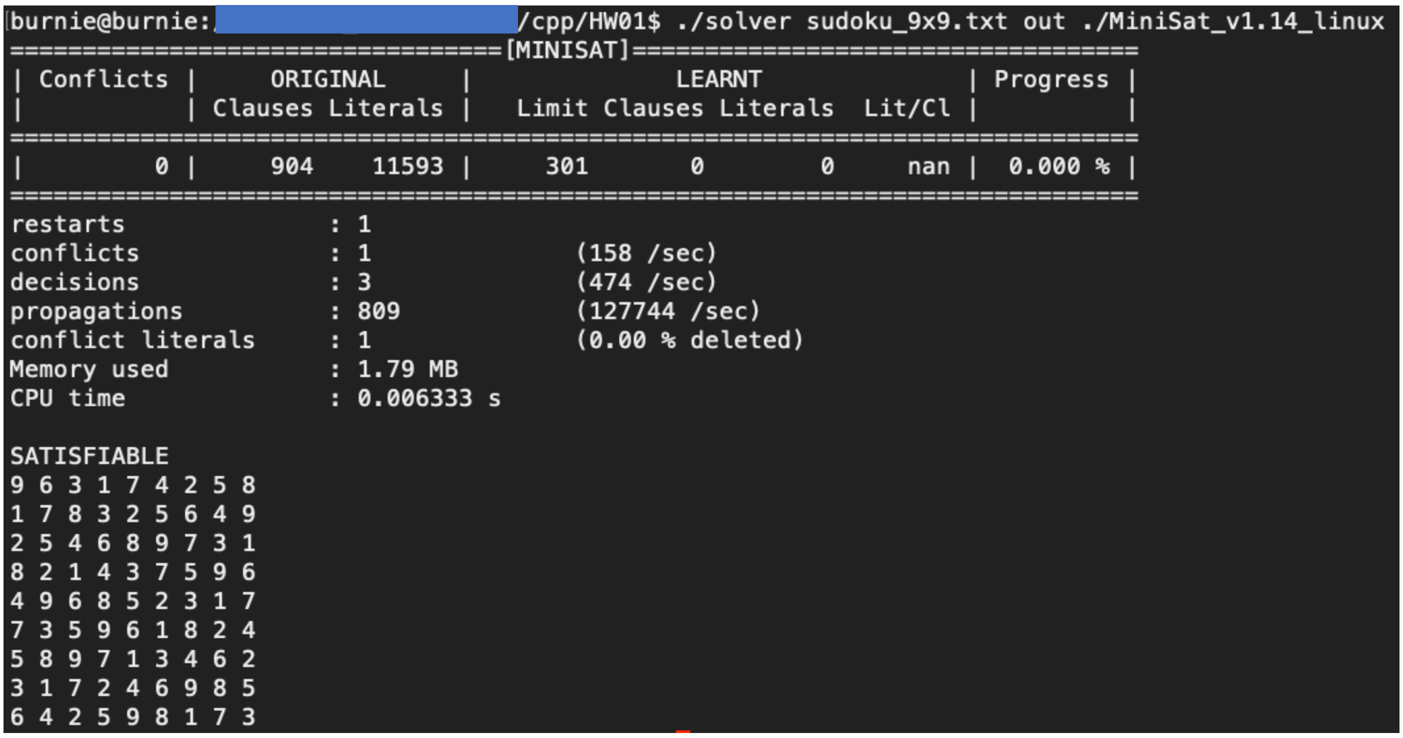
print

```
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;
```

```cpp
    }

    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;
}
```

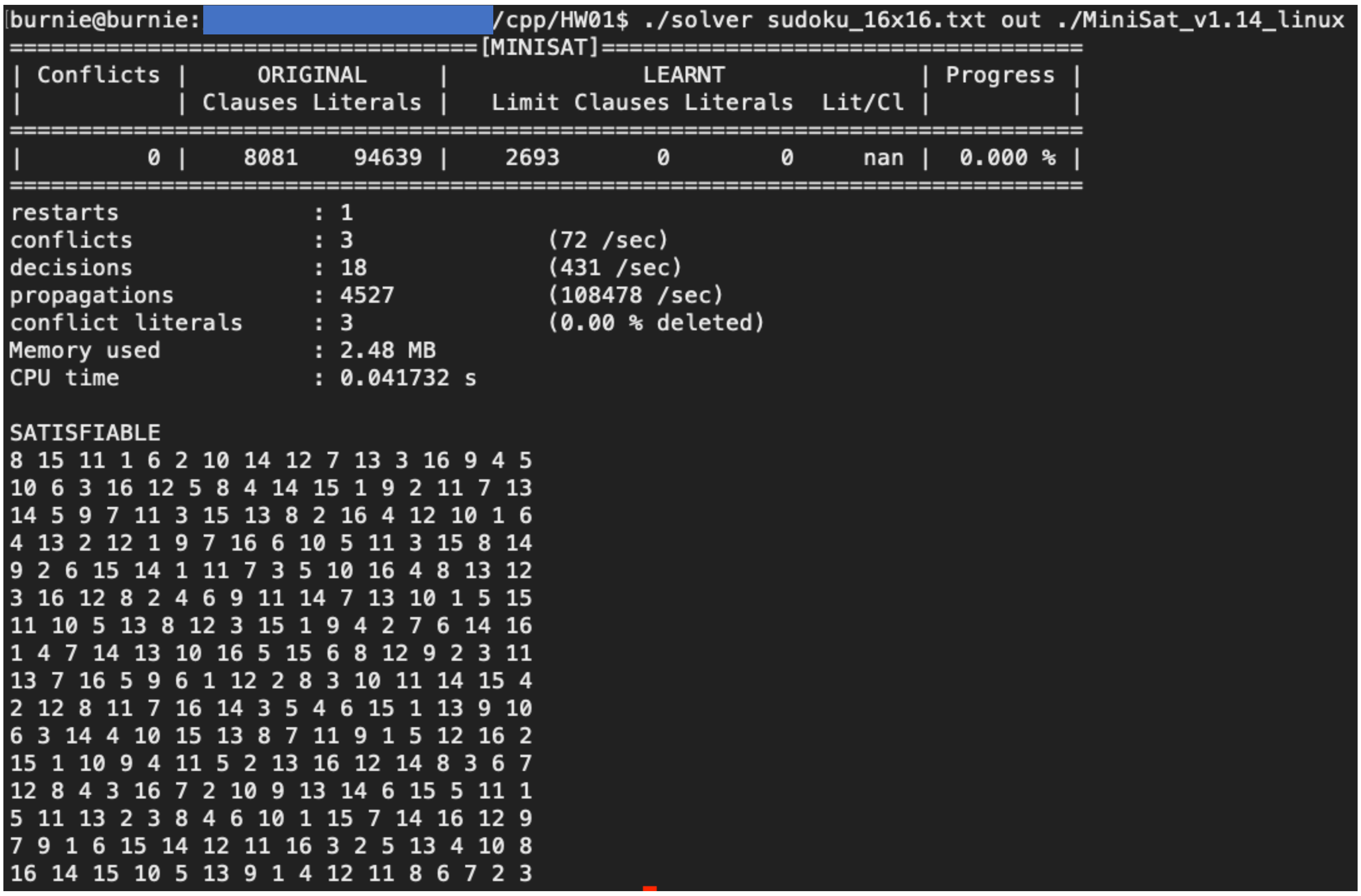
- **Result：**

1. 9x9：



```
burnie@burnie:[redacted]/cpp/HW01$ ./solver sudoku_9x9.txt out ./MiniSat_v1.14_linux
==================================[MINISAT]==================================
| Conflicts |     ORIGINAL     |              LEARNT              | Progress |
|           | Clauses Literals |   Limit Clauses Literals  Lit/Cl |          |
=============================================================================
|         0 |     904    11593 |     301       0        0     nan |  0.000 % |
=============================================================================
restarts              : 1
conflicts             : 1              (158 /sec)
decisions             : 3              (474 /sec)
propagations          : 809            (127744 /sec)
conflict literals     : 1              (0.00 % deleted)
Memory used           : 1.79 MB
CPU time              : 0.006333 s

SATISFIABLE
9 6 3 1 7 4 2 5 8
1 7 8 3 2 5 6 4 9
2 5 4 6 8 9 7 3 1
8 2 1 4 3 7 5 9 6
4 9 6 8 5 2 3 1 7
7 3 5 9 6 1 8 2 4
5 8 9 7 1 3 4 6 2
3 1 7 2 4 6 9 8 5
6 4 2 5 9 8 1 7 3
```

2. 16x16：



```
burnie@burnie:[redacted]/cpp/HW01$ ./solver sudoku_16x16.txt out ./MiniSat_v1.14_linux
==================================[MINISAT]==================================
| Conflicts |     ORIGINAL     |              LEARNT              | Progress |
|           | Clauses Literals |   Limit Clauses Literals  Lit/Cl |          |
=============================================================================
|         0 |    8081    94639 |    2693       0        0     nan |  0.000 % |
=============================================================================
restarts              : 1
conflicts             : 3              (72 /sec)
decisions             : 18             (431 /sec)
propagations          : 4527           (108478 /sec)
conflict literals     : 3              (0.00 % deleted)
Memory used           : 2.48 MB
CPU time              : 0.041732 s

SATISFIABLE
8 15 11 1 6 2 10 14 12 7 13 3 16 9 4 5
10 6 3 16 12 5 8 4 14 15 1 9 2 11 7 13
14 5 9 7 11 3 15 13 8 2 16 4 12 10 1 6
4 13 2 12 1 9 7 16 6 10 5 11 3 15 8 14
9 2 6 15 14 1 11 7 3 5 10 16 4 8 13 12
3 16 12 8 2 4 6 9 11 14 7 13 10 1 5 15
11 10 5 13 8 12 3 15 1 9 4 2 7 6 14 16
1 4 7 14 13 10 16 5 15 6 8 12 9 2 3 11
13 7 16 5 9 6 1 12 2 8 3 10 11 14 15 4
2 12 8 11 7 16 14 3 5 4 6 15 1 13 9 10
6 3 14 4 10 15 13 8 7 11 9 1 5 12 16 2
15 1 10 9 4 11 5 2 13 16 12 14 8 3 6 7
12 8 4 3 16 7 2 10 9 13 14 6 15 5 11 1
5 11 13 2 3 8 4 6 10 1 15 7 14 16 12 9
7 9 1 6 15 14 12 11 16 3 2 5 13 4 10 8
16 14 15 10 5 13 9 1 4 12 11 8 6 7 2 3
```