HW #1

1970030 박근아

먼저 손계산으로 가치반복 알고리즘을 수행할 수 있다.

우선 t=0 인 경우 모든 상태를 0으로 초기화해준다.

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

t=0

S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

그리고 t=1 일 때 우선 목표지점에 인접한 15번 칸에서 알고리즘을 적용해본다.

(각 행동은 앞글자를 따서 L, D, R, U 으로 표현한다, stochastic $P=1$)

$$v^{(1)}(15) = \max\{P(11,0\,|\,15,U)(0+v^{(0)}(11)),\ P(16,1\,|\,15,R)(1+v^{(0)}(16)),\ P(14,0\,|\,15,L)(0+v^{(0)}(14)),\ P(15,0\,|\,15,D)(0+v^{(0)}(15))\}$$

$$= \max\{0,1,0,0\} = 1$$

$\therefore A(15) = \{Right\}$

H에서는 이동이 일어날수 없으므로 제외하고 각 칸에서 알고리즘을 적용해보면 다음과 같다.

$v^{(1)}(1)=0,\ v^{(1)}(2)=0,\ v^{(1)}(3)=0,\ v^{(1)}(4)=0,\ v^{(1)}(5)=0,\ v^{(1)}(7)=0$

$v^{(1)}(9)=0,\ v^{(1)}(10)=0,\ v^{(1)}(11)=0,\ v^{(1)}(13)=0,\ v^{(1)}(14)=0$

S 0	0	0	0
0	H 0	0	H 0
0	0	0	H 0
0	0	1	G 0

t=1

이어서 2번째 반복을 수행한다. 이전 결과에 이어서 대부분의 칸의 결과는 0 이고 15칸과 근처값들만 다시 구해본다.

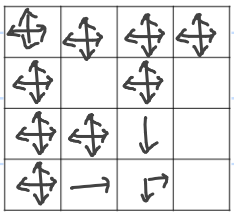
$v^{(2)}(11) = \max\{1\times(0+v^{(1)}(15)), 0, 0, 0\} = 1$, $A(11) = \{Down\}$

$v^{(2)}(14) = \max\{1\times(0+v^{(1)}(15)), 0, 0, 0\} = 1$, $A(14) = \{Right\}$

$v^{(2)}(15) = \max\{0,0,1,1\}$, $A\{Down, Right\}$

↳ 보드 밖으로 나갈 때도 보상은 0 이며 제자리에 머물기 때문에 행동이 추가된다.

S 0	0	0	0
0	H 0	0	H 0
0	0	1	H 0
0	1	1	G 0



t=2

3번째 반복에서도 각 칸에서의 가치함수를 구해준다.

$v^{(3)}(10) = 1$, $A(10) = \{Down, Right\}$

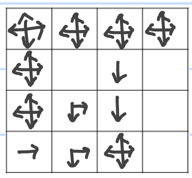
$v^{(3)}(13) = 1$, $A(13) = \{Right\}$

$v^{(3)}(7) = 1$, $A(7) = \{Down\}$

$v^{(3)}(11) = 1$, $A(11) = \{Down\}$ / $v^{(3)}(14) = 1$, $A(14) = \{Down, Right\}$

$v^{(3)}(15) = 1$, $A(15) = \{Down, Right, Left, Up\}$

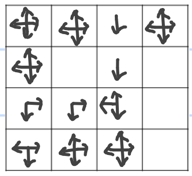
↳ 인근 칸의 보상이 겹에 따라 행동이 추가된다.



S 0	0	0	0
0	0 H	1	0 H
0	1	1	0 H
1	1	1	0 G

t=3

4번째 반복에서도 각 칸에서의 가치함수를 구해준다.



$v^{(4)}(10) = 1$, $A(10) = \{Down, Right\}$

$v^{(4)}(13) = 1$, $A(13) = \{Right, Down, Left\}$

$v^{(4)}(7) = 1$, $A(7) = \{Down\}$

$v^{(4)}(11) = 1$, $A(11) = \{Down, Left, Up\}$

$v^{(4)}(14) = 1$, $A(14) = \{Down, Right, Left, Up\}$

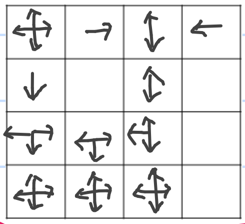
$v^{(4)}(15) = 1$, $A(15) = \{Down, Right, Left, Up\}$

$v^{(4)}(3) = 1$, $A(3) = \{Down\}$ / $v^{(4)}(9) = 1$, $A(9) = \{Down, Right\}$

S 0	0	1	0
0	0 H	1	0 H
1	1	1	0 H
1	1	1	0 G

$t=4$

마지막으로 반복을 진행하면 다음과 같다.



$v^{(5)}(2)=1$, $A(2)=\{Right\}$ / $v^{(5)}(4)=1$, $A(4)=\{Left\}$

$v^{(5)}(5)=1$, $A(5)=\{Down\}$

$t=4$와 비교해 행동의 종류가 달라지는 경우는 또한 다음과 같다.

$v^{(5)}(10)=1$, $A(10)=\{Down, Right, Left\}$ / $v^{(5)}(3)=1$, $A(3)=\{Down, Up\}$

$v^{(5)}(13)=1$, $A(13)=\{Down, Right, Left, Up\}$ / $v^{(5)}(7)=1$, $A(7)=\{Down, Up\}$

$v^{(5)}(9)=1$, $A(9)=\{Down, Right, Left\}$

0	1	1	1
1	H	1	H
1	1	1	H
1	1	1	0

$t=5$

이에 대해 코드를 실행한 결과는 다음과 같고 결과를 표현한 그림은 오른쪽과 같다.

```
In [14]: optimal_policy = extract_policy(optimal_value_function, gamma=1.0)
1.0
2.0
0.0
3.0
0.0
0.0
0.0
0.0
3.0
1.0
0.0
0.0
0.0
2.0
1.0
0.0

print(optimal_policy)

In [15]: print(optimal_policy)
[1. 2. 0. 3. 0. 0. 0. 0. 3. 1. 0. 0. 0. 2. 1. 0.]
```

↓	→	←	↑
←	←	←	←
↑	↓	←	←
←	→	↓	←

손계산한 결과와 비교하면 반복횟수를 고정했을 때, 손계산한 결과는 goal에 도착하는 모습을 확인할 수 있고 그 경로 역시 코드에 비해 조금 더 직관적으로 확인할 수 있다.