

DLX- Sudoku-Player 用户手册

开发者：2010137 于皓弛 2010387 迟文韬

1. 介绍

本项目所有代码与测试用例见：

[sudoku-player](https://github.com/LitchiHotpot/sudoku-player)(<https://github.com/LitchiHotpot/sudoku-player>)。

1.1 关于本用户手册

欢迎您使用本用户手册。本用户手册是为了 DLX- Sudoku-Player 程序而编写的程序指导手册。无论您是数独爱好者、数独谜题的解题者，还是对 DLX 算法感兴趣的人士，本手册都将帮助您了解程序的功能、特点和使用方法。

1.2 目标受众

本程序适用于以下目标受众：

数独爱好者：对数独谜题感兴趣，想要通过解决数独谜题进行休闲娱乐。

数独谜题解题者：对数独谜题有着深入的研究，希望找出数独谜题的特性。或者是渴望挑战自己的极限数独爱好者。

对 DLX 算法感兴趣的人士：想要了解 DLX 算法原理和在数独求解中的应用。

1.3 版本信息

本手册适用于 DLX- Sudoku-Player 测试版 1.0。

2. 安装和设置

2.1 环境要求

建议您使用 64bit Windows 10/11 环境运行本程序。如果您需要在 Linux/MacOS 环境中运行本程序，请参考 2.2 节有关如何下载源码并编译运行的教程。

2.2 下载和安装

我们提供了 Windows 环境的可执行程序供您使用，您可以在产品打包文件中找到 sudoku.exe。

如果您的系统不支持 exe 执行文件，或者您想要自己编译源码。可以前该链接 [sudoku-player](https://github.com/LitchiHotpot/sudoku-player) 进行下载并编译运行。经测试，源码在 Windows/Linux/MacOS

均可编译运行。

3. 快速入门指南

3.1 系统概述与核心功能

本程序的主要功能有：

• 生成数独终局：程序能够生成不重复的数独终局，确保每个数字在每行、每列和每个九宫格中都只出现一次。玩家可以选择生成的难度和挖空的数量，通过生成多个难度级别的终局，满足不同玩家的需求。同时，玩家还可以生成具有唯一解的数独。

• 保存和读取数独问题：用户可以通过程序将生成的数独终局保存至文件，方便后续的解题或分享。同时，程序也支持读取文件中的数独问题，以便用户可以利用求解功能得到结果。

• 求解数独问题：程序能够接受用户输入的数独问题，利用 DLX 算法，求解出正确的结果。求解过程高效、准确，输出方式友好，用户能够方便地查看解决方案。

3.2 基本操作

本程序通过命令行指令进行操作，指令和参数如下图所示：

参数名字	参数意义	范围限制	用法示例
-c	需要的数独终盘数量	1-1000000	示例：sudoku.exe -c 20 [表示生成20个数独终盘]
-s	需要解的数独棋盘文件路径	绝对或相对路径	示例：sudoku.exe -s game.txt [表示从game.txt读取若干个数独游戏，并给出其解答，生成到sudoku.txt中]
-n	需要的游戏数量	1-10000	示例：sudoku.exe -n 1000 [表示生成1000个数独游戏]
-m	生成游戏的难度	1-3	示例：sudoku.exe -n 1000 -m 1 [表示生成1000个简单数独游戏，只有m和n一起使用才认为参数无误，否则请报错]
-r	生成游戏中挖空的数量范围	20-55	示例：sudoku.exe -n 20 -r 20~55 [表示生成20个挖空数在20到55之间的数独游戏，只有r和n一起使用才认为参数无误，否则请报错]
-u	生成游戏的解唯一		示例：sudoku.exe -n 20 -u [表示生成20个解唯一的数独游戏，只有u和n一起使用才认为参数无误，否则请报错]

其中-r、-u 与-m 指令必须要和-n 指令一同使用。

数独难度设定如下：1 级 20~30 挖空、2 级 30~45 挖空、3 级 45~55 挖空。

同时，您可以通过-h 指令输出指令帮助。

3.3 常见任务示例

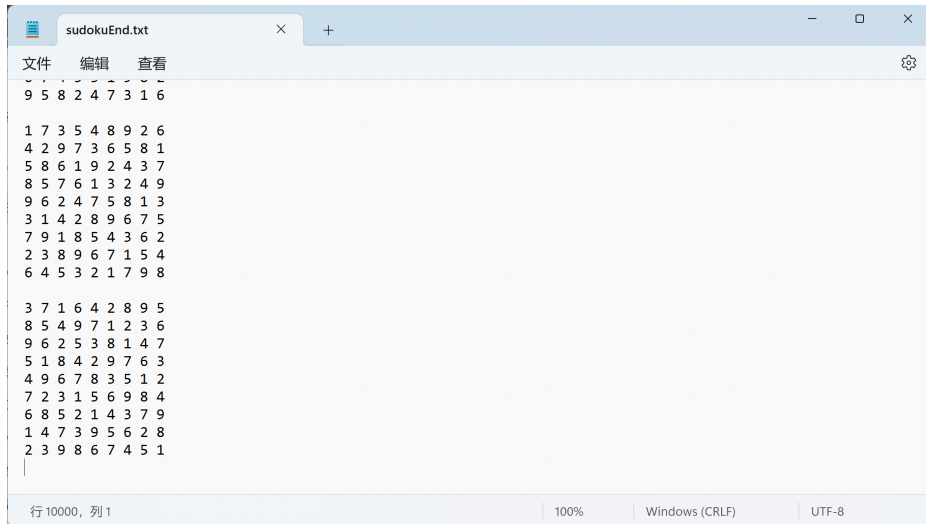
- 输出帮助信息

指令:sudoku.exe -h

```
PS C:\Users\sokoface\Desktop\test> .\sudoku.exe -h
Welcome to use the DLX Sudoku program. The instructions of the program are as follows:
-c Generate Sudoku end to file
-s Read the file and solve Sudoku
-n Generate Sudoku questions with a unique solution
-m Specify the difficulty of the question
-r Formulate the number of questions to be hollowed out
-u Generate a unique solution to the question (negligible)
```

- 生成 10000 个数独终局

指令:sudoku.exe -c 10000

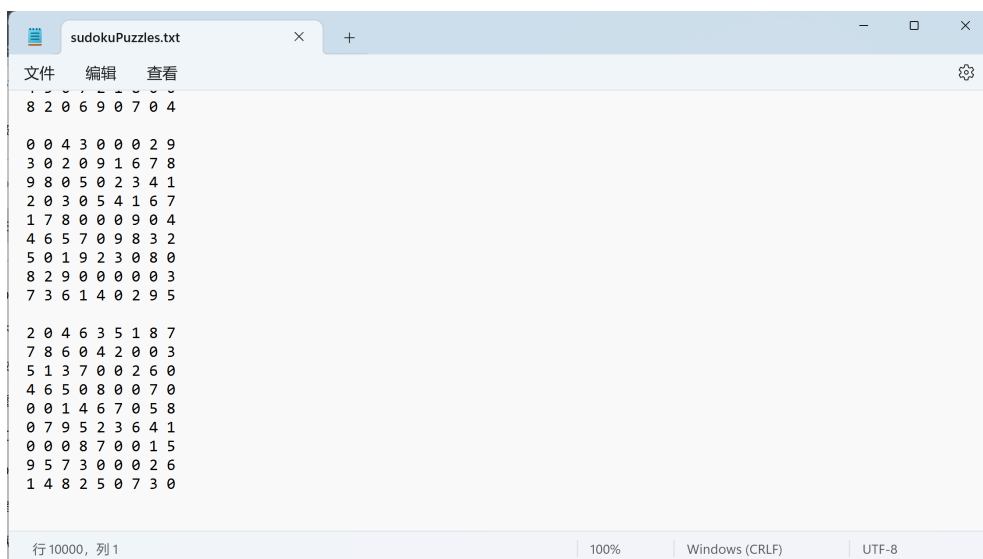


根据行数可以判断，成功生成了 1000 个终局。

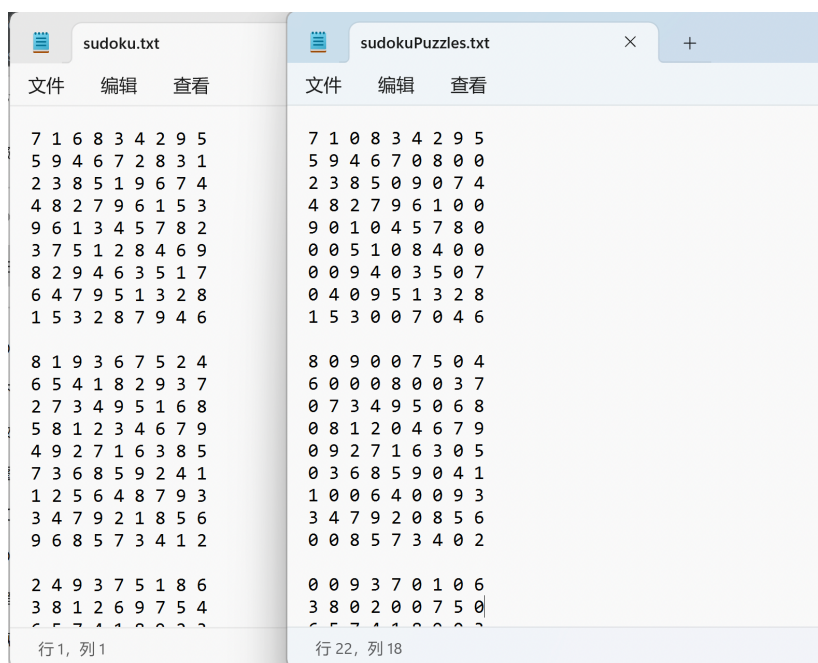
- 求解 sudokuPuzzle.txt 中的数独谜题

指令:sudoku.exe -s sudokuPuzzle.txt

对于 Puzzle 文件：

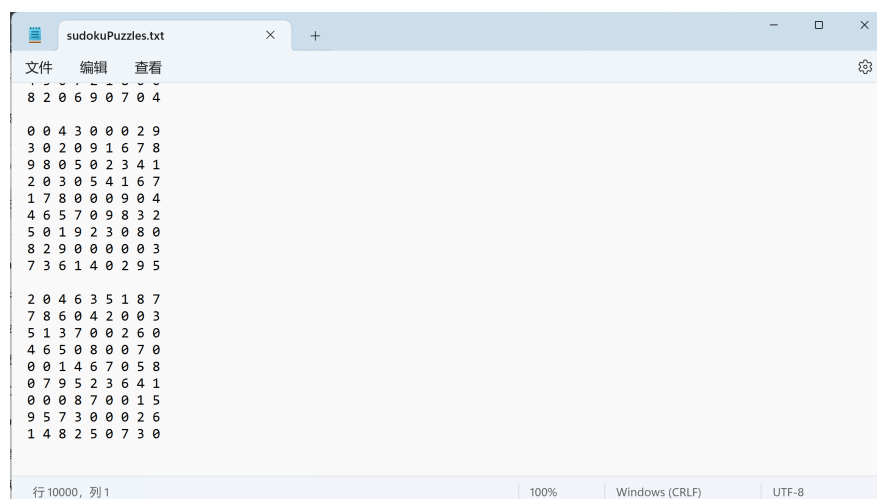


生成解保存在 sudoku.txt 文件中。可以看到，程序成功解出了数独答案。



- 生成 1000 个数独谜题，挖空个数随机(2~55)

指令:sudoku.exe -n 1000



- 生成 1000 个数独谜题，难度级别 2

指令:sudoku.exe -n 1000 -m 2

- 生成 1000 个数独谜题，挖空个数 30~43

指令:sudoku.exe -n 1000 -r 30~43

4. 选项与定制功能

本程序开放源码，如果您想自己定制其他功能，我们为您编写好了接口，您可以自行调用，定制其他符合您需求的功能。

- 求解模块

求解模块，见 SudokuSolver 类。

`solveSudoku` 是通用的求解方法，其输入 DLX 节点结构，待求解数独和答案载体。对数独进行求解并将解放入答案载体，并返回布尔类型值代表是否有解。

`solveWithAllAnswers` 用于解出数独谜题的所有解，并返回一个解的 `vector`。

- 生成模块

生成模块，见 `SudokuGenerater` 类。

其中 `generateSudokus` 方法用于生成数独终局，其输入参数为生成终局的个数。生成的具体步骤简述如下：观察数独特点，可以发现其对角线三个宫可以随意安排 1~9 全排列而不发生冲突。所以将这三个宫设置为生成核。每次随机生成生成核，放入数独结构，之后利用 DLX 算法求出该数独的一个解作为生成的数独终局。这样做保证了生成终局的随机性。

您可以更改生成核的生成方式从而定制您需要的数独结构。具体见 `SudokuGenerater.cpp/generateSudoku`。

`generatePuzzles` 方法用于生成数独谜题，其输入参数为生成谜题的个数与挖空数目。

`isAnswerUnique` 方法用于检测数独是否具有唯一解，用来实现生成唯一解谜题功能。

5. 常见问题

5.1 常见问题

Q1:生成终局最高可支持数目？

A1:理论上来说，最高可支持 $9!^3 = 4.778 \times 10^{16}$ 个终局。

Q2:生成谜题是否具有唯一解？

A2:我们保证生成谜题具有唯一解。对于每次挖空，我们都会进行求解，如果挖空后存在多个解，我们将抛弃这个空，进而从随机序列中选择下一个空。直到只存在唯一解。

5.2 错误信息

本程序会在您输入参数或者程序运行出错是抛出一些错误信息：

No solution for sudoku No.x

对于求解的第 x 数独，不存在解决方案。

Difficulty x not exist, please check!

输入难度有误，请检查。

Spaces x not exist or out of range, please check!

生成谜题时挖空超限。

Instruction x not exist, please check!

指令错误。

There are too many spaces to guarantee the only solution, please retry!

挖空太多，导致无法生成唯一解，请适当减小挖空数量，或者重试。

6. 测试覆盖率

6.1 测试用例设计

对于生成测试，我们针对不同功能设计了不同的测试用例。

对于数独终局生成，我们使其生成要求的最大范围 100000 个。对于数组谜题生成，我们针对不同难度分别生成 10000 个数独谜题，同时用挖空形式再生成 10000 个数独谜题。再生成时保证生成的数独谜题有唯一解。同时我们设计了错误输入的情况，程序应当输出正确的错误信息。

对于解谜测试，我们设计不同的测试用例。

- 空白数独：输入一个没有已知数字的空白数独，程序应当利用 DLX 生成一个固定的数独。

- 有解数独集：分别输入不同数量的数独谜题集，最大数独集数量略微超过要求范围（最大数目设置为 30000）。

- 无解数独集：手动生成无解数独和有解数独混存的数独集，程序应当可以输出对应数独无解信息。

- 多解数独：提供一个有多个解的数独谜题，程序应当返回一个解，但是可以利用程序接口程序找到所有可能的解。

- 特殊格式：只有一行或者一列的数独。

6.2 覆盖率测试

利用 Clion 中的覆盖率测试工具用测试用例进行测试，生成覆盖率报告。

前期覆盖率较低，有一些无用分支和冗余代码，进行删除后生成覆盖率报告如下：

覆盖率: suduku_player		
元素	行覆盖率(%)	分支覆盖率(%)
▼ suduku-player	100% 文件, 71% 行已覆盖	覆盖 70% 的分支
▼ cmake-build-debug		
▼ cmake-build-debug-		
DLXGenerator.cpp	100% 行已覆盖	覆盖 100% 的分支
DLXGenerator.h		
DLXNode.cpp	50% 行已覆盖	覆盖 100% 的分支
DLXNode.h	100% 行已覆盖	覆盖 100% 的分支
DLXSolver.cpp	75% 行已覆盖	覆盖 75% 的分支
DLXSolver.h		
main.cpp	34% 行已覆盖	覆盖 32% 的分支
SudokuGenerator.cpp	96% 行已覆盖	覆盖 93% 的分支
SudokuGenerator.h		
SudokuLoader.cpp	100% 行已覆盖	覆盖 100% 的分支
SudokuLoader.h		
SudokuSolver.cpp	92% 行已覆盖	覆盖 71% 的分支
SudokuSolver.h		

总行覆盖率为 71%。但是 main 函数因为测试时没有去掉其他分支而导致覆盖率较低。观察功能文件，覆盖率基本在 90%以上。证明测试用例测试了程序主要功能，同时也证明程序冗余代码较少。

7. 质量分析

利用 cppcheck 工具进行代码质量分析。

统计 30.06.2023

错误：0

警告：0

风格警告：30

移植可能性警告：0

性能警告：1

信息：20

没有错误与警告。接下来对代码风格警告进行处理。

8. 风格检查

利用 cpplint 进行风格检查，统一代码风格，使其符合代码规范。

修改完后再进行检查，生成报告：

```

PS D:\code\project\pro> cpplint .\DLXGenerator.cpp
.DLXGenerator.cpp:None: pro/DLXGenerator.cpp should include its header file pro/DLXGenerator.h [build/include] [5]
Done processing .DLXGenerator.cpp
Total errors found: 1
PS D:\code\project\pro> cpplint .\DLXGenerator.h
Done processing .DLXGenerator.h
PS D:\code\project\pro> cpplint .\DLXNode.h
Done processing .DLXNode.h
PS D:\code\project\pro> cpplint .\DLXNode.cppcpplint .\DLXNode.cpp
.DLXNode.cpp:None: pro/DLXNode.cpp should include its header file pro/DLXNode.h [build/include] [5]
Done processing .DLXNode.cpp
Skipping input '.\DLXNode.cppcpplint': Can't open for reading
Total errors found: 1
PS D:\code\project\pro> cpplint .\DLXSolver.cpp
.DLXSolver.cpp:None: pro/DLXSolver.cpp should include its header file pro/DLXSolver.h [build/include] [5]
Done processing .DLXSolver.cpp
Total errors found: 1
PS D:\code\project\pro> cpplint .\DLXSolver.h
.DLXSolver.h:0: No #ifndef header guard found, suggested CPP variable is: PRO_DLXSOLVER_H_ [build/header_guard] [5]
Done processing .DLXSolver.h
Total errors found: 1
PS D:\code\project\pro> cpplint .\main.cpp
Done processing .\main.cpp
PS D:\code\project\pro> cpplint .\SudokuGenerator.h
Done processing .\SudokuGenerator.h
PS D:\code\project\pro> cpplint .\SudokuLoader.h
Done processing .\SudokuLoader.h
PS D:\code\project\pro> cpplint .\SudokuLoader.cpp
.SudokuLoader.cpp:None: pro/SudokuLoader.cpp should include its header file pro/SudokuLoader.h [build/include] [5]
Done processing .\SudokuLoader.cpp
Total errors found: 1
PS D:\code\project\pro> cpplint .\SudokuGenerator.cpp
.SudokuGenerator.cpp:None: pro/SudokuGenerator.cpp should include its header file pro/SudokuGenerator.h [build/include] [5]
Done processing .\SudokuGenerator.cpp
Total errors found: 1
PS D:\code\project\pro> cpplint .\SudokuSolver.h
.SudokuSolver.h:0: No copyright message found. You should have a line: "Copyright [year] <Copyright Owner>" [legal/copyright] [5]
.SudokuSolver.h:0: No #ifndef header guard found, suggested CPP variable is: PRO_SUDOKUSOLVER_H_ [build/header_guard] [5]
Done processing .\SudokuSolver.h
Total errors found: 2
PS D:\code\project\pro> cpplint .\SudokuSolver.cpp
Done processing .\SudokuSolver.h
PS D:\code\project\pro> cpplint .\SudokuSolver.cpp
.SudokuSolver.cpp:None: pro/SudokuSolver.cpp should include its header file pro/SudokuSolver.h [build/include] [5]
Done processing .\SudokuSolver.cpp
Total errors found: 1
PS D:\code\project\pro> |

```