**HIDATO PUZZLE  
최종보고서**

15조

20142765 조영선

20142772 최승호

20142721 심재욱

20142731 우승민

20162824 박지선

20142773 최인정

**목차**

[**1.** **역할 분담 및 진행상황** 3](#_Toc532436603)

[**1.1** **역할분담** 3](#_Toc532436604)

[**1.2** **프로젝트 진행상황** 3](#_Toc532436605)

[**2.** **프로젝트 구성** 4](#_Toc532436606)

[**3.** **클래스 구조 및 세부설명** 5](#_Toc532436607)

[**3.1. GENERATOR** 5](#_Toc532436608)

[3.1.1. GeneratorManager 5](#_Toc532436609)

[3.1.2. HidatoGenerator 5](#_Toc532436610)

[**3.2. SOLVER** 7](#_Toc532436611)

[3.2.1. SolverManager 7](#_Toc532436612)

[3.2.2. HidatoSolver 7](#_Toc532436613)

[**3** **실행결과** 11](#_Toc532436614)

[**4** **개선사항** 12](#_Toc532436615)

[**4.1. 개선 대상** 12](#_Toc532436616)

[**4.2. 문제점** 12](#_Toc532436617)

[**4.3. 개선 방법** 12](#_Toc532436618)

[**5** **Github 주소 및 참고자료** 12](#_Toc532436619)

# **역할 분담 및 진행상황**

## **역할분담**

- 조영선: 팀장, 코드 모듈화, 설계 및 구현, PM

- 최승호: 외부 코드 분석, solver &generator 구현, 주석 관리

- 우승민: solver & generator 세부 구현, 입출력 구축, 코드 형식 통합

- 심재욱: solver & generator 세부 구현, 알고리즘 개선, test

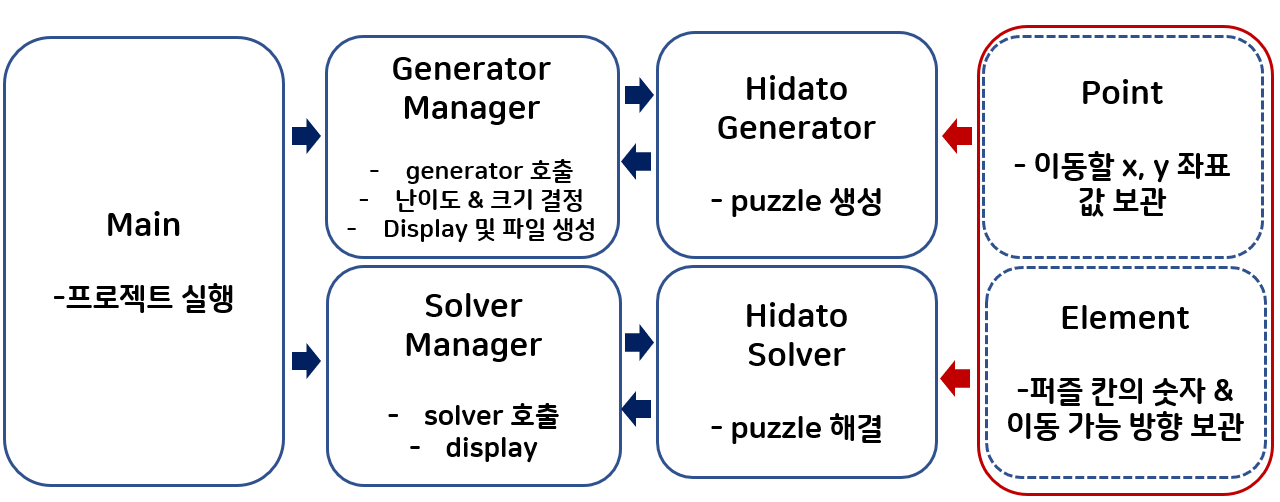
- 최인정: 코드 분석, 프로젝트 계획서 작성, 중간 및 최종 보고서 작성

- 박지선: 회의록 작성, 발표 ppt 제작, 피드백 정리, 참고자료 수집

## **프로젝트 진행상황**

|  |  |
| --- | --- |
| **일정** | **진행사항** |
| 1주차(11.02 ~11.06) | SOLVER TASK 분석 및 구조 설계 |
| 2주차(11.08 ~11.13) | SOLVER 1차 개발 및 디버깅 |
| 3주차(11.14 ~11.20) | SOLVER TEST  SOLVER알고리즘 개선  중간 점검 및 보고서 제작 |
| 4주차(11.21 ~ 11.27) | GENERATOR TASK 분석 및 구조 설계 |
| 5주차(11.28 ~ 12.03) | GENERATOR 1차 개발 완료 |
| 6주차(12.04 ~ 12.13) | GENERATOR TEST  알고리즘 최적화  프로젝트 마무리 및 최종 보고서 작성 |

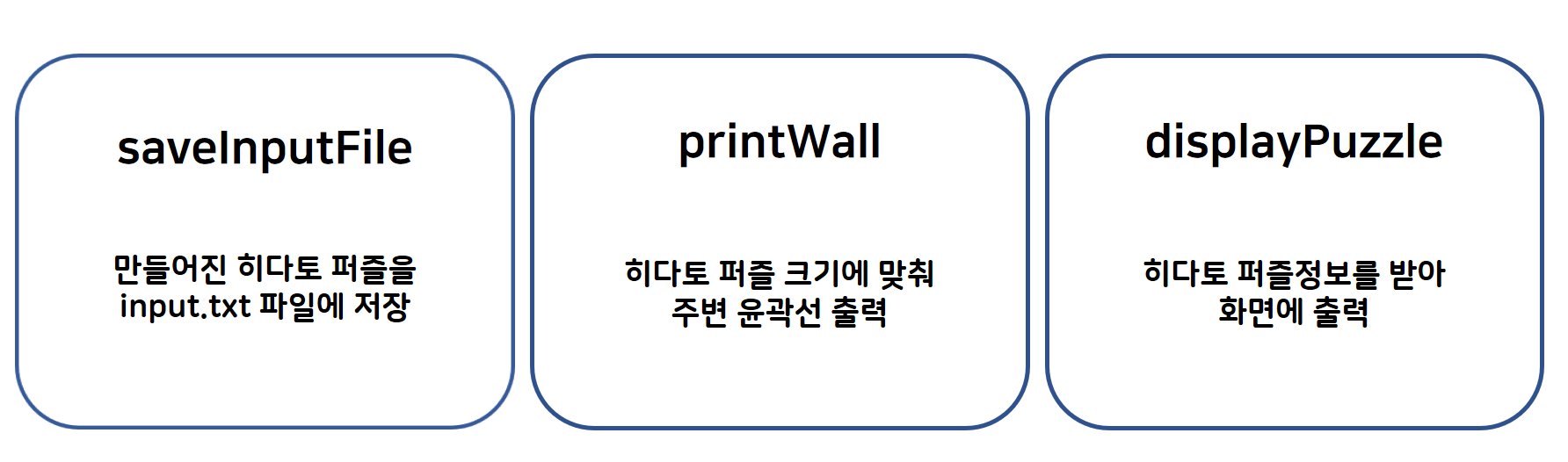
# **프로젝트 구성**



# **클래스 구조 및 세부설명**

## **3.1. GENERATOR**

### 3.1.1. GeneratorManager



### 3.1.2. HidatoGenerator



#### getRandomDiff

* 사용자가 설정한 난이도에 따라 퍼즐 내 공개된 숫자 사이의 간격을 조정하는 메소드
* Easy인 경우: 숫자 간 2~3 칸의 빈칸 생성
* Normal인 경우: 숫자 간 4~7 칸의 빈칸 생성
* Hard인 경우: 숫자 간 8~10 칸의 빈칸 생성

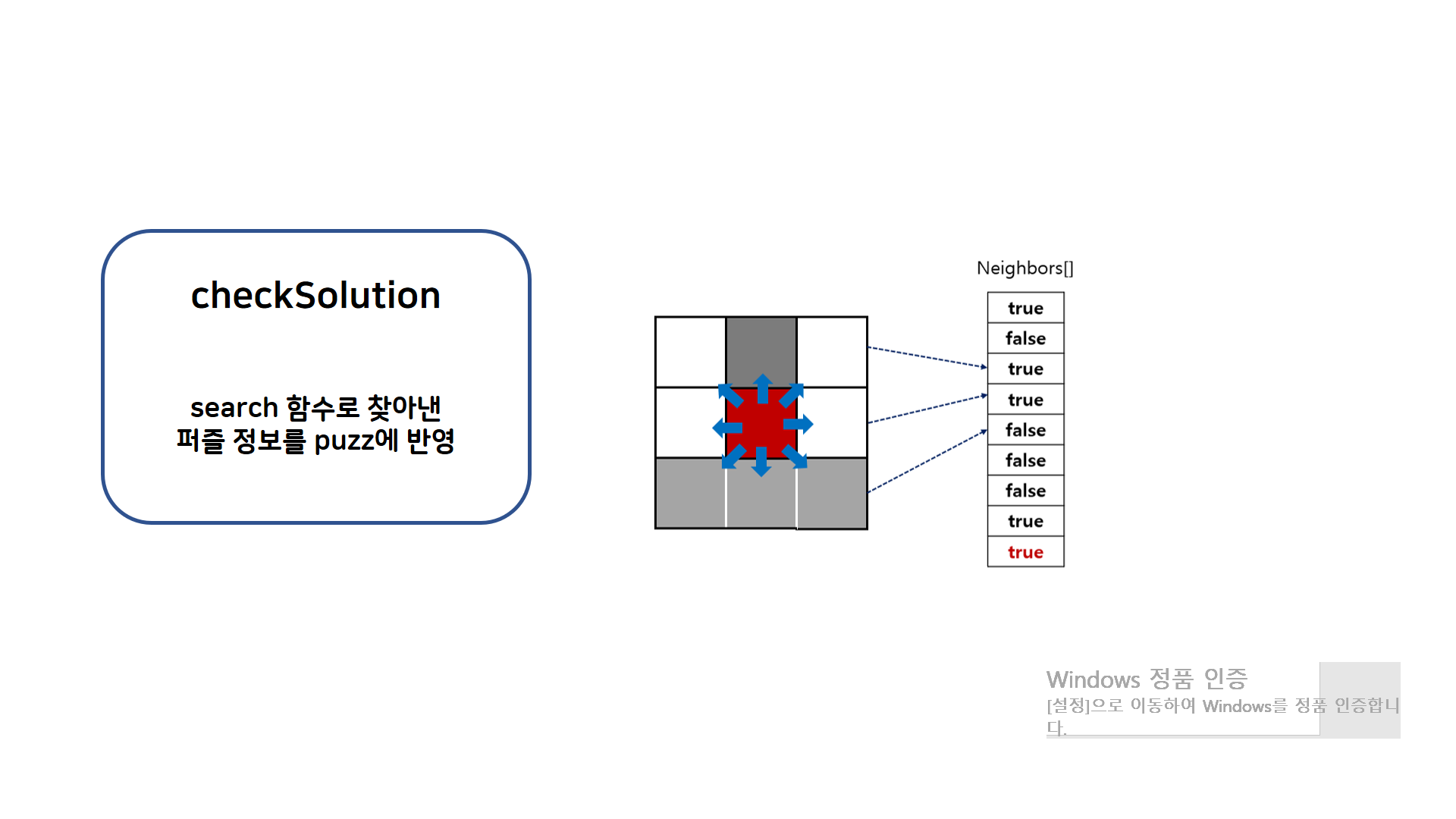
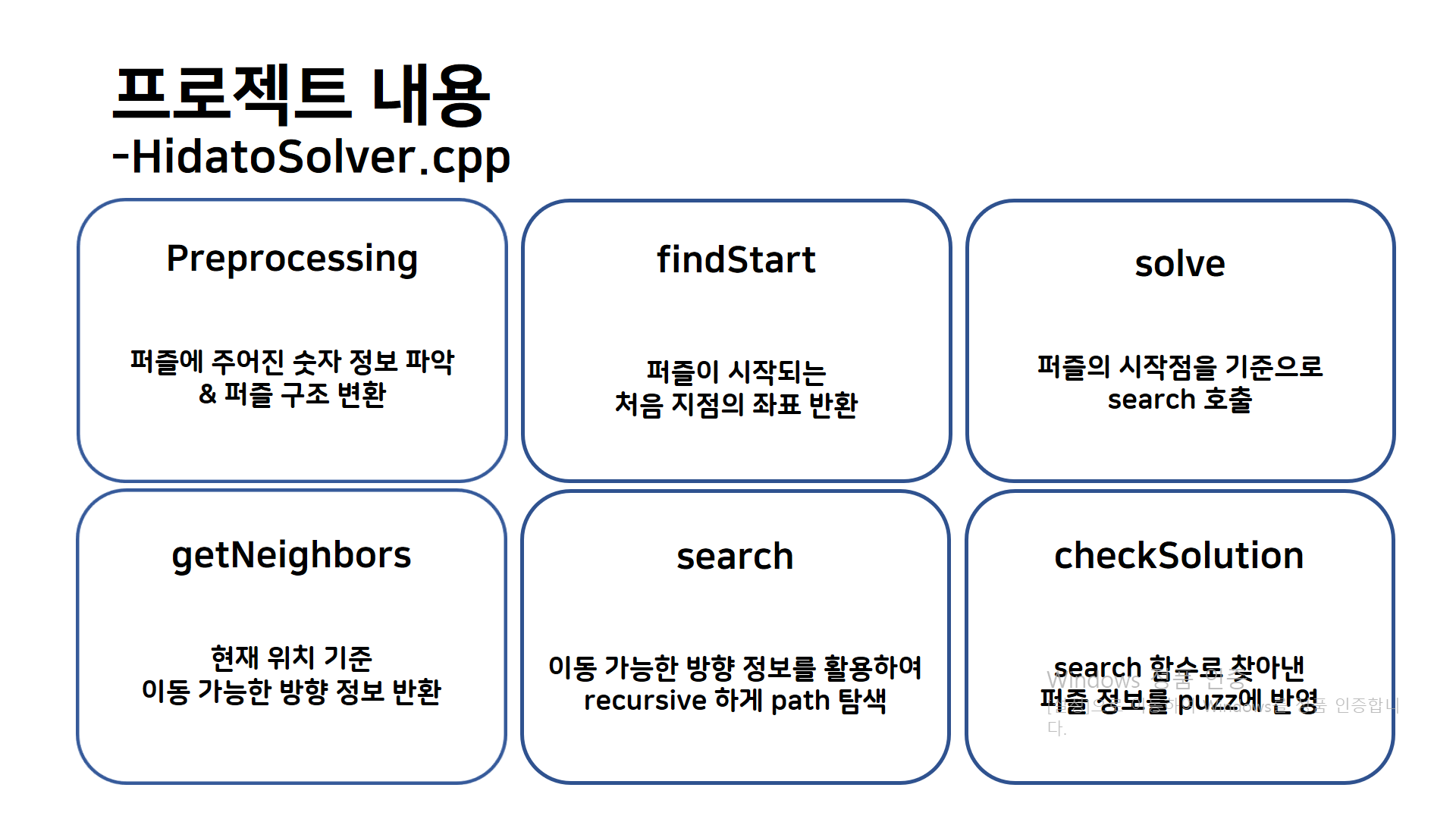
#### makePuzzle

* 지정된 난이도에 따라 숫자 간격을 조정하며 유효한 히다토 퍼즐을 생성하는 메소드.
* 현재 칸과 이웃하는 칸으로 이동했음을 가정 후, 이동 가능 여부를 체크
* 벽에 부딪히거나 이미 숫자가 배정된 칸으로 이동한 경우, count 값 증가
* 숫자가 배정되지 않은 빈칸으로 이동한 경우, 현재 넣을 숫자인 order 값을 해당 칸에 할당 한 후 order 값 증가
* Order 값이 바뀔 때마다 max point 위치를 같이 갱신하여 현재 위치좌표를 보관
* 현재 order가 직전에 표기된 숫자와 diff 만큼 차이 나지 않는 경우, 해당 칸의 숫자를 0으로 표기
* 만약 현재 order가 퍼즐의 마지막 숫자인 경우 이전 숫자와의 간격을 고려하지 않고 퍼즐에 숫자 표기
* Count 값이 일정 값 이상으로 넘어가는 경우, 목표로 하는 히다토 퍼즐 생성이 어렵다고 판단하여 퍼즐을 다시 생성

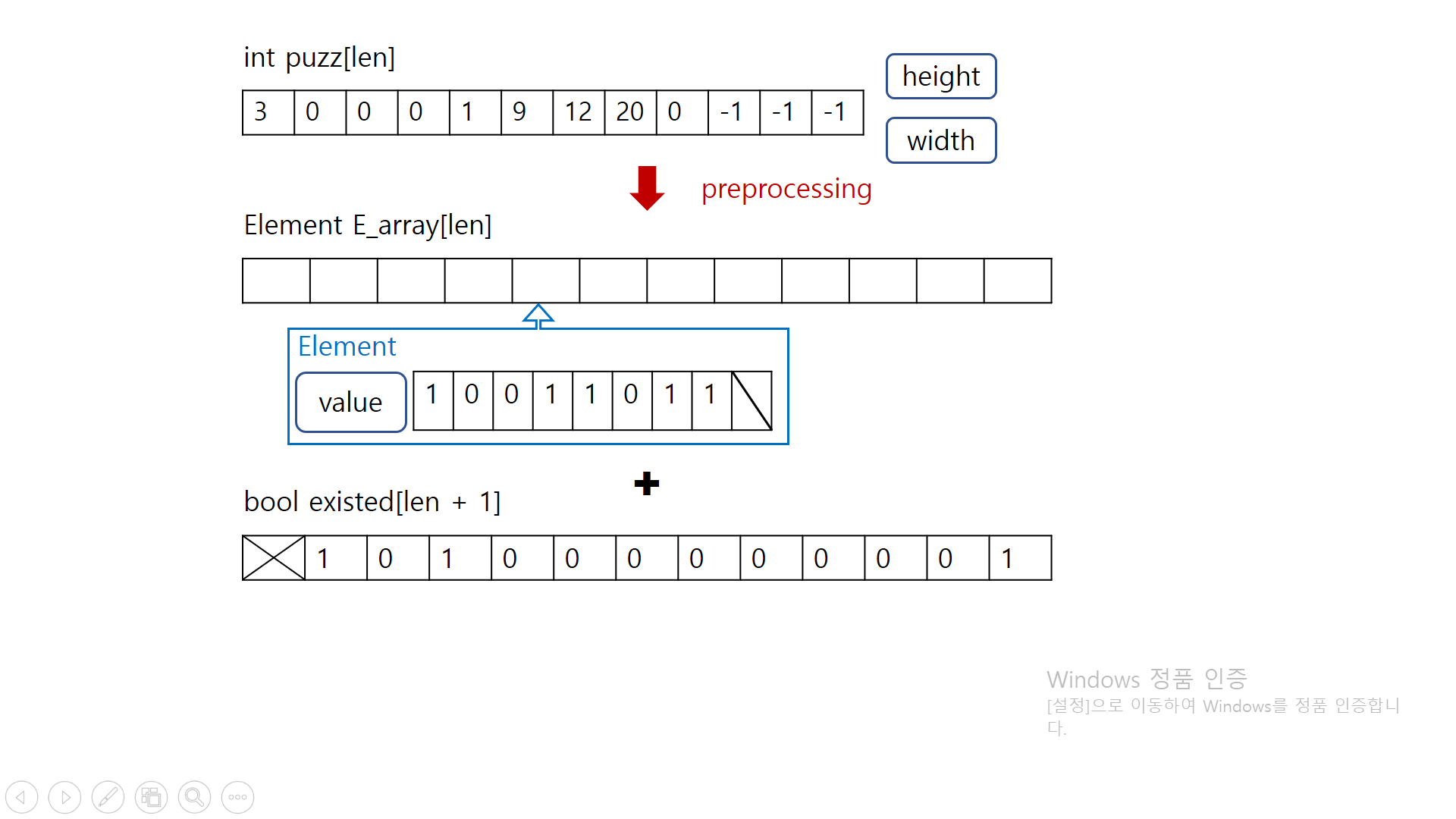
## **3.2. SOLVER**

### 3.2.1. SolverManager

### 3.2.2. HidatoSolver

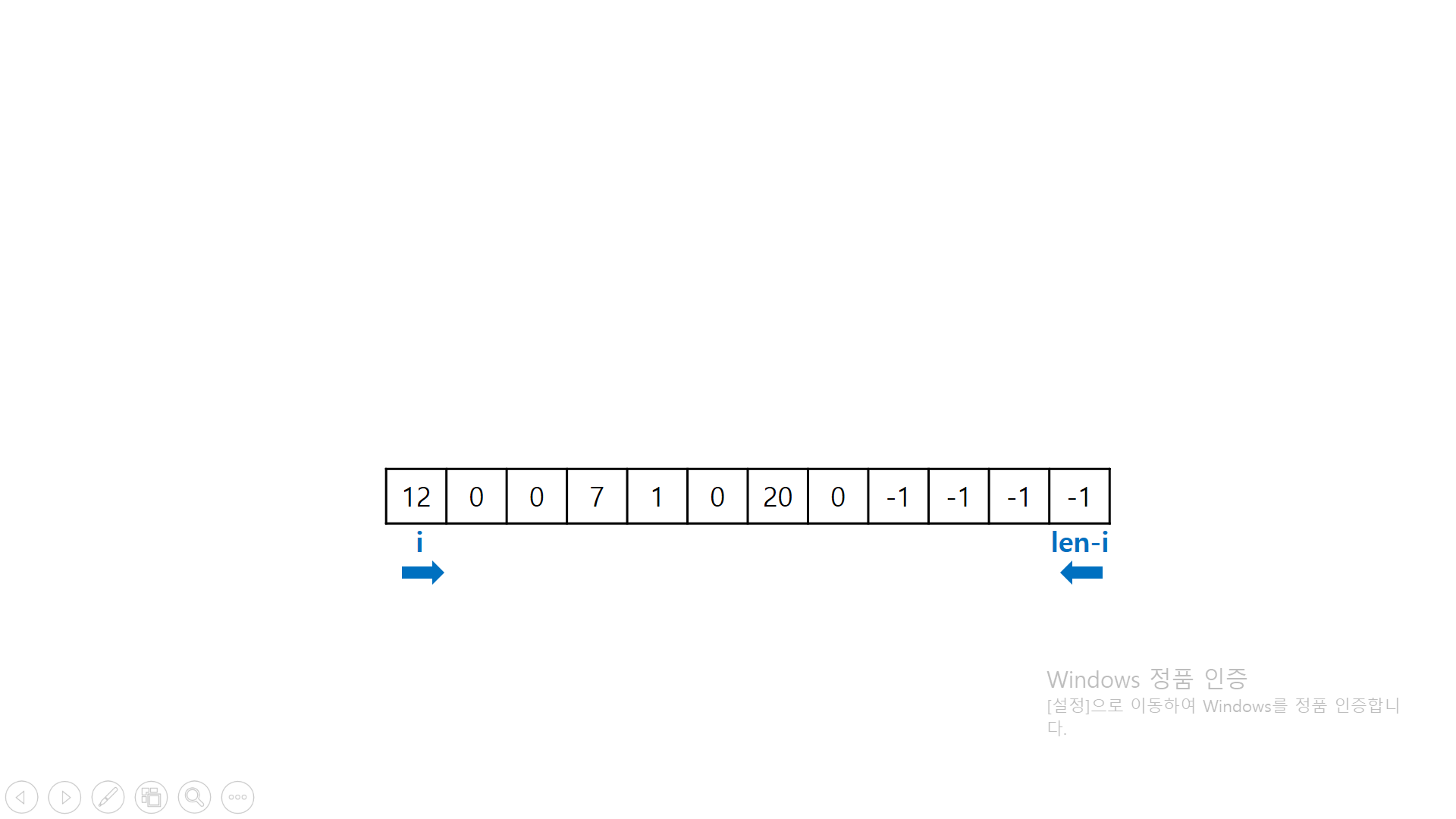


#### Preprocessing



* 퍼즐 해결을 위해 퍼즐의 구조를 변환하고 퍼즐의 마지막 숫자와 퍼즐 내부 숫자 정보를 얻기 위한 메소드
* Int type배열 puzz와 높이, 너비 값을 받아 element type배열인 E\_array와 bool type배열인 existed, 마지막 숫자인 max 값 생성
* Puzz를 loop하며 내부 숫자 값을 E\_array의 value값으로 지정하고, existed의 해당 위치 값을 true로 설정하여 숫자가 존재함을 표시
* Puzz의 loop에서 최대값 정보를 max에 넣어 전체 퍼즐의 마지막 숫자 정보를 파악

#### findStart



* 퍼즐의 처음 시작지점인 1을 퍼즐 내부에서 찾기 위한 메소드
* 칸의 value가 1인 경우, 해당 칸의 x, y 좌표 할당 후 true 반환

1이 아닌 경우, false 반환

* 퍼즐의 처음부터 끝까지 순차적으로 loop하며 퍼즐 전체를 탐색하는 기존의 형태를 퍼즐의 처음과 끝에서 동시 탐색하는 형태로 수정하여 효율성 증대

#### Search

* 현재 위치하는 퍼즐의 좌표와 찾을 숫자 값을 받아 DFS 방식으로 재귀 호출하여 퍼즐의 해답을 만드는 메소드
* getNeighbors()를 호출하여 현재 좌표에서 8개의 방향 중 이동 가능한 방향 정보를 생성한 후 이동 가능한 이웃 칸을 순회하며 진행
* 현재 찾을 숫자가 처음에 주어진 퍼즐에 존재할 때,

- 찾던 숫자를 이웃 칸에서 발견한 경우: 찾을 숫자에 +1 하여 다시 search() recursive call

- 발견하지 못한 경우: false 반환

* 현재 찾을 숫자가 처음에 주어진 퍼즐에 존재하지 않을 때,

-현재 빈 칸인 이웃의 value를 현재 찾는 order로 가정

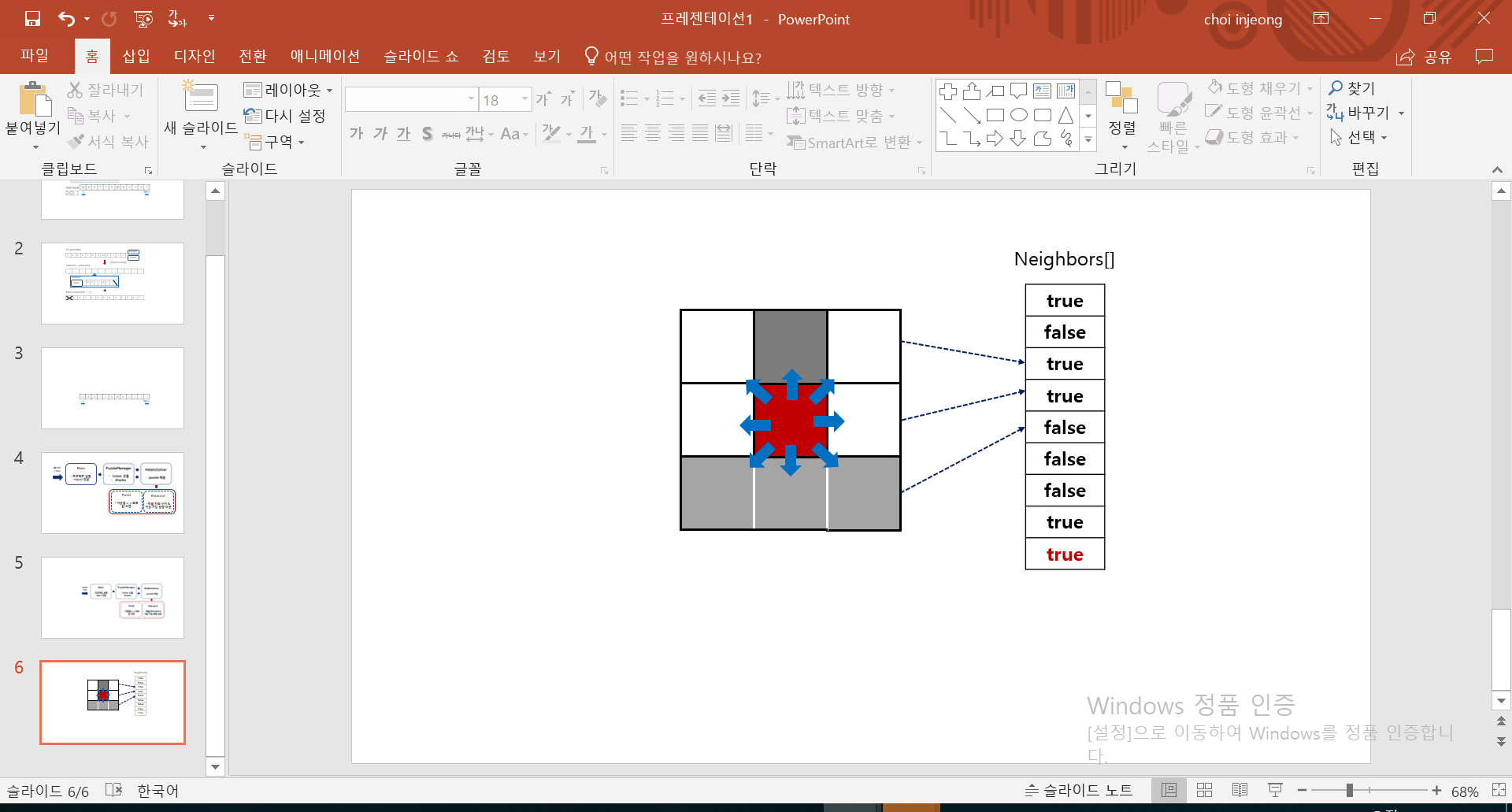
-찾는 숫자를 +1 하여 search() 호출

-호출한 search()가

-true를 반환: 가정이 맞다고 판단하여 계속 진행

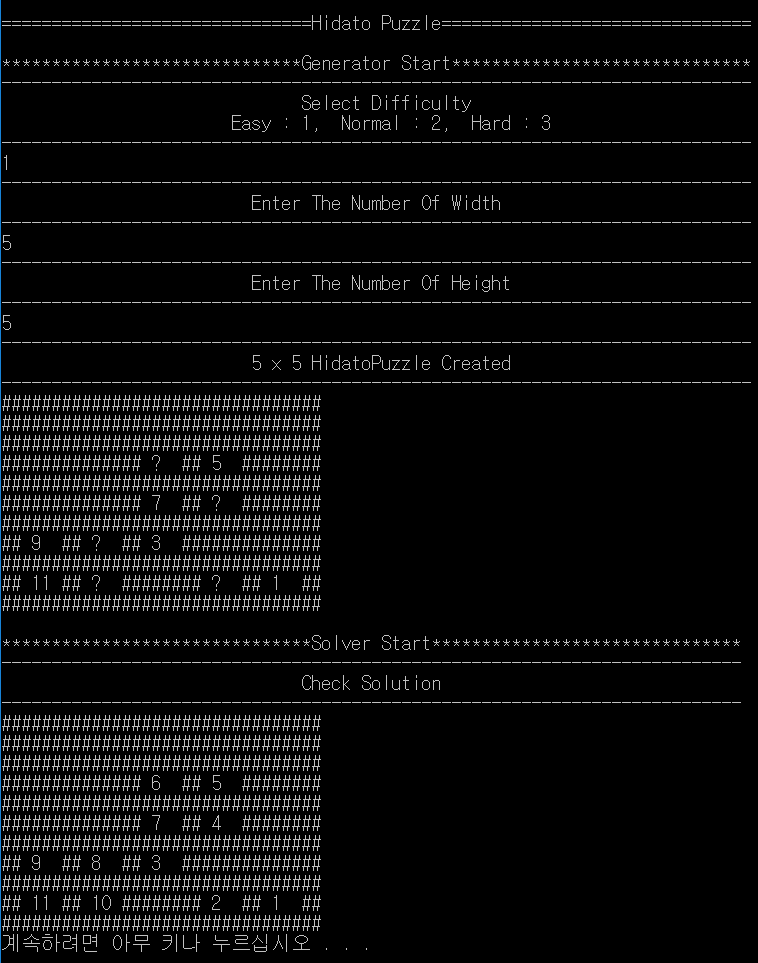
-false를 반환: 가정이 틀리다 판단하여 가정을 recover

#### getNeighbors



* 퍼즐 칸의 좌표를 받아 해당 좌표에서 이동할 수 있는 칸의 방향 정보와 해당 칸이 getNeighbors()를 이미 실행했다는 기록 정보를 만드는 메소드
* 현재 칸과 이웃하는 칸으로 이동했음을 가정 후, 현재 이동한 칸으로의 이동 가능 여부를 체크
* 만약 이동한 칸이 빈칸 or 숫자인 경우, 해당 방향의 이동가능 여부를 의미하는 element의 neighbors 값을 true로 변경
* 현재 퍼즐 좌표에 대한 이웃 중복 확인을 막기위해 현재 칸의 neighbors 배열 마지막 칸에 기록 정보를 남겨, 시간 효율성 증대

# **실행결과**



# **개선사항**

## **4.1. 개선 대상**

- generator의 makepuzzle 메소드

## **4.2. 문제점**

- 퍼즐 생성 단계에서 인접한 8칸 중 이동 가능한 칸이 없는 경우, 목표한 너비와 높이를 만족하지 않는 작은 크기의 퍼즐만이 생성됨

## **4.3. 개선 방법**

- count 변수 추가: 벽에 부딪히거나 이미 숫자가 들어있어 이동이 불가능한 경우에 대해 발생 횟수를 저장하여, 해당 숫자가 허용 범위를 초과한 경우 현재 퍼즐을 포기한 후 새로 퍼즐을 생성

# **Github 주소 및 참고자료**

* Github 주소: <https://github.com/YoungsunCho/Algorithm_HidatoPuzzle>
* 참고자료: <https://rosettacode.org/wiki/Solve_a_Hidato_puzzle>

<https://github.com/fogleman/Hidato>

<https://github.com/YoungsunCho/Algorithm_HidatoPuzzle/blob/master/reference/hidato-master.zip>

<https://github.com/YoungsunCho/Algorithm_HidatoPuzzle/blob/master/reference/solver.cpp>