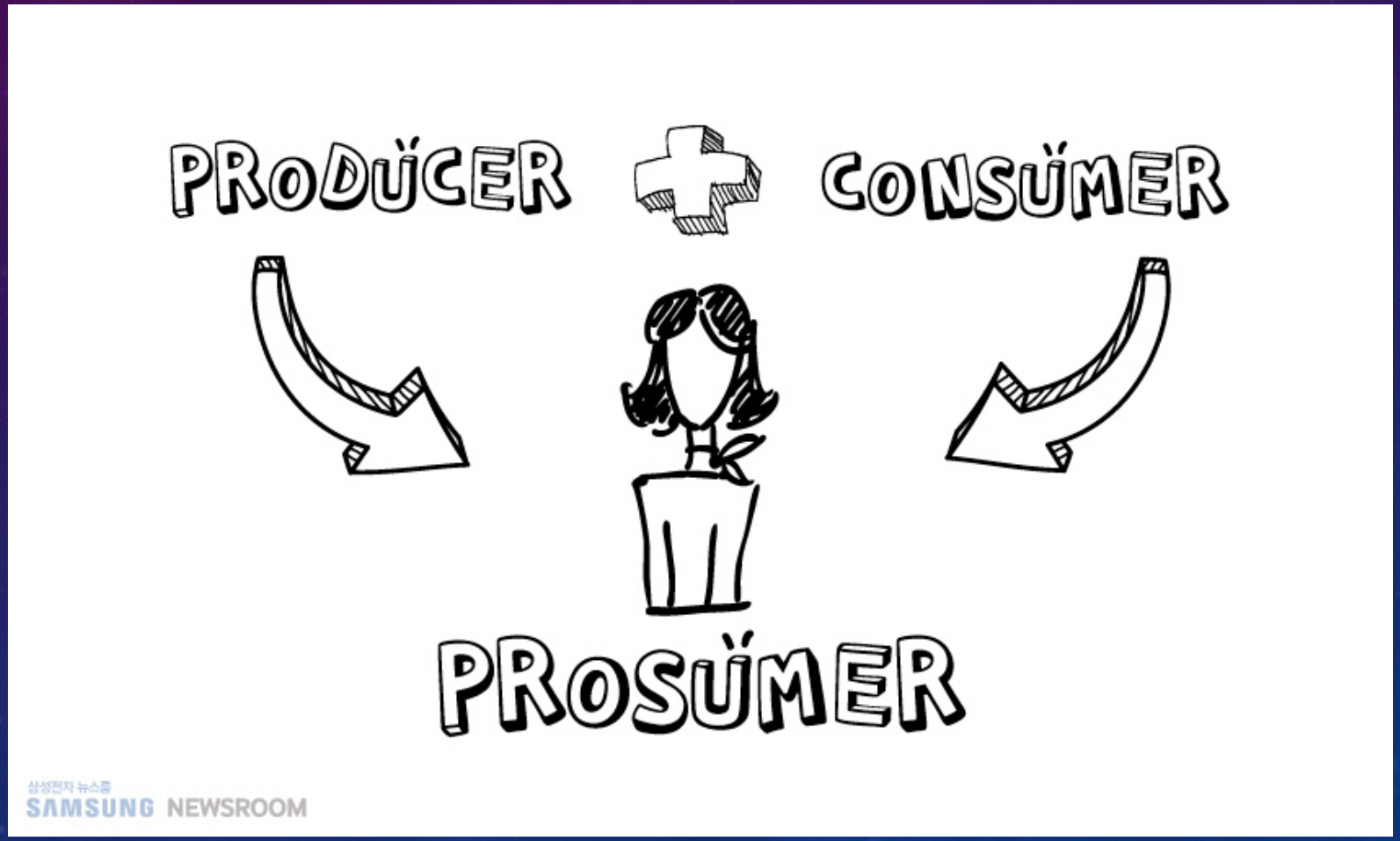
# COMPUTATIONAL THINKING, 3D PRINTER

건국대학교 소프트웨어학과 19학번 강현우

# BEFORE WE START ON


# BEFORE WE START ON



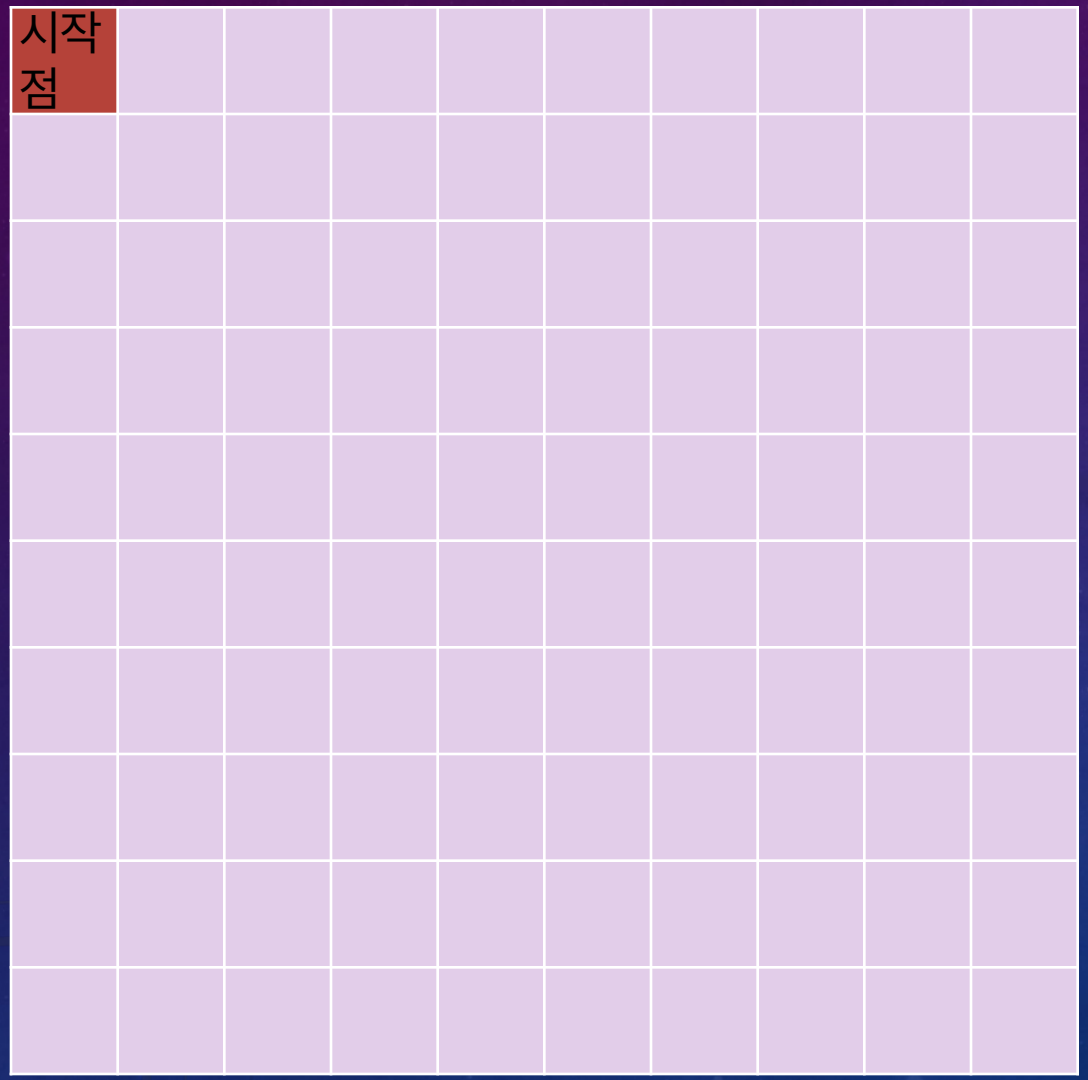
# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER

- 여러분은 로봇 청소기의 개발자입니다.
- 로봇 청소기가 전력 소모를 최대한 줄이기 위해 경로를 찾는 알고리즘을 찾아야되는 상황입니다.
- 여러분은 어떻게 최적화된 경로를 찾아낼 것인가요?

# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER - 조건

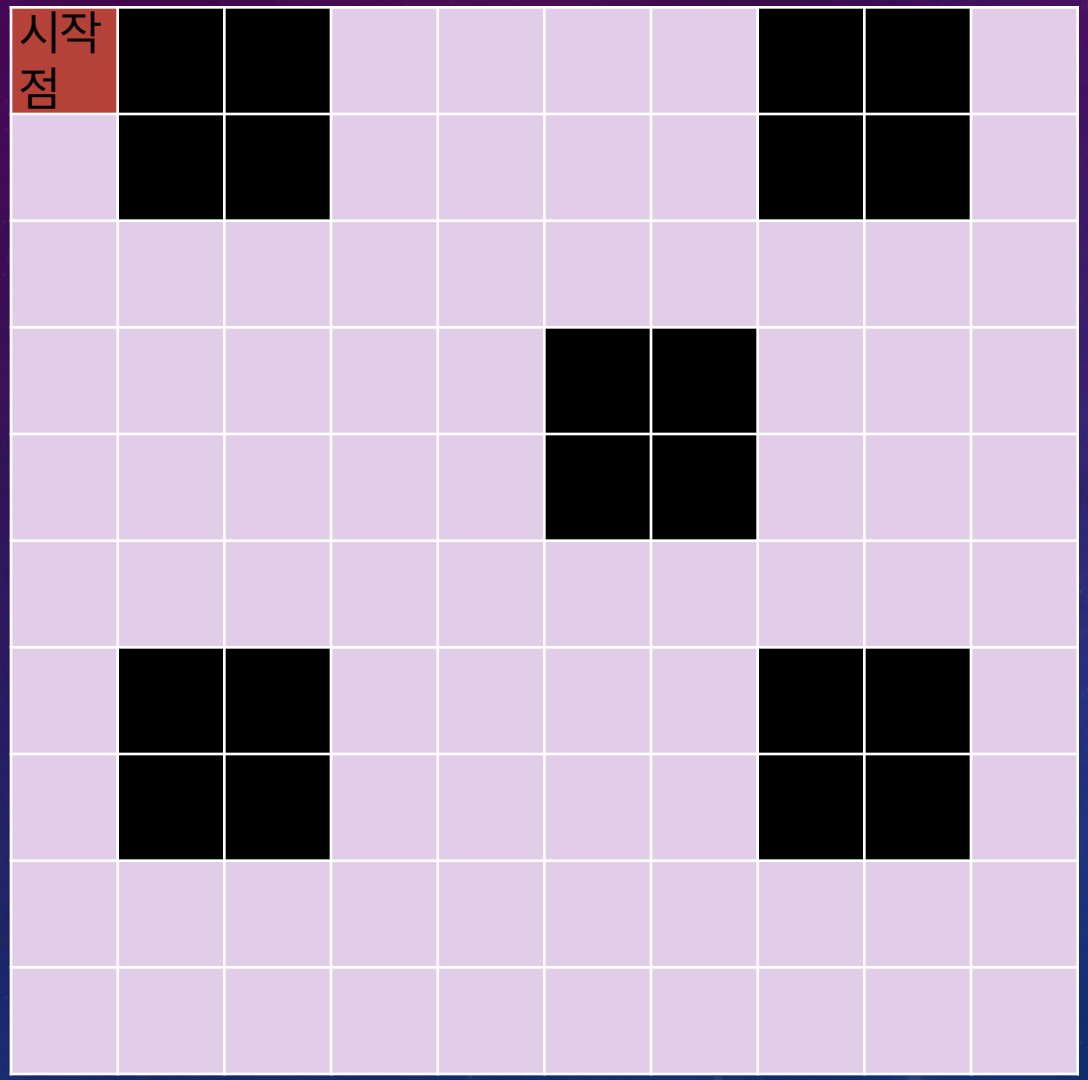
- 여러분은 전력(배터리) 용량을 정해진 이상 사용하지 못합니다.
- 배터리 런타임 = 200분
- 장애물이 없으면 칸의 값은 0, 장애물이 있으면 칸의 값은 1입니다.
- 즉, 장애물이 있는 경우 (x,y) = 1이 성립합니다.
- 임의의 장애물이 있는데, 여러분들의 알고리즘을 적용했을 때, 첫 번째로 (장애물을 제외한) 빈 장소들을 모두 청소했는지를 보고, 그 다음으로 배터리 런타임이 많이 남는 것이 높게 채점됩니다.(채점 비율: 60:40)
- 임의의 장애물의 위치와 갯수는 제가 코딩한 “랜덤 장애물 배치기”에 의해서 결정됩니다.
- 단, 장애물의 위치와 갯수는 모두 청소기가 알고 있다고 가정합니다.
- 즉, 방정식에 어떤 수가 나와도 대입이 가능한 것 처럼 여러분의 아이디어는 임의의 장애물이 있어도 해결 가능해야 됩니다.

# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER



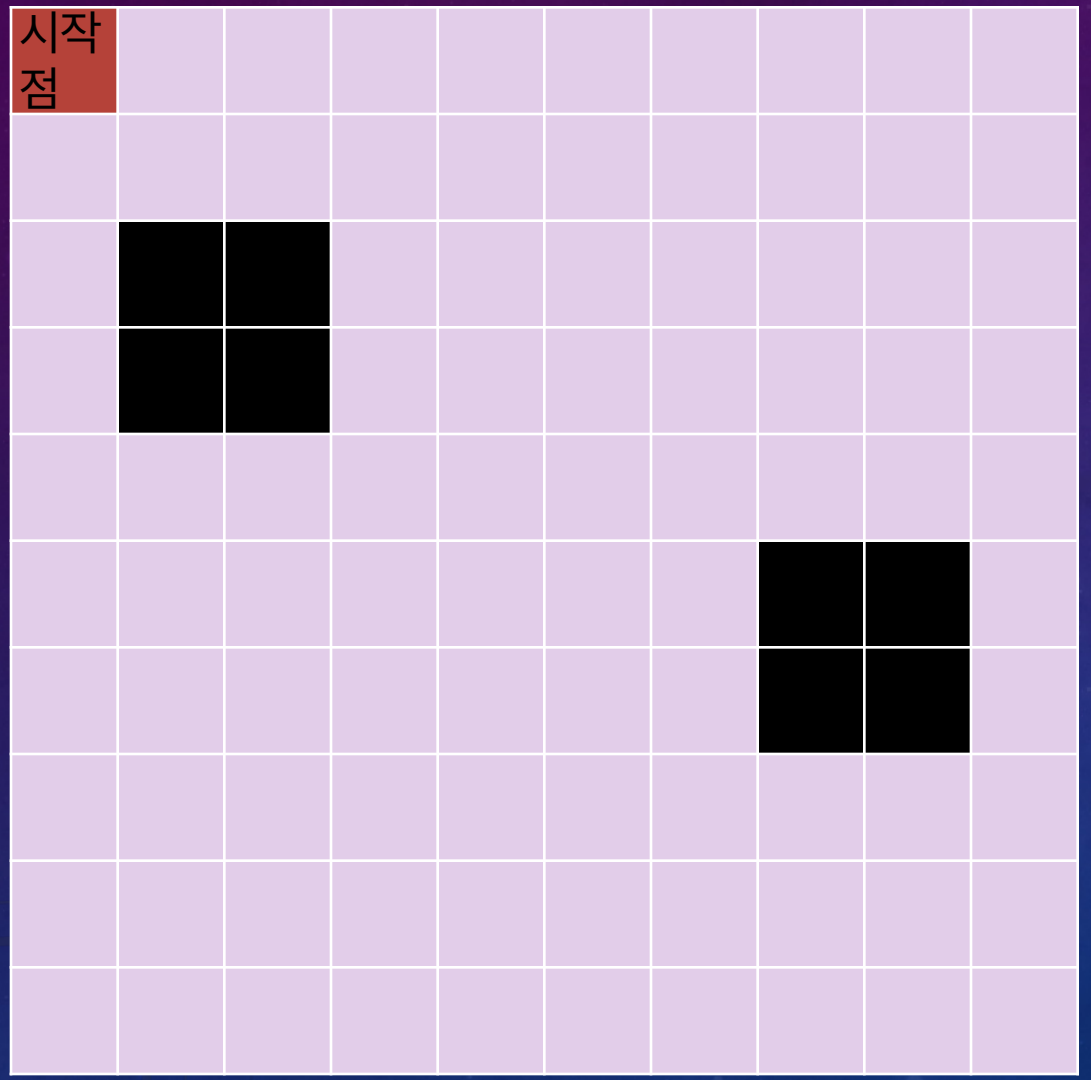

- 한 칸을 청소하는대 걸리는 시간 = 1분
- 총 칸 = 100칸
- 장애물이 있을수도, 없을 수도 있음.
- 또한 한번에 대각선으로 못움직임

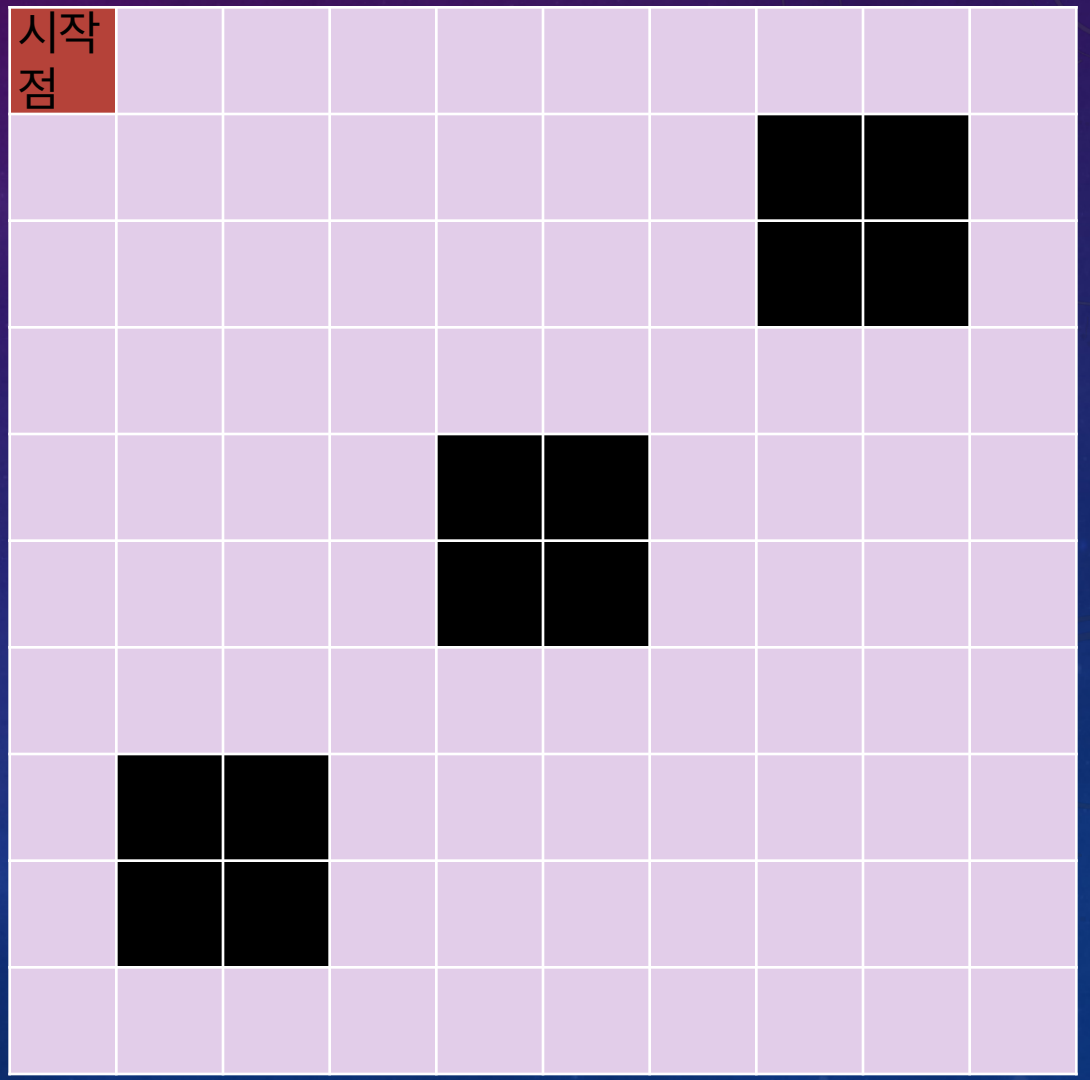
# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER




- 한 칸을 청소하는(이동)대 걸리는 시간 = 1분
- 총 칸 = 100칸
- 장애물이 있을수도, 없을 수도 있음.
- 장애물이 있는 칸은 움직일 수 없음

# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER



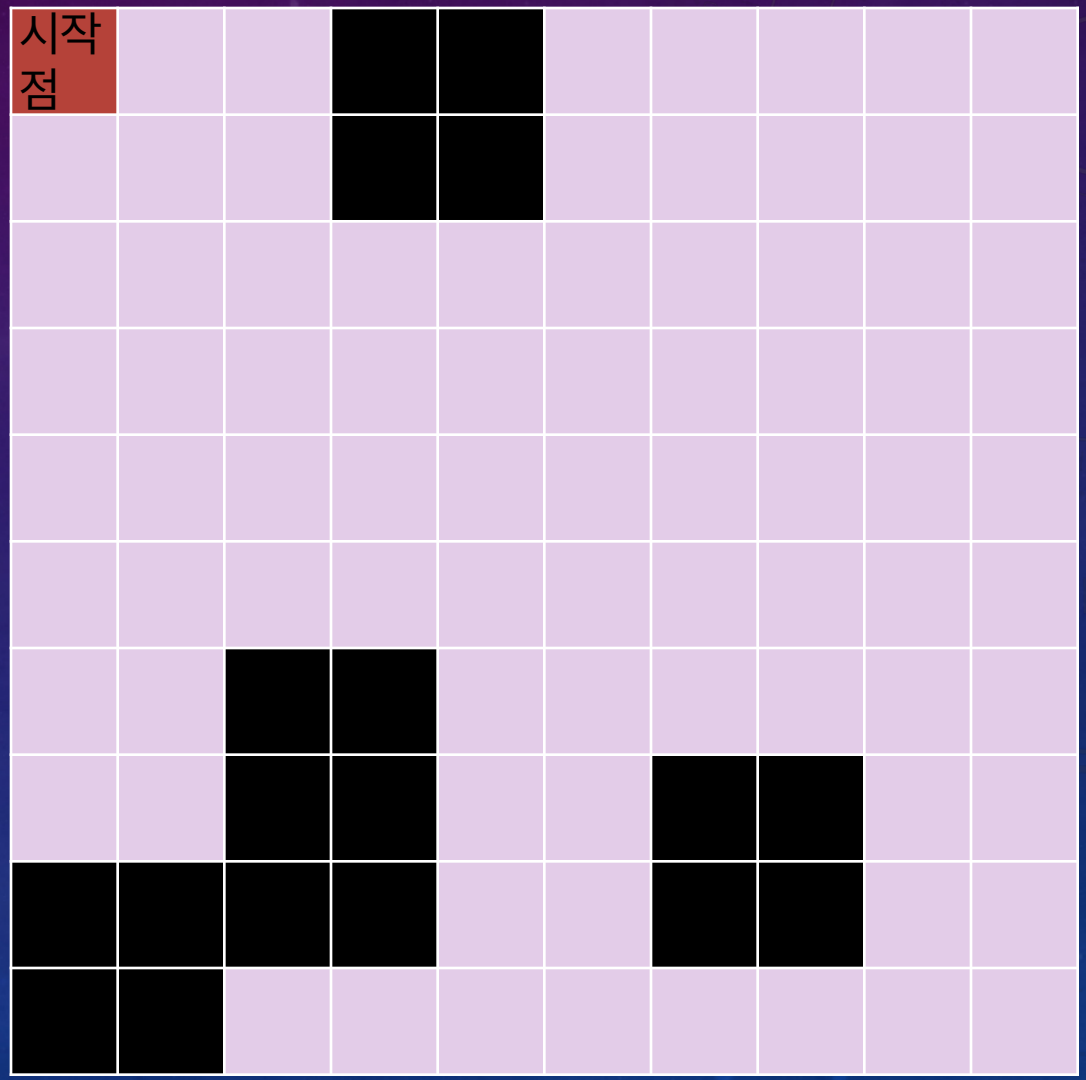



# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER - 기준표

| 장애물 1개 | 장애물 2개 | 장애물 3개 | 장애물 4개 |
| --- | --- | --- | --- |
| Cleared : 60%<br>Battery Runtime: 40% | | | |

# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER - 장애물 예시

```
0 0 0 1 1 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0
0 0 1 1 0 0 1 1 0 0
1 1 1 1 0 0 1 1 0 0
1 1 0 0 0 0 0 0 0 0
```



# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER - 해설 및 7단계 모델

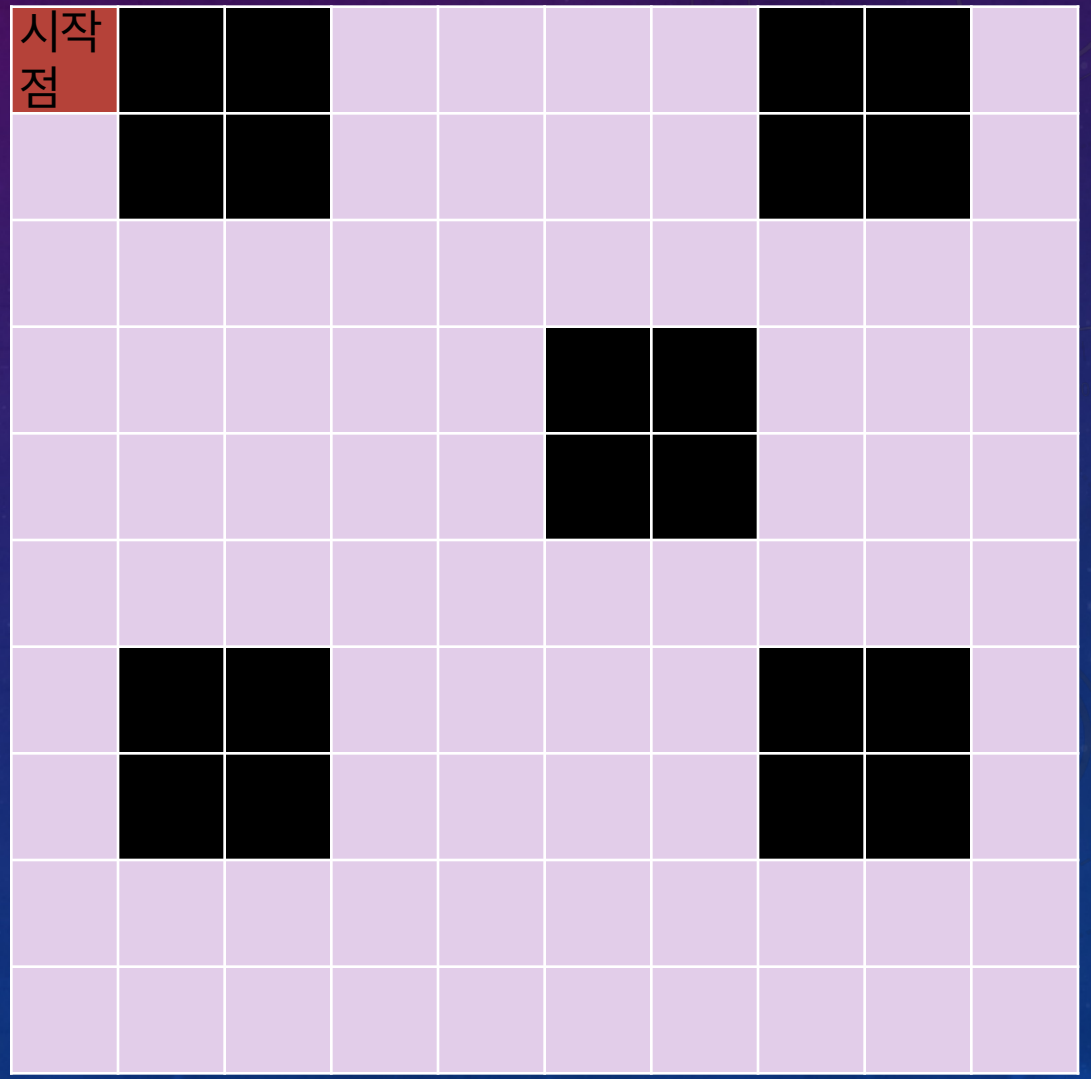
- 문제 분석
- 데이터 수집 및 표현
- 분해
- 패턴인식
- 추상화
- 알고리즘
- 평가

# 문제 분석

- 주어진 문제나 시스템에 대한 논리적 분석을 통해 핵심 사항들을 구체적으로 점검하고 분석
- 결국에는 문제에서 정확히 어떤 것을 원하는지 분석하는 단계.

# 문제 분석

- 문제에서 원하는것?
  —> 최대한의 효율과 최대한의 청소능력을 갖춘 효율적 경로 알고리즘
- 조건 및 제한점 확인 및 인지

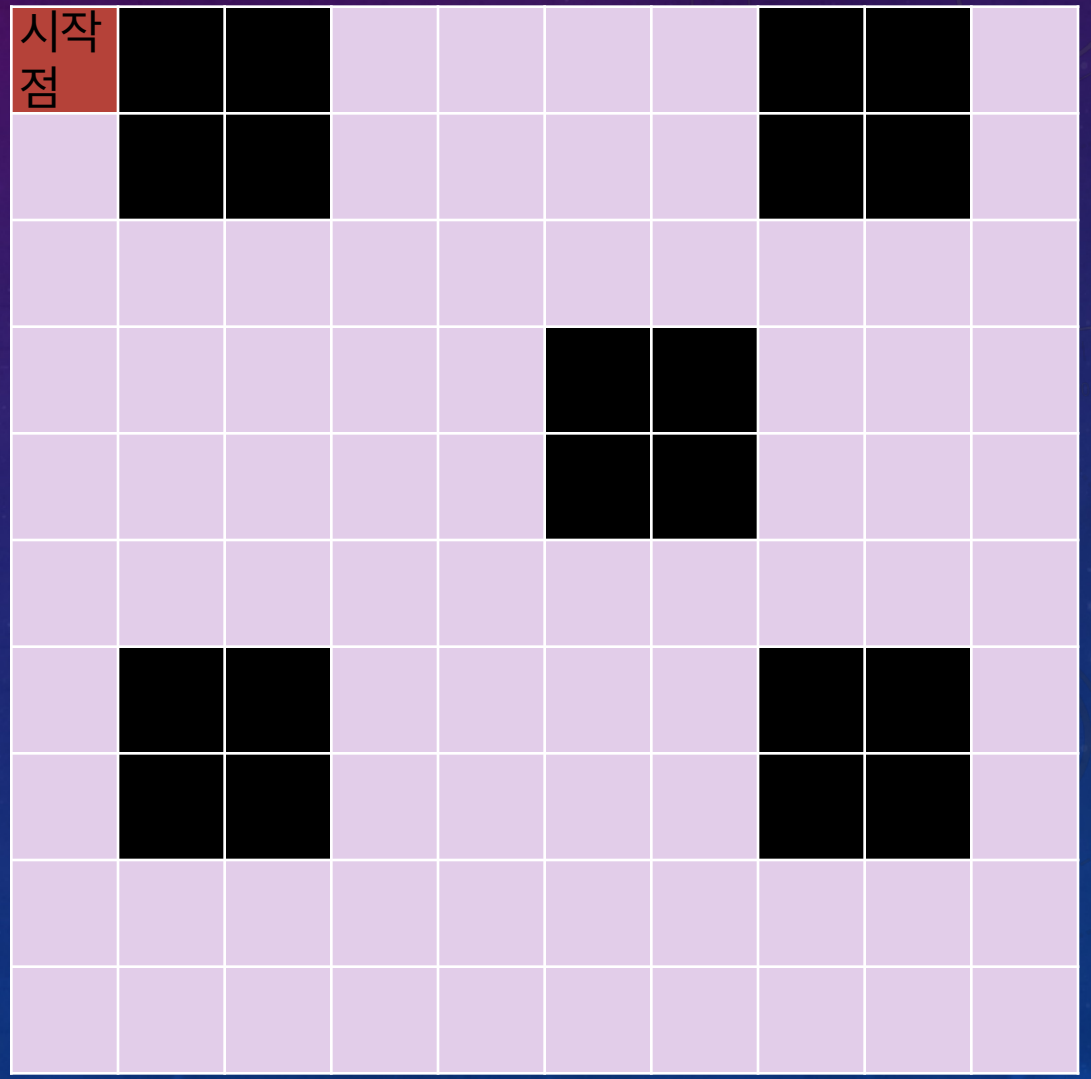


# 데이터 수집 및 표현

- 문제 해결과 관련된 정보들을 컴퓨터를 통하여 수집함
- 이러한 데이터를 적절한 그래프, 차트, 영상 등의 형태로 표현
- 결론은 문제를 푸는데에 있어서 필수적인 정보(조건, 변수 등)을 얻기.

# 데이터 수집 및 표현

- 이 단계의 키포인트: 변수 지정 및 좌표값 이해.
- 시작점:(0,0)
- 종료점:(10,10)
- 장애물:(x, y) = 1
- 한번에 x축으로 1칸, y축으로 한칸 이동 가능
- 장애물로는 이동 불가

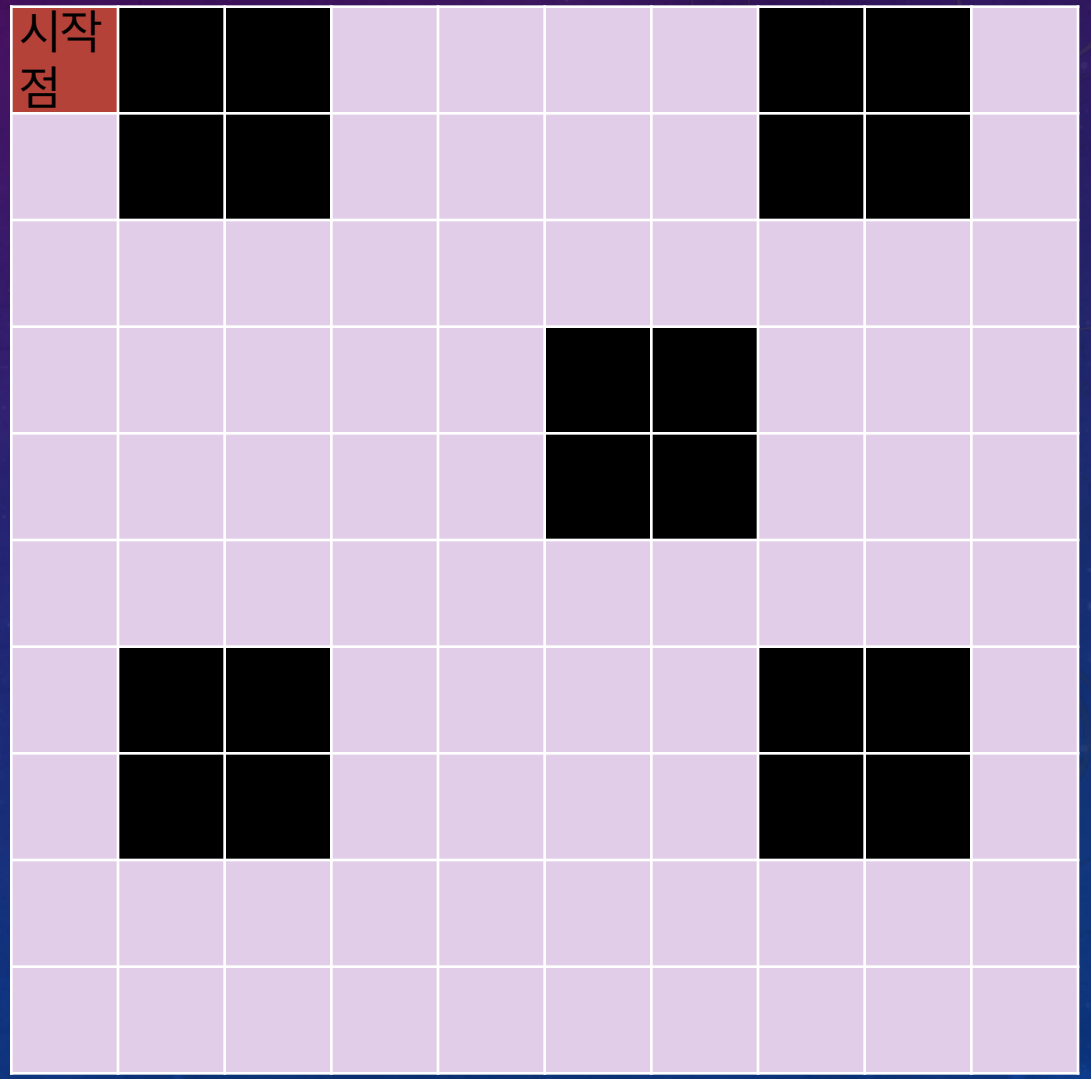


# 분해

- 하나의 큰 복잡한 문제를 여러가지의 문제로 쪼개는 형식의 단계
- 즉, 큰문제 –> 여러가지 작은 문제 –> 작은문제를 하나하나 해결해가면서 큰 문제를 해결해 나아가기.

# 분해

- 효율을 얻는다
- 배터리 런타임을 얻는다

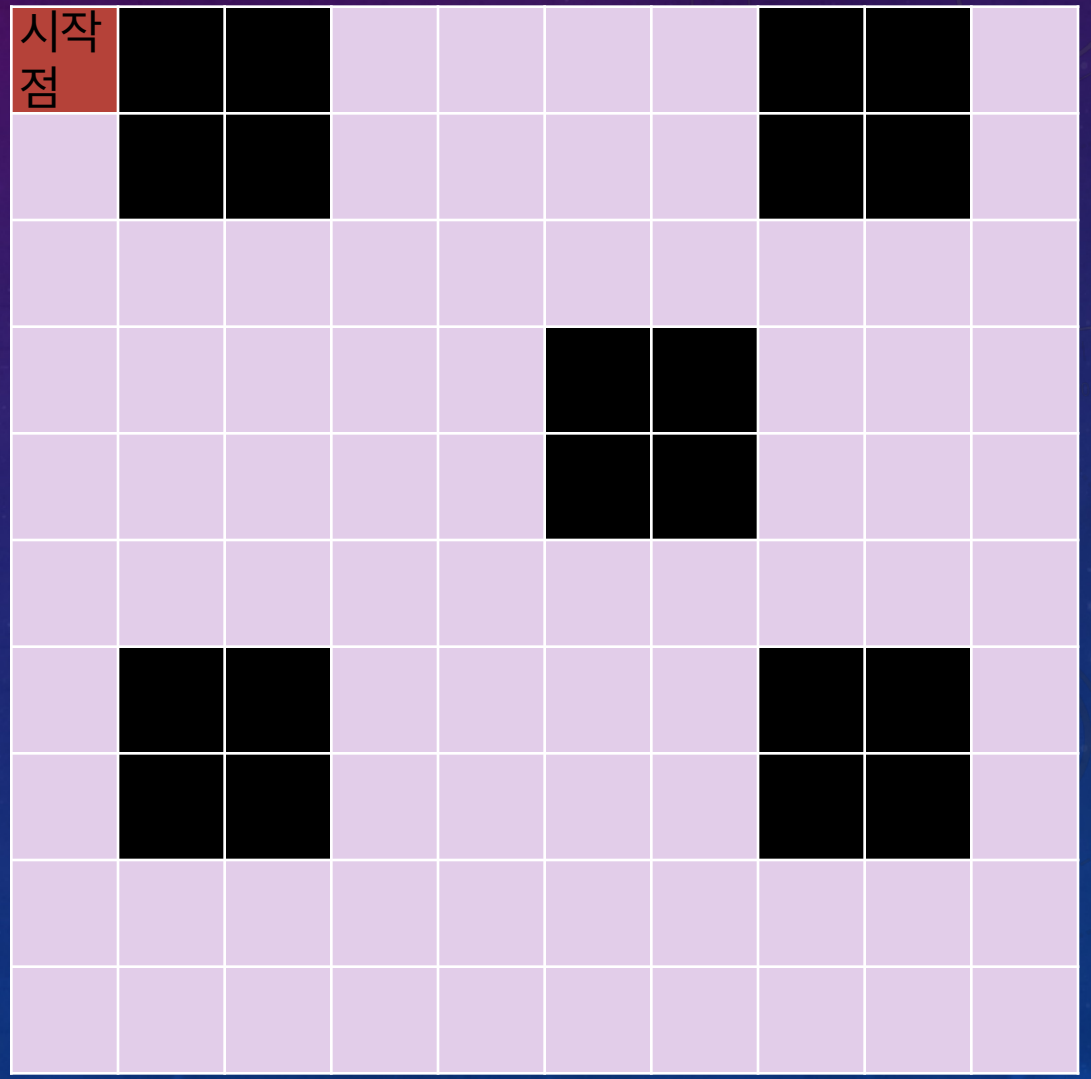


# 패턴인식

- 분해한 문제들 사이에서 규칙성이나 유사성 찾아내기
- 혹은 수학적 규칙성 찾아내기
- (지극히 개인적 의견): 수학적 규칙성을 찾아내는 단계이자, 이 패턴인식까지 무난하게 넘어가면 40%정도 넘어간것.

# 패턴인식

- 장애물은 랜덤하게 배치되는 것이니 “패턴”이라고 정의하기 힘들다.
- 단, 청소기가 움직이는 경로에 있어서 동-서-남-북 이렇게 4방향으로만 움직일 수 있고
- x축을 기준으로 오른쪽으로 가면 (present_x_coord) + 1이 성립되고, Y축도 X축과 마찬가지이다.
- 장애물이 위치해 있는 칸은 항상 1로 표시되어있다.
- 매 순간 청소기가 이동할때마다 칸의 값을 확인할 수 있다.

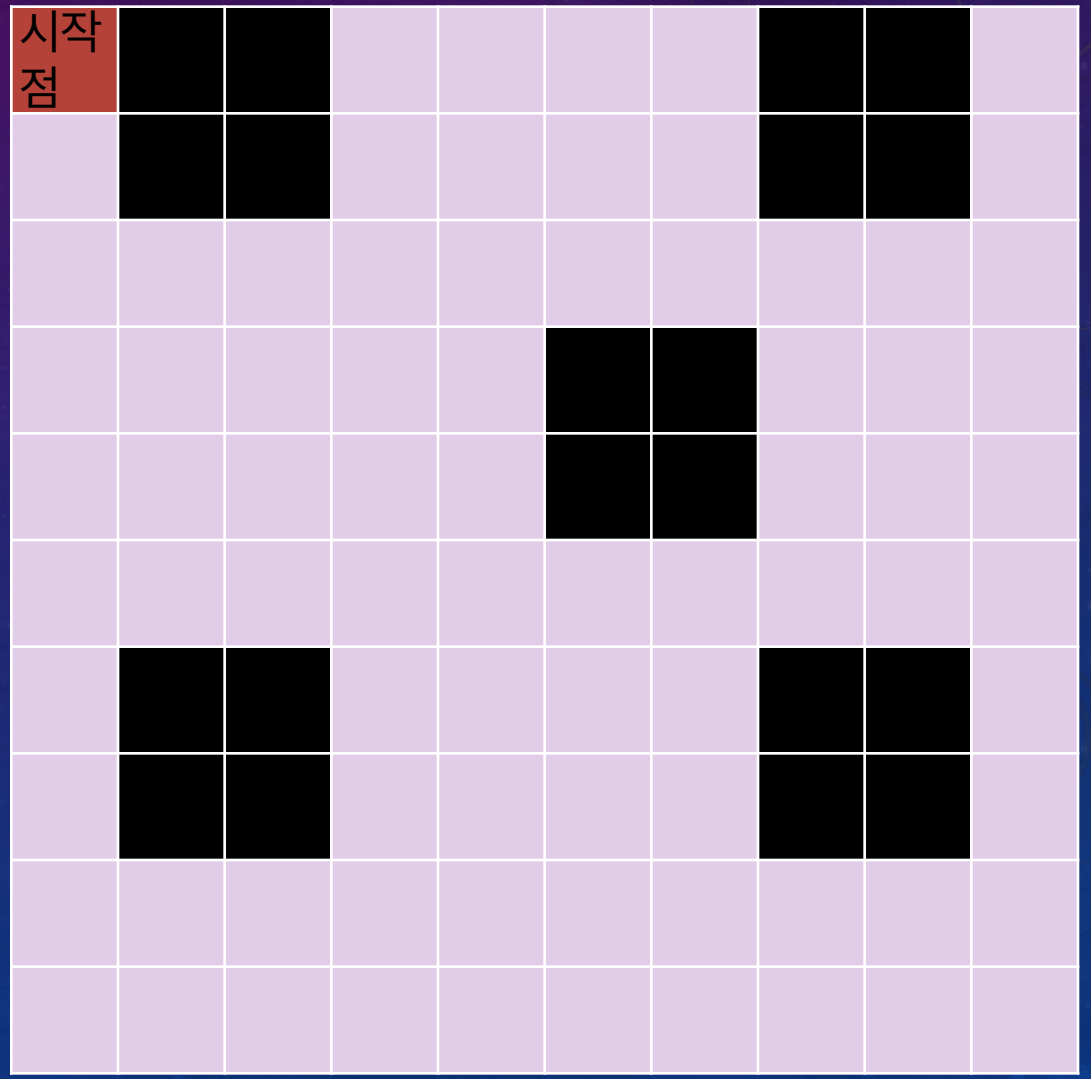


# 추상화

- 패턴 인식 후, 필요 없는 정보나 관련 (수학) 식들을 소거해나감.
- 여기서부터 필요한 것에만 집중
- 패턴인식에서 찾은 규칙을 이용해 핵심적인 개념에 초점을 맞추어 일반적 원리 도출
- 결국 이 단계는 “일반적 원리”를 찾아내는 것이 목적임.

# 추상화

- moveDown
- moveUp
- moveRight(counter_right)
- moveLeft(counter_right)

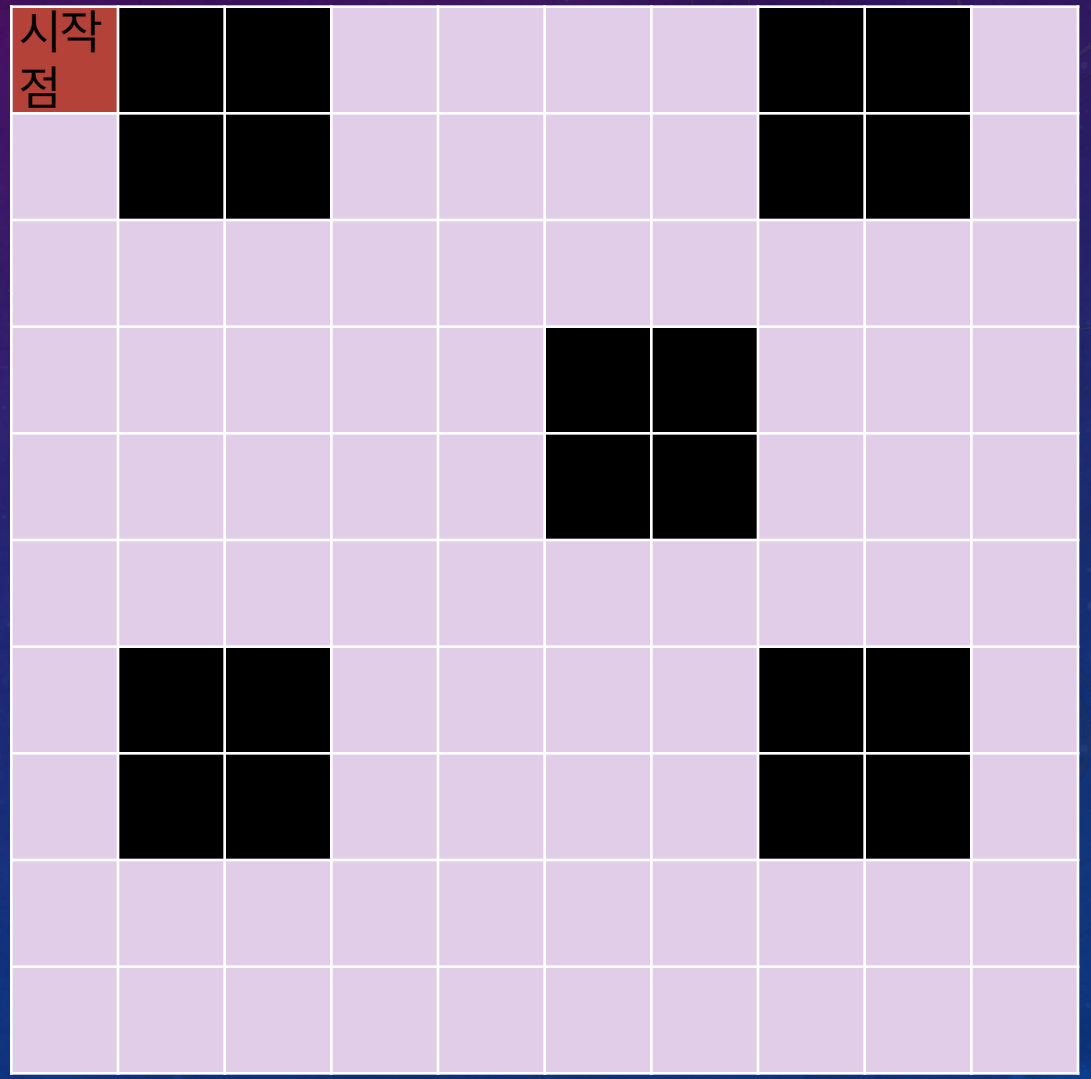


# 알고리즘

- 문제 해결의 방법을 순서대로 나타낸 것 –> 알고리즘!
- 결국 수학 서술형 답안지에서 풀이 과정에 해당하는 것.
- eg: 요리 레시피, 비상탈출 안내도 등
- 이 단계에서는 알고리즘을 도출해내어 수기/컴퓨터 타이핑으로 풀이 과정이 어느 정도 완성해야됨.

# 알고리즘

- moveDown(till —> endOfSquare)
  - 만약 (장애물 존재)
    - moveRight(counter_right)
      - 만약 (아래 장애물이 존재하지 않음)
      - moveDown(till —> no_exist_obs)
      - moveLeft(counter_right)



# 평가

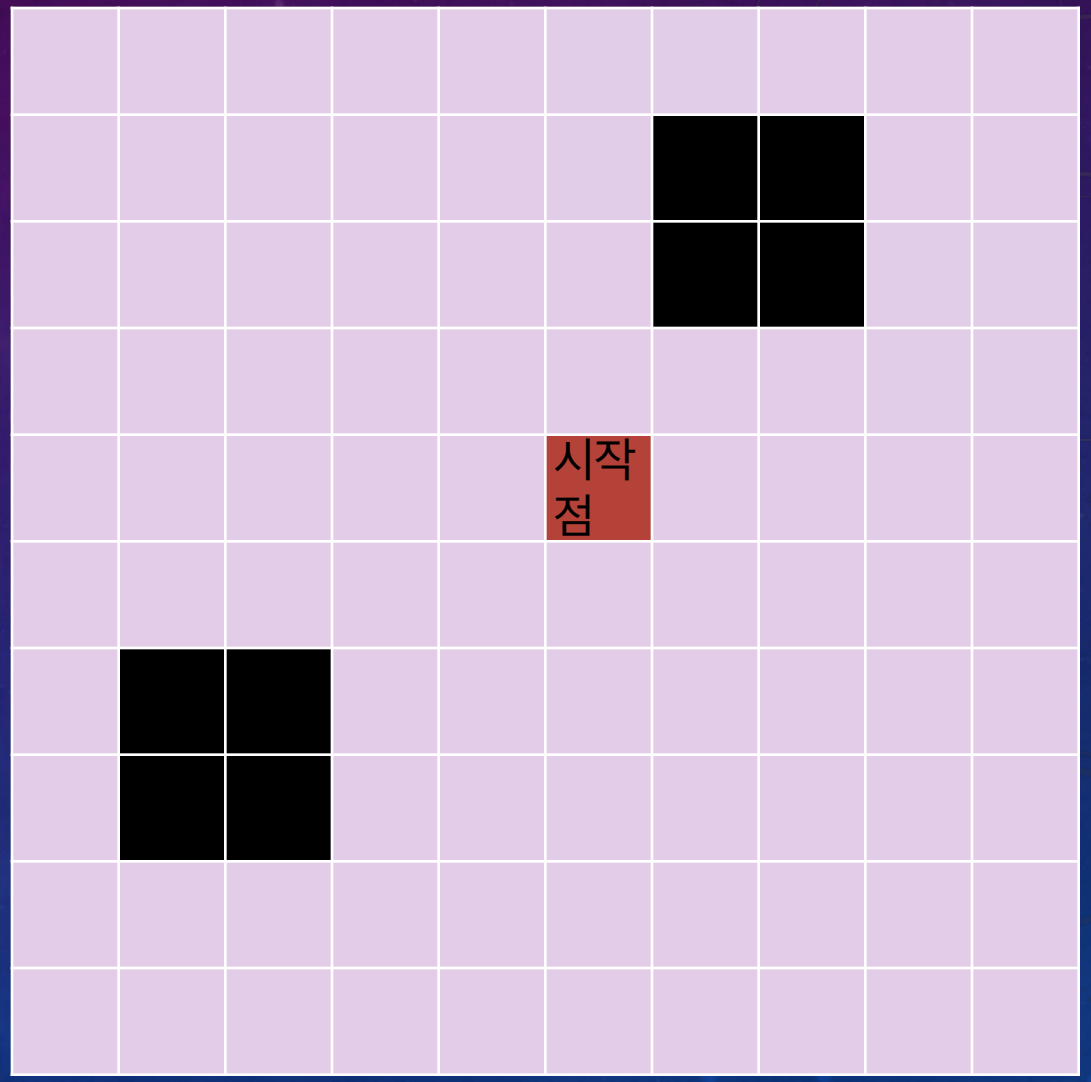
- 과연 이 알고리즘(풀이 과정)이 효율적인가?
- 이 알고리즘이 문제에서 요구하는 조건을 만족 시켰는가?
- 어떠한 상황이 와도 이 알고리즘은 일반적으로 작동하는가?
- 결국, 최종 점검 및 단계

# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER - 조건 CHANGE

- 과연 시작점이 고정이 아닌 랜덤이라면?

# EG: OPTIMIZATION OF WAY FOR ROBOTIC CLEANER - 장애물 예시

```
0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 1 1 0 0 0
1 1 0 0 0 0 0 0 1 1
1 1 0 0 0 0 0 0 1 1
1 1 0 0 0 5 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```



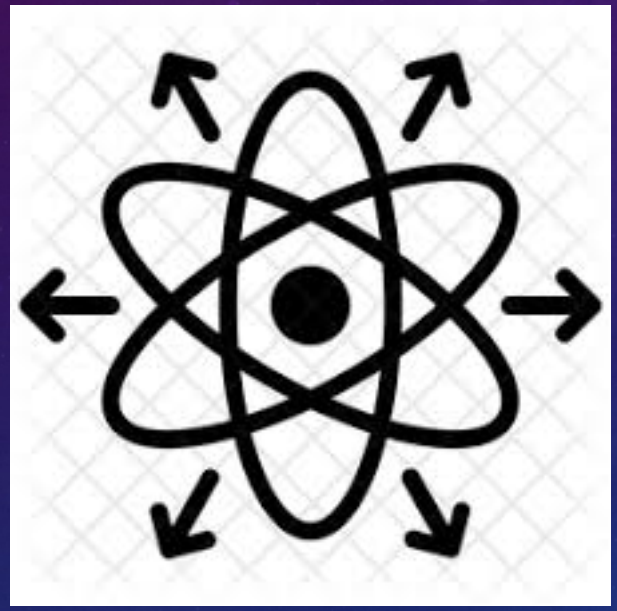
# 도대체 왜 이런 문제를???

# 도대체 왜 이런 문제를???

# 핵심, MAIN

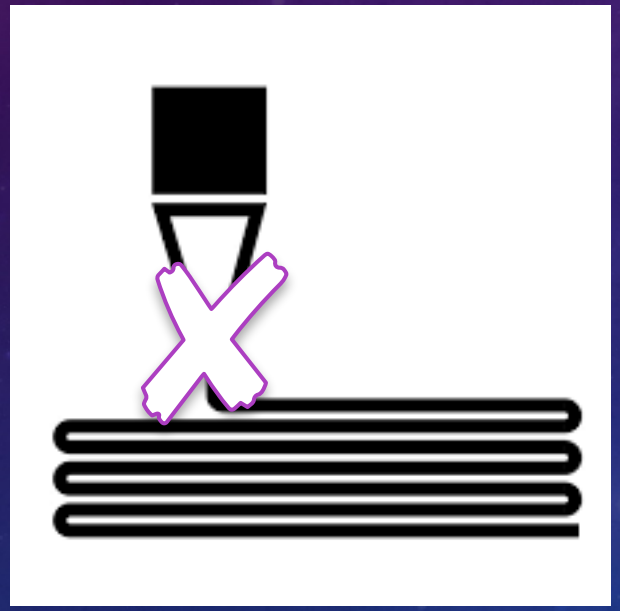
# 핵심, MAIN

# 3D 프린터에서(이-공학계열)의 적용



# Open-Possibility

# 3D 프린터에서의 적용 - 예시(1)



노즐 막힘

# 3D 프린터에서의 적용 - 예시(1) - 시나리오

- 메인보드 문제(전력 분산 관련 오류 가능성)
- 모터관련 문제(E 모터 자체의 가능성, 스텝드라이버의 오류 - 전압 부족, fatal error 가능성)
- 노즐의 문제(노즐 안이 두껍게 막힌 경우)
- 물리 구조적 문제(Teflon튜브 등)

# 3D 프린터에서의 적용 - 예시(1) - 해결책(메인보드)

- Case 1. 메인보드 쇼트 및 전력 분배: 5V레귤레이터 교체
- Case 2. 메인보드 전체 교체

단, 2번은 1번보다 돈이 많이 들어가
효율성이 떨어질 가능성 있음

# 3D 프린터에서의 적용 - 예시(1) - 모터 관련

- Case 1. E모터의 자체적 결함
- Case 2. 스텝드라이버 전압 부족
- Case 3. 스텝드라이버 쇼트 혹은 자체적 결함

- 모터 교체 + 모터 선 교체 시도
- 스텝드라이버 전압 체크 및 조절
- XYZ축의 스텝드라이버 중, E축의 스텝드라이버와 교체해보기

# 3D 프린터에서의 적용 - 예시(1) - 해결책(노즐 관련)

- Case 1. 노즐 안에 필라멘트가 꼬여 clotting현상 발생
- 이건 답없음, 노즐 교체 필요

# 3D 프린터에서의 적용 - 예시(1) - 물리

- Case 1. 테프론 튜브 상당히 휨 및 마찰 증가
- Case 2. E 모터의 기어(MK8/MK7)에 필라멘트 가루가 많이 끼어 마찰력 감소로 인해 필라멘트 압출 불가

- 테프론 튜브 교체 및 필라멘트 직경과 테프론 내경 확인
- 모터를 프린터에서 분리시켜 기어 청소(휴지로 닦으면 잘됨)

# 3D 프린터에서의 적용 - 실전

- Auto Home이 안되는 상황
- 프린터는 “드드드득” 이라는 굉음과 함께 오토홈이 진행이 되질 않음
- 결국 오토홈이 성공적으로 되지 않자 “Homing Failed”라고 뜨고 프린트가 더이상 작동하지 않음(재부팅 필요)
- 그럼 오토홈이 안되는 이유는 무엇일까? 한번 생각해보자