**<인공지능 assignment2 보고서>**

2019019016 서시언

**[코드 설명 및 함수 설명]**

**1. def main()**

- input.txt의 내용을 char array 형태로 map에 저장하고, Start point의 위치를 sPoint에 저장해준다.

- map은 1차원 배열로, 5\*5 크기의 맵을 나타낸다.

- map의 첫 번째 row의 각 col은 0~4, 두 번째 row 의 각 col은 5~9, 세 번째 row 의 각 col

은 10~14, 네 번째 row 의 각 col은 15~19, 다섯 번째 row 의 각 col은 20~24 의 index를 갖는다.

- map의 각 element는 ‘S’, ‘G’, ‘T’, ‘B’, ‘P’ 중 하나의 값을 갖는다.

- sPoint에는 element가 ‘S’인 칸의 index가 저장된다.

- open(), read()를 이용해 input.txt의 내용을 읽고, replace(‘\n’, ’’)를 이용해 ‘\n’을 없애준다.

- 읽은 문자열을 list()를 이용해 array로 변환해 map에 저장한다.

- find(‘S’)를 이용해 ‘S’가 있는 칸의 Index를 sPoint에 저장한다.

- map, sPoint를 파라미터로 QLearning 객체 q를 생성하고, QLearning의 learning()을 호출한다.

**2. class QLearning**

- def \_\_init\_\_(self, map, sPoint)

- Q는 estimated utility function을 나타내는데, {a: Q(s,a), …} 형태의 딕셔너리를 25개 갖는

배열으로, Q(s,a)에는 Q[s][a] 형태로 접근할 수 있다.

- s는 state를 의미하며, 0~24의 값을 가지고, map에서 어떤 칸에 있는지를 나타낸다.

- a는 action으로, ‘D’, ‘U’, ‘L’, ‘R’이 들어갈 수 있으며, 이는 각각 down, up, left, right로, 현재

state에서 해당 방향으로 이동한다는 의미이다.

- action은 현재 state에서 해당 방향으로 이동했을 때 다음 state를 계산하기 위한 것으로,

{ a: offset, … } 형태의 딕셔너리이다. 현재 state에서 a action을 취했을 때의 다음 state

는 다음과 같이 계산할 수 있다.

newState = curState + action[a]

- reward는 현재 state에서 특정 action을 선택했을 때 얻을 수 있는 즉각적인 reward를 계산

하기 위한 것으로, { pointKind: reward, … } 형태의 딕셔너리이다. pointKind에는 ‘S’, ‘G’, ‘T’, ‘B’, ‘P’

가 들어갈 수 있다.

- r은 reward가 전파되는 정도를 나타내는 계수이다.

- map, sPoint를 넘겨받은 값으로 초기화하고, initQ()를 호출해 Q를 초기화한다.

- action에 {'D': 5, 'U': -5, 'R': 1, 'L': -1}를, reward에 {'S': 0, 'G': 100, 'T': 10, 'B': -100, 'P': 0}를, r에 0.9를

저장한다.

- def initQ()

- 모든 가능한 s, a 쌍에 대해 Q(s,a)를 0으로 초기화한다.

- s = 0~24에 대해 {'D': 0, 'U': 0, 'R': 0, 'L': 0}로 시작해서 마지막 row (s/5 == 4)이면 ‘D’를, 첫 번째

row(s/5 == 0)이면 ‘U’을, 마지막 col(s%5 == 4)이면 ‘R’를, 첫 번째 row(s%5 == 0)이면 ‘L’을 딕셔

너리에서 제거해, 각 state에 대해 가능한 action만을 남겨두고, Q(s,a)를 0으로 초기화한다.

- def learning()

- Q-Learning 알고리즘에 따라 Q(s,a)를 업데이트한다.

- random.randint(0,24)를 이용해 현재 state를 나타내는 curState에 0~24 사이의 랜덤한 숫자

를 저장한다.

- for문으로 아래 과정을 반복한다.

- map[curState]가 ‘B’(Bomb) 또는 ‘G’(Goal Point)이면, random.randint(0,24)를 이용해

curState에 랜덤한 숫자를 저장해 재시작해준다.

- random.choice(list(self.Q[curState]))를 이용해 현재 state에서 선택가능한 임의의 action을

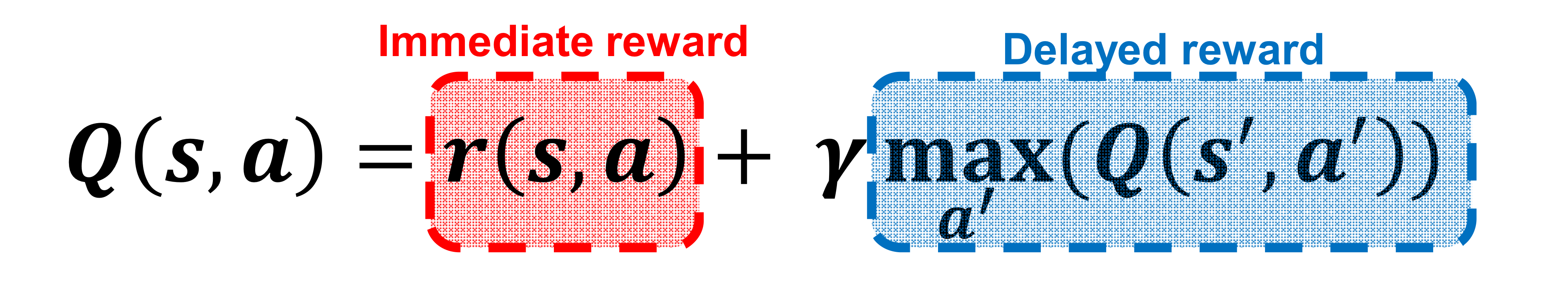
선택해서 현재 action을 나타내는 curAction에 저장한다.

- 새로운 state를 나타내는 newState에 curState + self.action[curAction]을 저장한다.

- immediate reward를 나타내는 curReward에 self.reward[self.map[newState]]를 저장한다.

- self.Q[curState][curAction]에 curReward+ self.r \* max(self.Q[newState].values())를 저장해,

다음 식에 따라 Q(s,a)를 업데이트 해준다.



- curState에 newState를 저장해, 현재 state를 다음 state로 갱신해준다.

- 아래와 같이 for문 반복 횟수를 1000으로 설정했을 때와 10000으로 설정했을 때의 Q

table값이 같기 때문에, 10000이 이 문제에서의 Q table이 수렴하기에 충분히 큰 값이라

생각해 for문 반복횟수를 10000으로 설정했다.

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

- printOutput()을 호출한다.

- def printOutput()

- output.txt에 출발지점에서 도착지점까지 최적의 경로, 출발지점에서의 max Q값을 출력한다.

- curState에 sPoint(Start Point)를 저장한다.

- map[curState]가 ‘G’가 될 때(Goal Point에 도달할 때)까지 아래 과정을 반복한다.

- curState에 curState + self.action[max(self.Q[curState], key = self.Q[curState].get)]을 저장해, Q(s,a)

가 최대가 되도록 하는 a를 선택해 state를 갱신해준다.

- max(self.Q[curState], key = self.Q[curState].get)은 Q[curState][a]가 최대가 되도록 하는 a

를 리턴한다.

- 이때 최대가 되는 a가 여러 개 존재한다면 ‘D’, ‘U’, ‘L’, ‘R’ 순으로 했을 때 가장 앞에

있는 것을 선택한다.

- curState를 output.txt에 출력한다.

- max(self.Q[self.sPoint].values())로 Start Point에서의 max Q값을 구해 output.txt 파일에 출력한다.

**[실험 결과 및 분석]**

**1. input. txt**

- 주어진 input.txt는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명 

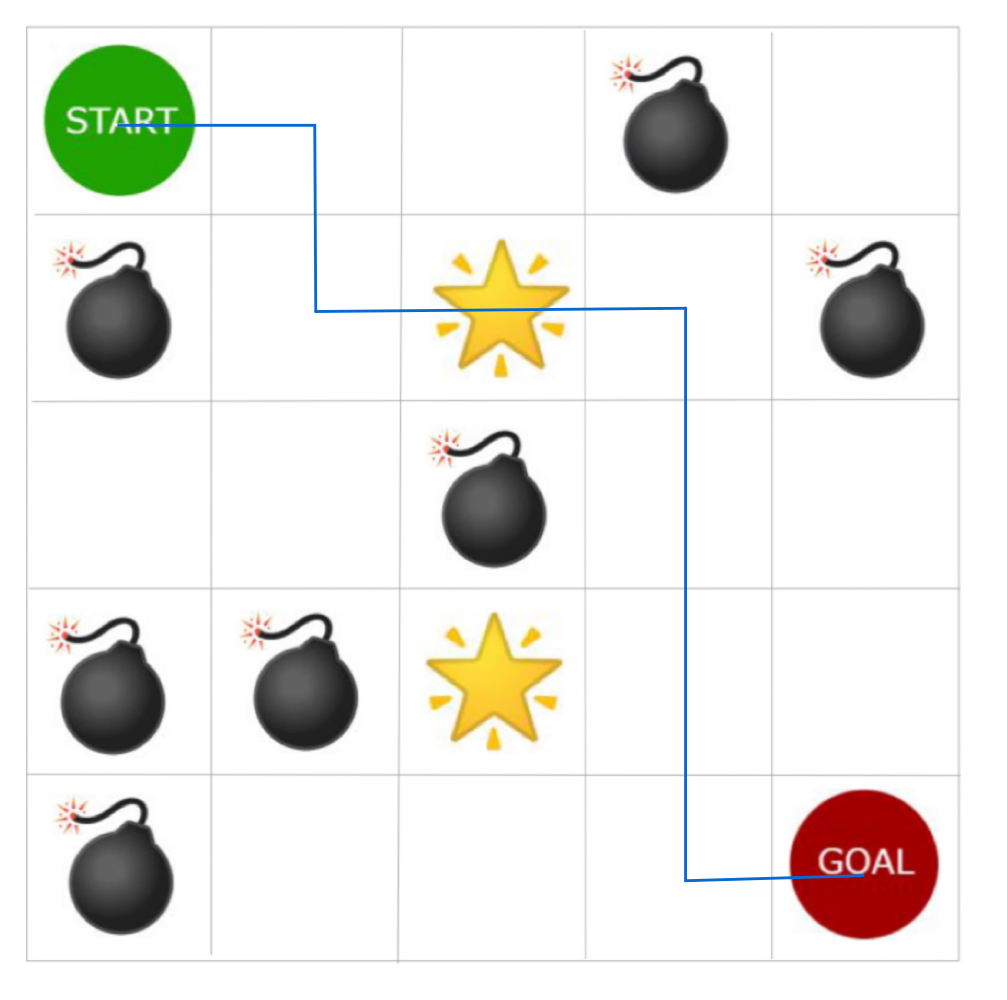
**2. 보너스 지점의 reward가 1일 때**

- output.txt

텍스트이(가) 표시된 사진

자동 생성된 설명

- 경로



- Q table

텍스트이(가) 표시된 사진

자동 생성된 설명

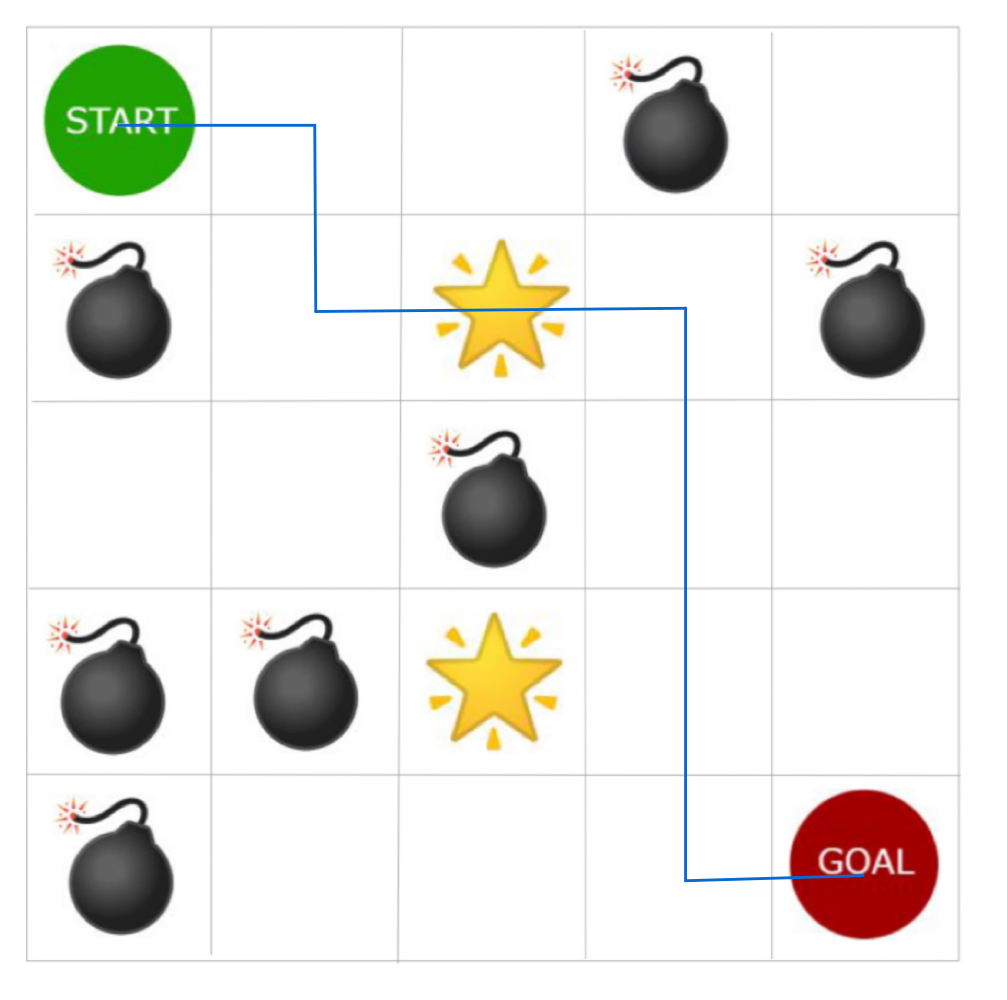
**3. 보너스 지점의 reward = 10**

- output.txt

텍스트이(가) 표시된 사진

자동 생성된 설명

- 경로



- Q table

텍스트이(가) 표시된 사진

자동 생성된 설명

- 7의 아래는 bomb이므로, 7번 state에서 Goal 방향으로 가려면 반드시 ‘R’ action을 취해야한다.

- 7번 state에서 Goal 방향이 아닌 다른 방향으로 가는, ‘U’, ‘L’ action을 취했을 때의 Q값이 bonus point의 reward가 1인 경우보다 더 커졌지만, ‘R’ action을 취할 때의 Q값은 그대로인 것을 알 수 있다.

- 8번 state에서도 마찬가지로, Goal 방향으로 가는 action인 ‘D’를 취했을 때의 Q값은 그대로지만,

bonus point인 7번 state로 가는 ‘L’을 취했을 때의 Q값은 bonus point의 reward가 1인 경우에

비해 증가했음을 알 수 있다.

**4. 보너스 지점의 reward = 20**

- output.txt

텍스트, 전자기기, 컴퓨터, 스크린샷이(가) 표시된 사진

자동 생성된 설명

- 경로



- Q table

텍스트이(가) 표시된 사진

자동 생성된 설명

- bonus point의 reward가 20으로 커지면, Goal point 방향으로 가는 것(Q(7,‘R’))보다, Bonus point와 이

웃한 state(2)와 Bonus point(7) 사이를 왔다 갔다 하는 것(Q(7,‘U’))이 더 커지게 된다.

- 여기서는 Q값이 같은 action이 여러 개 있을 경우 ‘D’, ‘U’, ‘R’, ‘L’ 순서로 했을 때 가장 앞에 있는

것을 선택하기 때문에 ‘U’가 선택되어 7번 state와 그 위에 있는 2번 state를 계속해서 왔다갔다하게 되어 goal에 도달할 수 없게 된다.

- 만약 ‘R’이 선택되어 8번 state로 가더라도, 가장 Q값이 높아지는 action이 ‘L’이기 때문에 7

번과 8번 state를 계속해서 왔다갔다하게 되어 goal에 도달할 수 없다.

- 마찬가지로 ‘L’이 선택되고 6번 state로 가더라도, 가장 Q값이 높아지는 action이 ‘R’이기 때

문에 6번과 7번 state를 계속해서 왔다갔다하게 되어 goal에 도달할 수 없다.