Q-learning

- 어제 배운 Q 알고리즘은 완전한 알고리즘이 아니다! 2 가지의 문제가 있다.

1 한번 길을 찾으면 더이상 모험을 하지 않기 때문에 찾은 길이 항상 **최적의 길이라는 보장이 없다.**
2 최적의 방향이던 아니던 올바른 방향의 값이 **항상 1 이다**!

Exploit & Exploration

- Q 알고리즘이 어느정도 모험을 하게 만들어주는 방법

1 **E-greedy**

- 기준이 될 e 값을 정하고 랜덤한 값을 뽑아 e 값보다 작으면 랜덤하게 실행한다.

```
e = 0.1
if rand < e :
    a = random
else :
    a = argmax(Q(s,a))
```

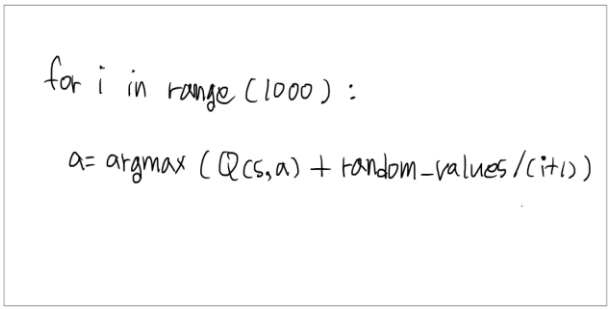
2 **decaying E-greedy**

- 최적의 길을 찾은 이후에도 쓸데없이 랜덤 하게 실행되는 것을 최소화시키는 방법
- 반복 실행할수록 랜덤하게 실행될 확률이 적어진다

```
for i in range(1000):
    e = 0.1 / (i+1)
    if random(1) < e:
        a = random
    else :
        a = argmax(Q(s,a))
```

3 **add random noise**

- 랜덤한 노이즈값을 더해서 큰 값을 따라가는 방법
- 반복 실행할수록 노이즈의 크기를 줄여 노이즈의 영향을 적게 주게 된다.
- 2 개의 E-greedy 알고리즘과 다른점은 완전한 랜덤이 아니라는 점이다. 노이즈가 있어도 원래 큰 값이 선택될 확률이 크기 때문에 ~



Discounted (future) reward

- 0 보다 크고 1 보다 작은 임의의 감마 값을 설정하여 나중에 받게 될 Q 값에 곱해주는 방법
- 멀리 돌아가는 길일수록 reward 의 값이 작아지고 최적의 길로 갔을 때 reward 의 값이 가장 크다.