



# SUDOKU PUZZLE GENERATOR



**Người thực hiện: Chu Ngọc Thắng**



# 1. Ngôn ngữ & Thư viện sử dụng

1.1

Ngôn ngữ lập trình: Java

1.2

Thư viện sử dụng

java.util.Collections: Shuffle để xáo trộn danh sách số cần gán vào sudoku

java.nio.file và java.io: Sử dụng để lưu đề vào file



## 2. Ý tưởng & Thuật toán

2.1

Ý tưởng

Thực hiện quy trình ngược: Sinh một bài sudoku đầy đủ và loại bỏ ô để tạo ra đề

2.2

Thuật toán

Sinh bài sudoku đầy đủ:

- Sử dụng thuật toán Backtracking (Quay lui) để tạo ra một sudoku đầy đủ 81 ô số.
- Các số được sắp xếp ngẫu nhiên để tăng tính đa dạng đề

Tạo đề bài:

- Sau khi có một sudoku đầy đủ, thực hiện xóa bỏ một số ô để tạo thành đề
- Chọn các vị trí, gán giá trị của chúng về 0 và thực hiện kiểm tra số nghiệm của đề sau khi xóa



## 3. Các điều kiện của đề

### 3.1

### Tuân thủ quy tắc đề sudoku

Khi sinh sudoku đầy đủ, thuật toán Backtracking liên tục gọi hàm isValidNumber()

Hàm kiểm tra kỹ 3 quy tắc sudoku trước khi điền bất kỳ số nào

### 3.2

### Có 1 nghiệm duy nhất

Thực hiện tạo đề từ một sudoku đầy đủ và luôn thực hiện kiểm tra số nghiệm sau khi xóa ô

```
public boolean isValidNumber (int row, int col, int num){ 1 usage  Chu Ngô  
    // Kiểm tra hàng  
    for (int c = 0; c < SIZE; c++){  
        if (sudoku[row][c] == num) return false;  
    }  
  
    // Kiểm tra cột  
    for (int r = 0; r < SIZE; r++){  
        if (sudoku[r][col] == num) return false;  
    }  
  
    // Kiểm tra khối 3x3 chứa ô đang xét  
    int startRow = (row / 3) * 3;  
    int startCol = (col / 3) * 3;  
  
    for (int i = 0; i < 3; i++){  
        for (int j = 0; j < 3; j++){  
            if (sudoku[startRow + i][startCol + j] == num) return false;  
        }  
    }  
    return true;  
}
```



## 4. Tối ưu hiệu năng

### 4.1 Sử dụng Backtracking

Tự động loại bỏ nhánh sai, nên sinh một sudoku đầy đủ rất nhanh.

Đáp ứng chính xác yêu cầu có nghiệm bằng cách loại bỏ ô không làm thừa bước kiểm tra tồn kém

### 4.2 Ghi File

Dùng BufferedWriter để gom nhiều lệnh ghi thành một lệnh lớn, giúp giảm số lần truy cập I/O file

```
for (int num : numbers) {
    if (sudokuMatrix.isValidNumber(row, col, num)){
        sudokuMatrix.setCell(row, col, num);

        // Gọi đệ quy
        if (generateFullSudokuMatrix(sudokuMatrix)){
            return true;
        }

        // Nếu đệ quy thất bại, quay lui
        sudokuMatrix.setCell(row, col, EMPTY_CELL);
    }
}

// Không số nào hợp lệ, quay lại
return false;
```



## 5. Cách chạy chương trình

### 5.1

### Tải code và biên dịch

Mở Terminal và chạy lệnh để tải code về:

```
git clone https://github.com/T-Smilling/Sudoku-Puzzle-Generator.git
Sudoku-Puzzle-Generator'...
ating objects: 82, done.
ng objects: 100% (82/82), done.
```

Ở Terminal, trở đến thư mục src của code và chạy lệnh: *javac -encoding UTF-8 Main.java SudokuGenerator.java SudokuMatrix.java*

```
D:\BackEnd> cd Sudoku-Puzzle-Generator
D:\BackEnd\Sudoku-Puzzle-Generator> cd src
```

### 5.2

### Chạy chương trình

Ở Terminal, chạy lệnh: *java Main* để chạy chương trình

```
[1 NGHIỆM]
--- Puzzle 36 saved to file: samples/Sudoku_36.txt

--- Puzzle 36 (43 empty cells) ---
_ _ _ | 2 _ 1 | 6 4 7
6 2 _ | _ _ 9 | _ _ 5
8 _ _ | 5 _ _ | _ 9 _
-----
2 8 _ | _ 7 3 | 1 6 4
_ _ _ | _ _ _ | _ _ _
_ 9 1 | 6 _ 8 | 5 _ _
-----
_ _ 8 | _ 1 2 | 9 5 6
_ 5 2 | _ _ 6 | 7 3 1
_ _ 3 | _ _ _ | 4 _ _
-----
```



## 6. Kết quả

6.1

### Đảm bảo đầy đủ yêu cầu

```
Generating 40 Sudoku puzzles
[1 NGHIỆM]
--- Puzzle 1 saved to file: samples/Sudoku_01.txt

--- Puzzle 1 (46 empty cells) ---
4 _ _ | 9 _ 7 | 5 _ _
_ 3 _ | _ _ 6 | 2 _ _
_ 7 _ | 5 8 2 | _ 3 6
-----
_ 4 1 | _ _ _ | _ 5 _
8 6 3 | _ 4 _ | _ _ _
_ _ _ | 3 9 _ | 8 6 4
-----
3 _ 2 | _ _ _ | _ 7 8
7 1 5 | _ _ 9 | 6 _ _
_ _ _ | 1 _ _ | _ _ _
-----
```

6.2

### Đảm bảo hiệu suất sinh đề

```
--- PERFORMANCE ---
Generated 40 puzzles.
Total time: 181.0 ms
Average time per puzzle: 4,53 ms
---

--- SYSTEM INFO ---
Name       : Windows 11
Arch       : amd64
Version    : 10.0
Cores      : 8 logical cores
JVM Max RAM : 3004 MB
Total Memory : 188
```



**THANKS  
FOR  
WATCHING**

