

# 인공지능 프로젝트 보고서

## Wumpus World



컴퓨터정보공학부 201821234 김 진경  
미디어기술콘텐츠학과 201821406 이 재용  
컴퓨터정보공학부 202020300 주 채연

## < 목차 >

### 1. 서론

-문제 정의

1) Wumpus World

-역할 분배 및 프로그램 환경설정

1) 역할 분배

2) 프로그래밍 언어

3) UI

### 2. 본론 - 알고리즘 설명 및 구현방법

-Agent 구조

1) Percept, Reasoning, Act

-격자이동 알고리즘

2) State 유지 방법

3) UI 구현 방법

-알고리즘 정의 및 구현

1) World 초기화

2) Reasoning 알고리즘

3) 탐색 알고리즘

-가보지 않은 곳 탐색

-최단 경로 탐색

### 3. 결론 - 프로그램 실행

-프로그램 요약

-프로그램 예제 적용

### 4. 논의

-더 효율적인 알고리즘 적용

-프로젝트 시 아쉬웠던 부분

-발표 후 Comments★

# 참고자료, 함수설명, Github주소

## 1. 서론

-Domain 정의 : Wumpus World

4 \* 4 로 이루어진 동굴 어딘가에 숨은 Wumpus괴물, Pitch(1/10의 확률로 생성), 하나의 Gold

Wumpus와 Pitch과 같은격자에 있게된다면 죽는다

Wumpus와 Pitch 주위 격자에는 stench, breeze를 받아들일 수 있다

Agent는 동굴에 대한 정보 없이 (1, 1), East에서 시작해 Gold를 찾는 탐험을 한 후 다시 (1, 1)로 돌아와 World를 빠져나오는 것을 목표로 한다

World : 실제 월드

	S	W	P	G	
	B	P	B	O	
	O	B	O	B	
	O	O	B	P	

-Domain 분석 :

Wumpus World Domain은 탐색한 노드 State를 유지하며 Protocol에 기반해 Percept, Reasoning, Act를 반사하는 **Simple Reflex Agent with State**가 적합하다

Partially: 전체환경 중 Agent 탐색한 노드만 볼 수 있으므로 Partially하다.

Deterministic: 현재상태에 의해 다음상태가 명확히 정해지므로 Deterministic하다.

Sequential: 다음 격자까지 이동 등 state간의 의존성이 존재하므로

Sequential하다. Static: 생각하는동안 Agent의 환경은 변하지 않으므로 Static하다.

Discrete: Percepts와 Actions가 유한집합이므로 Discrete하다.

목표: 격자를 탐험하면서 금을 찾아 (1, 1)로 되돌아오는 것

또한 효율적으로 구현하기 위해 최대한 죽지 않고, 최단거리로 탐색하며 프로그램을 수행한다

-프로그램 환경

1) 프로그래밍 언어

python 3.11.3을 이용해 프로젝트를 수행한다

2) UI

tkinter를 이용하여 python으로 간단한 텍스트맵으로 구현

## -역할 분배

김진경 : 아이디어 설계, 주코드 구현, 발표 및 보고서 제작

이재용 : 아이디어 설계, 부코드 구현, 보고서 제작

주채연 : 아이디어 설계, 부코드 구현, ppt 및 보고서 제작

## 2. 본론 - 알고리즘 설명 및 구현방법

### -World 초기화

World는 6\*6의 2차원 배열로 이루어진다

S: Stench, B: Breeze, O: Safe, P: Pitch, W: Wumpus,

G: Glitter, Gold, X: Wall 로 저장시킨다.

각 격자마다 1/10확률로 W, P를 생성 후 그에 맞게

Update해준다

■Gold는 월드에 하나 생성, 각 장애물은 반드시 1개이상 생성된다

### -Agent 구조

#### 1) State 유지 방법

Agent의 현재위치, 현재방향, 생사여부, 화살갯수,

그리고 6\*6의 탐색한 맵(state history)를 저장하고 이를

Update시키면서 진행한다.

### Agent 구조 : reflex agent with state

```
class Agent:
    def __init__(self):
        self._curLoc = (1, 1) # 현재위치
        self._curDir = 'East' # 현재방향
        self._isAlive = True # 살았는지
        self._arrowCnt = 2 # 화살갯수
```

### explore

: agent가 탐색하는 월드 (6 by 6)

	?	?	?	?	
	?	?	?	?	
	?	?	?	?	
	A	?	?	?	

? 격자의 모든 경우의 수  
간단히 문자로 표현 (텍스트맵)

O->Safe  
S->Stench                      B->Breeze  
SB->Stench,Breeze        W->Wumpus  
P?->Pitch?                P->Pitch  
W?->Wumpus?              G->Gold,Glitter

update 될 격자의 모든 경우의 수는 위와 같이 문자로 변환하여  
World, Explore의 6\*6격자에 간단히 텍스트맵으로 저장, 표현한다

## 2) Percept, Reasoning, Act

### -Percept

실제 World에서 Sensor를 통해 Percept를  
받아온다

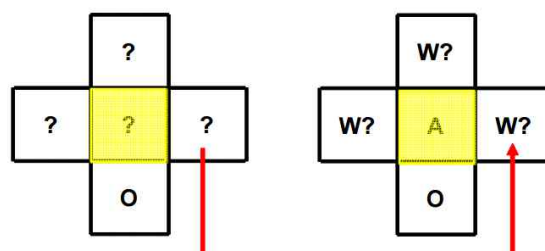
- > S일 때: [Stench, None, None, None, None]
- > B일 때: [None, Breeze, None, None, None]
- > SB일 때: [Stench, Breeze, None, None, None]
- > O일 때: [None, None, None, None, None]
- > P일 때: Die
- > W일 때: Die
- > G일 때: [None, None, Glitter, None, None]
- > 벽에 부딪힐 때: [None, None, None, Bump, None]
- > 화살을 쏜 곳에: Wumpus가 존재할 때  
[None, None, None, None, Scream]

### -Reasoning:

받아온 Percept에 따라 Reasoning해서 Explore를 업데이트  
State History와 현재 State를 합쳐서 추론  
Percept를 받아와 저장된 State와 비교해 새롭게  
Update시켜준다 # 주변격자가 새로 Update되는 경우의 수  
> S일 때: ?->W?, P?->O, WP?->W? (1)

**[Stench, None, None, None, None]** : 해당 격자 S로 update

A = S 으로 업데이트



방문되지않은 곳을 W?로 업데이트

> B일 때:  $? \rightarrow P?$ ,  $W? \rightarrow O$ ,  $WP? \rightarrow P?$  (2)

> SB일 때:  $? \rightarrow WP?$

> O일 때:  $W? \rightarrow O$ ,  $P? \rightarrow O$ ,  $? \rightarrow O$  (3)

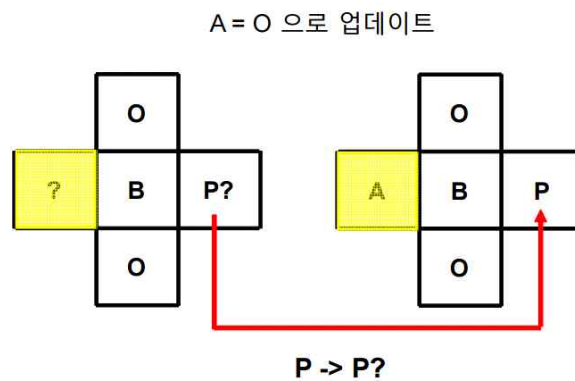
> P일 때:  $P? \rightarrow P$ ,  $WP? \rightarrow P$

> W일 때:  $W? \rightarrow W$ ,  $WP? \rightarrow W$

> G일 때:  $\rightarrow G$

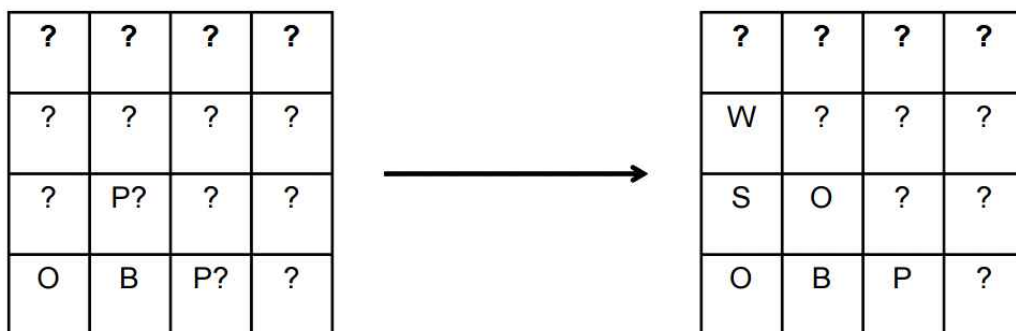
>>(1), (2), (3)의 O로 변경되는 경우

변경된 O의 상하좌우가 S또는 B일 때 그 S 또는 B의 상하좌우의 W? P?인지 판단하여 W 또는 P를 확정시켜준다



위 그림과 같이 가고자할 격자 주변에 B가 있을때 격자가 O로 업데이트 되면서 P?였던 격자를 P로 업데이트한다. (W, P 확정)

이는 Stench에서도 똑같이 적용(Symmetric)



이를 응용하여 (2,1)(y좌표, x좌표)에서 Stench를 받아들이면 (2,2)P?가 O로 바뀌면서 그 격자 주변의 S또는 B가 있다면 위의 업데이트를 실행해준다

★

×W, P를 확정함으로써 얻을 수 있는 이득은 나중에 안전한 곳이 모두 탐색되고, W, W?순으로 화살을 쏠 때, 화살이 없어 W?, P?로 죽으러 갈 때 얻을 수 있다.

- W가 확정된다면 화살을 낭비하지 않고

- W,P가 확정된다면 W?P?로 정보를 얻으면서 죽으러 갈 때, 다른 위치의 정보를 얻을 수 있다.

(그곳은 안전한 곳일지도 모른다)

이와 같은 Percept에 따른 각각의 추론들을 Explore에 업데이트 해준다

그리고 그에 맞는 Act를 취해준다

-Act:

> 격자를 이동할 때 : 방향에 따라

Go Forward, Turn Left, Turn Right

> 금을 발견했을 때 : Grab

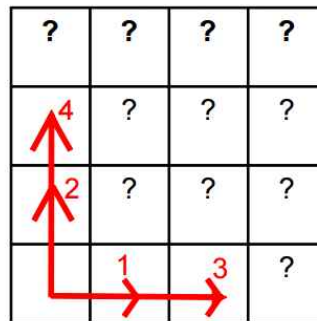
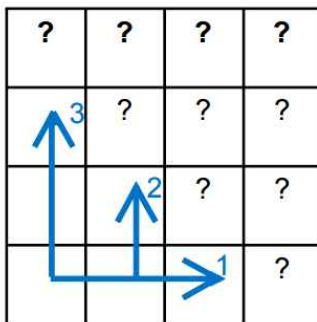
> 모든 가능한 곳의 탐색이 끝난후 (1, 1)에 돌아왔을 때 : Climb

### # 탐색(격자이동) 알고리즘

#### DFS를 이용

가보지 않은 곳을 탐색할 때에는,

다시 돌아오는 것을 생각해 갈 수 있는 한 깊이 탐색 후 돌아오는 DFS를 채택하였다



DFS이용하여 방문 : 파란색 선  
BFS이용하여 방문 : 빨간색 선

만약 BFS로 가보지 않은 곳을 탐색한다면 왔던 길을 반복해 돌아가야 하므로 비효율적이다

#### -최단 경로 탐색

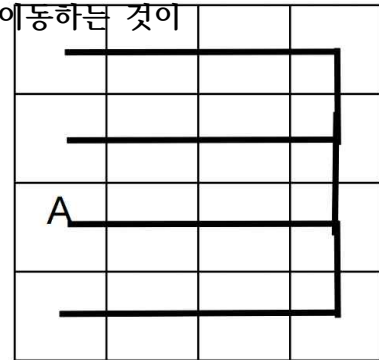
##### BFS를 이용

DFS 안에서 다음노드가 상하좌우가 아닐 때,

W?, W를 찾아 화살을 쏠 수 있는 자리로 이동할 때,

더 이상 갈 수 있는 곳이 없어 W?, P?를 몸으로 탐색할 때, 등

이미 지나온 경로들을 다시 탐색해야할 때, 최단 경로로 이동하는 것이 목표이므로 BFS로 최단경로를 찾아 탐색한다



다음노드  
(0,1)

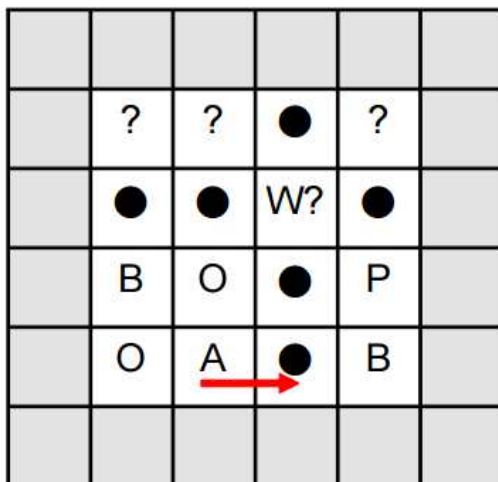
-안전한 곳을 모두 방문한 후

bfs로 history에 존재하는 w, w? 를 찾고,

W->W?순으로 Wumpus가 있는 곳의 같은 x,y좌표로 이동하여 화살을 소비해 죽인다

그 후 Sensor로 Scream여부가 들어온다

### explore



Shoot arrow from (1, 3) to (3, 3)

[None, None, None, None, Scream]

-만약 화살이 부족하다면 w? p?를몸으로 탐색하거나 이가 없을때에는 자살해 (1, 1), East, 화살 2개로 다시시작합니다

(죽었을때 explore를제외한 agent의 state를 초기값으로 초기화해준다)

화살을 쏜 후 Scream이 들어오면 월드가 바뀐다. 만약 Scream이 들어오지않았더라도, 그곳은 안전한 곳이므로,

-화살을 쏜 격자와 격자 주변의 방문여부를 초기화해주고, 다시 dfs 탐색해 새로생긴 안전한곳을 모두 탐색합니다. 그 후 W나 W?가 새로 생기면 다시 위를 반복한다



-W나 W?가 history에 없다면 몸으로 W?, P?를 탐색합니다  
그 후 W나 W?가 생기면 또다시 위를 반복한다

-이 모든것이 실행된 후에도 찾을 수 없다면 탐색 실패를 출력하고 (1, 1)로  
돌아와 Climb한다

-반대로 실행 중 Glitter가 들어온다면 금을 grab을 출력하고 (1, 1)로 돌아와  
Climb해 종료

### 3) UI 구현

처음 월드 초기화시 실제 월드를 보여준다

격자이동시마다 현재 좌표와 Percept, Explore(State History)를 출력한다

(단, 벽에 부딪힐 때,

BFS로 격자이동시에는 이미 방문한 노드이므로 Explore를 출력하지 않는다)

액션시 액션을 출력한다

Wumpus\_World

World INIT... The World is

[	'	X	'	'	X	'	'	X	'	'	X	'	'	X	'	'	X	'	]
[	'	X	'	'	S	'	'	O	'	'	O	'	'	G	'	'	X	'	]
[	'	X	'	'	W	'	'	S	'	'	O	'	'	O	'	'	X	'	]
[	'	X	'	'	S	'	'	O	'	'	B	'	'	O	'	'	X	'	]
[	'	X	'	'	O	'	'	B	'	'	P	'	'	B	'	'	X	'	]
[	'	X	'	'	X	'	'	X	'	'	X	'	'	X	'	'	X	'	]

State History:

[	'	X	'	'	X	'	'	X	'	'	X	'	'	X	'	'	X	'	]
[	'	X	'	'	?	'	'	?	'	'	?	'	'	?	'	'	X	'	]
[	'	X	'	'	W	'	'	?	'	'	?	'	'	?	'	'	X	'	]
[	'	X	'	'	A	'	'	O	'	'	?	'	'	?	'	'	X	'	]
[	'	X	'	'	O	'	'	B	'	'	P	'	'	?	'	'	X	'	]
[	'	X	'	'	X	'	'	X	'	'	X	'	'	X	'	'	X	'	]

Agent is go to  
(2, 2)  
TurnRight GoForward

### 3. 결론

-프로그램 요약

- 1) 시작노드인 1,1은 무조건적으로 안전하고 각 장애물은 적어도 하나 존재
  - 2) 1,1에서 dfs시작 -> dfs(1,1) 상하좌우 연결된 안전한 곳 모두 탐색
    - 다음 dfs노드가 연결되지않았다면,
    - bfs로 다음노드까지 간 후, 탐색
    - (모든 탐색에서 방향에 알맞은 액션 출력)
    - ex)Turn Left, Go Forward
  - 3) 안전한 곳을 모두 탐색 후 Explore(state history)의 W, W?가 있다면 W, W? 순으로 화살을 쏘 Wumpus Kill
  - 4) 쏜 곳은 확실히 안전한 곳이므로(업데이트) 쏜 격자, 주변을 dfs로 재방문  
(방문여부 초기화)
  - 5) 화살이 모자라거나 더 이상 W, W?를 Explore에서 찾을 수 없다면 W?, P?를 몸으로 탐색(방문하지 않은 곳이므로 DFS탐색) 없을 때 자살
  - 6)위 과정에서 중간에 W, W?가 발견되면 3)을 반복
  - 7)위 과정에서 Gold를 발견하면 과정을 멈추고 Grab후 (1, 1)로 돌아와 Climb
  - 8)위 과정이 모두 반복되고 더 이상 갈 곳이 없을 때 Gold를 찾지 못했다면 탐색실패를 출력 후 (1, 1)로 돌아와 Climb
- (1, 1)로 돌아올때는  
이미 방문했던 격자들을 되돌아오는 것이므로 최단경로탐색 BFS를 이용

## - 데모 실행

Wumpus\_World

World INIT... The World is

```
[[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'S' ' ' 'S' ' ' 'O' ' ' 'X' ' ' ]
[ 'X' ' ' 'S' ' ' 'W' ' ' 'W' ' ' 'S' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'S' ' ' 'G' ' ' 'B' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X' ' ' ]
[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]]
```

State History:

```
[[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]]
```

Agent is go to

(1, 1)

[ ' None ' ' None ' ' None ' 'None' ' None' ]

Wumpus\_World

World INIT... The World is

```
[[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'S' ' ' 'S' ' ' 'O' ' ' 'X' ' ' ]
[ 'X' ' ' 'S' ' ' 'W' ' ' 'W' ' ' 'S' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'S' ' ' 'G' ' ' 'B' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X' ' ' ]
[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]]
```

State History:

```
[[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' 'A' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]]
```

Agent is go to

(1, 2)

GoForward

[ ' None ' ' None ' ' None ' 'None' ' None' ]

Wumpus\_World

World INIT... The World is

```
[[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'S' ' ' 'S' ' ' 'O' ' ' 'X' ' ' ]
[ 'X' ' ' 'S' ' ' 'W' ' ' 'W' ' ' 'S' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'S' ' ' 'G' ' ' 'B' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X' ' ' ]
[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]]
```

State History:

```
[[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' 'O' ' ' 'A' ' ' '?' ' ' '?' ' ' 'X' ' ' ]
[ 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' ]]
```

Agent is go to

(1, 3)

GoForward

[ ' None ' 'Breeze' ' None ' 'None' ' None' ]

World INIT... The World is

```

[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]
[['X' ' ' 'O' ' ' 'S' ' ' 'S' ' ' 'O' ' ' 'X']]
[['X' ' ' 'S' ' ' 'W' ' ' 'W' ' ' 'S' ' ' 'X']]
[['X' ' ' 'O' ' ' 'S' ' ' 'G' ' ' 'B' ' ' 'X']]
[['X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X']]
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]

```

State History:

```

[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]
[['X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X']]
[['X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X']]
[['X' ' ' '?' ' ' '?' ' ' 'P?' ' ' '?' ' ' 'X']]
[['X' ' ' 'O' ' ' 'O' ' ' 'A' ' ' 'P?' ' ' 'X']]
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]

```

(1, 2)

TurnLeft TurnLeft GoForward

Agent is go to

(2, 2)

TurnRight GoForward

['Stench' ' ' None ' ' None ' 'None' ' ' None']

World INIT... The World is

```

[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]
[['X' ' ' 'O' ' ' 'S' ' ' 'S' ' ' 'O' ' ' 'X']]
[['X' ' ' 'S' ' ' 'W' ' ' 'W' ' ' 'S' ' ' 'X']]
[['X' ' ' 'O' ' ' 'S' ' ' 'G' ' ' 'B' ' ' 'X']]
[['X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X']]
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]

```

State History:

```

[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]
[['X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X']]
[['X' ' ' '?' ' ' 'W?' ' ' '?' ' ' '?' ' ' 'X']]
[['X' ' ' 'W?' ' ' 'A' ' ' 'O' ' ' '?' ' ' 'X']]
[['X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X']]

```

Agent is go to

(2, 3)

TurnRight GoForward

['None' ' 'None' ' 'Glitter' 'None' ' ' None']

World INIT... The World is

```

[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]
[['X' ' ' 'O' ' ' 'S' ' ' 'S' ' ' 'O' ' ' 'X']]
[['X' ' ' 'S' ' ' 'W' ' ' 'W' ' ' 'S' ' ' 'X']]
[['X' ' ' 'O' ' ' 'S' ' ' 'G' ' ' 'B' ' ' 'X']]
[['X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X']]
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]

```

State History:

```

[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]
[['X' ' ' '?' ' ' '?' ' ' '?' ' ' '?' ' ' 'X']]
[['X' ' ' '?' ' ' 'W?' ' ' '?' ' ' '?' ' ' 'X']]
[['X' ' ' 'W?' ' ' 'S' ' ' 'A' ' ' '?' ' ' 'X']]
[['X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X']]
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X']]

```

Find gold!

Grab

way home

(2, 2)

TurnLeft TurnLeft GoForward

(1, 2)

TurnLeft GoForward

(1, 1)

TurnRight GoForward

Agent is go to

(1, 1)

['None' ' 'None' ' 'None' ' 'None' ' ' None']

World INIT... The World is

Climb

```
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ']]
[['X' ' ' 'O' ' ' 'S' ' ' 'S' ' ' 'O' ' ' 'X' ' ']]
[['X' ' ' 'S' ' ' 'W' ' ' 'W' ' ' 'S' ' ' 'X' ' ']]
[['X' ' ' 'O' ' ' 'S' ' ' 'G' ' ' 'B' ' ' 'X' ' ']]
[['X' ' ' 'O' ' ' 'O' ' ' 'B' ' ' 'P' ' ' 'X' ' ']]
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ']]
```

State History:

```
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ']]
[['X' ' ' ' ? ' ' ' ? ' ' ' ? ' ' ' ? ' ' 'X' ' ']]
[['X' ' ' ' ? ' ' ' W? ' ' ' ? ' ' ' ? ' ' 'X' ' ']]
[['X' ' ' ' W? ' ' ' S ' ' ' O ' ' ' ? ' ' 'X' ' ']]
[['X' ' ' ' A ' ' ' O ' ' ' B ' ' ' P ' ' 'X' ' ']]
[['X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ' 'X' ' ']]
```

## 4. 논의

-프로젝트 시 아쉬웠던 부분

1) 격자이동시 각 격자마다의 W, P의 확률을 계산해 있을 확률이 적은 격자를 먼저 탐색하는 알고리즘을 적용했다면 더 효율적인 Agent가 될 것 같습니다.

2) 더 나은 GUI 모듈 또는 프로그램으로 구현했다면 프로그램의 퀄리티가 더 높아질 것인데 퀄리티에 대한 부분이 아쉽습니다.

-발표 후 Comments★

아래 사진의 Reasning에 대한 질문 (6페이지 아래쪽 별표)

(y좌표, x좌표)로 격자를 표현할 때

(2, 1)이 ?에서 S로 변경된다면(격자 방문, Agent의 위치)

(2, 2)P?->O가 됨으로써 (3, 1), (1, 3)이 각각 W, P로 확정됩니다

수업에서는 이를 KB에서 Unit Resolution의 방식으로 추론했는데 이는 복잡도가 너무 커집니다

따라서 이 부분만 떨어뜨려서 Agent가 새로 방문한 격자가 Stench일 때,  
if Percept[0] == 'Stench' :

주변 노드의 state history가 P?라면

for i in [현재위치의 상하좌우노드들]:

if explore[i] == ' P?' :

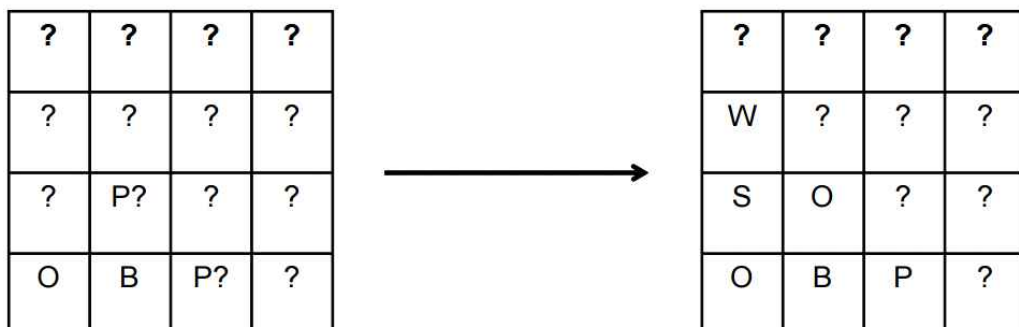
explore[i] = ' O ' #O를 업데이트해주고

#그 주변의 격자들이 S 나 B라면 주변에 따라 W, P 확정

for j in [i의 상하좌우노드들]:

if explore[j] == ' S ' or explore[j] == ' B '

주변에 따라 W, P 확정



위를 통해 (W, P를 확정함으로써) 얻을 수 있는 이득은 나중에 안전한 곳이 모두 탐색되고, W, W?순으로 화살을 쏘 때, 화살이 없어 W?, P?로 죽으러 갈 때 얻을 수 있습니다.

- W가 확정된다면 화살을 낭비하지 않고

- W, P가 확정된다면 W?P?로 정보를 얻으면서 죽으러 갈 때, 다른 위치의 정보를 얻을 수 있으므로 정보가 유의미할 것입니다.

● 위예제를 World로 하여 데모로 따로 실행한 영상도 github에 같이 올려두었습니다

#참고문헌 및 자료: 인공지능 수업ppt자료

코드 및 실행 파일 github 주소:

[https://github.com/realview45/Wumpus\\_World\\_AI/tree/main](https://github.com/realview45/Wumpus_World_AI/tree/main)