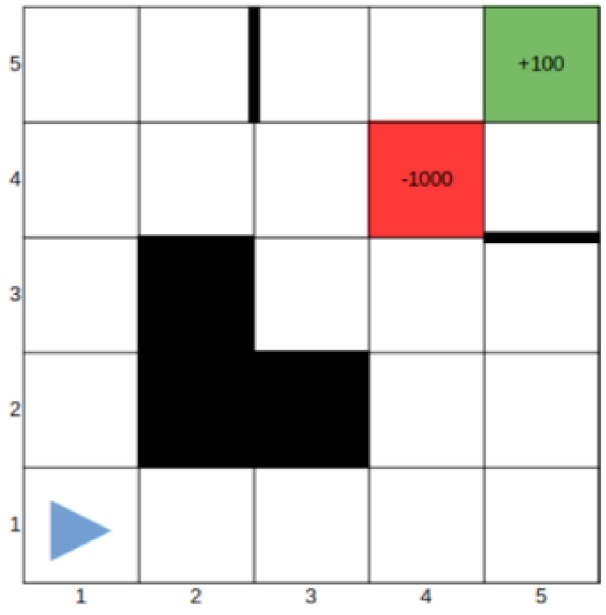
CPSC 4420/6420: ARTIFICIAL INTELLIGENCE

ASSIGNMENT 2

NAME: Sreekar Malladi

Consider the following puzzle. The green and red states are both terminal states, with the rewards as shown (so we can consider the green state the "goal", and the red a "game over" state with a large negative reward). Thick borders between cells represent walls that the robot player cannot cross, and the black squares contain obstacles and cannot be entered. The robot player is represented by the blue triangle, and the direction the triangle points is the way the robot is facing.



The robot can take the following actions:

- A_1: Move one cell forward in the direction it is facing. Cost: 1.5
- A_2: Move two cells forward in the direction it is facing. Cost: 2
- A_3: Turn to its left, and stay in the same cell. Cost: 0.5
- A_4: Turn to its right, and stay in the same cell. Cost: 0.5

Note that each action has a cost. This can also be considered an immediate negative reward (so $R(s,A_1,s') = -1.5$). The cost is evaluated on the current state, the one the robot is in when it begins the action, not the one it ends up in after the action. The same way is the value of state V(s) shows the value of current state and you should initialize the algorithm with $V_1(5,5,x)=+100$, $V_1(4,4,x)=-1000$ (for x=1,2,3,4), and zero for all other states.

Let's represent a state with (x,y,d), where x and y represent horizontal and vertical position (i.e. location), and d represents the direction the robot is facing (1: up, 2: down, 3: left, and 4: right).

So, for example, if the robot is in state (4,1,4), it means that it is in location (4,1) and facing right. The result of possible actions for this state are as follows:

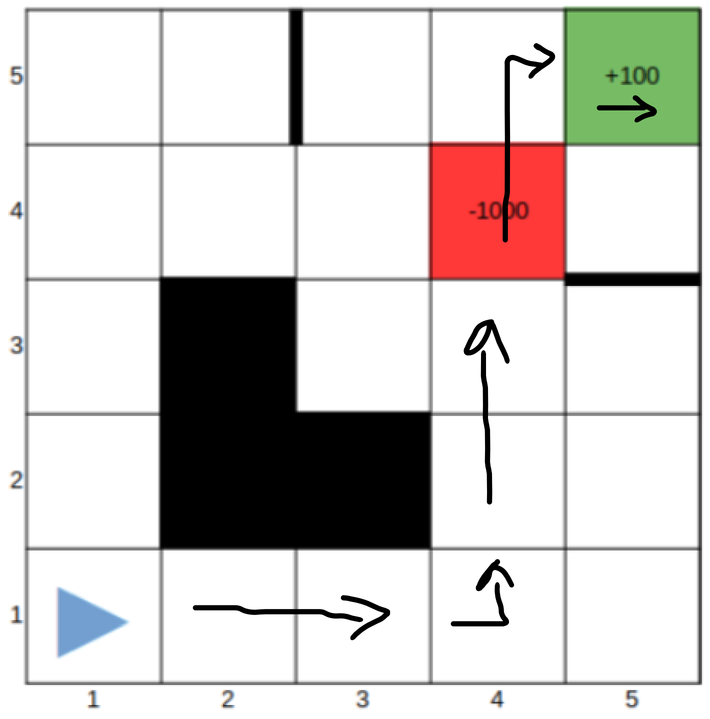
A_1 (move 1 cell forward) --> (5,1,4)
A_2 (move 2 cells forward) --> impossible
A_3 (turn left) --> (4,1,1) : the robot stays in (4,1) but now faces up
A_4 (turn right) --> (4,1,2) : the robot stays in (4,1) but now faces down

A move is impossible if it would result in landing on blocked cells (2,2), (2,3), or (3,2), or if it would result in crossing a barrier, like moving from state (2,5) to (3,5), or (5,3) to (5,4). A move that would take the robot outside of our 5x5 grid is also impossible.

Note that we have more states than there are cells, because the robot facing a different direction produces a new state, even if it does not change location. In the example above, if we move to (4,1,1), where the robot is facing up, this is a different state from the one we were in, (4,1,4), even though the robot has not moved cells.

A) If there is no living reward, no noise, and no discount (gamma = 1), use your common sense to find the best possible route from (1,1) to (5,5).

A. If there is no living reward, no noise, and no noise, the best possible route from (1,1) to (5,5) would be:



Path: (1,1,4) -> (3,1,4) -> (4,1,4) ->(4,1,1)->(4,3,1)->(4,5,1)->(4,5,4)->(5,5,4)

Cost: -2 -1.5 -0.5-2 – 2 -0.5 -1.5 +100 = 90

B) With no discount (gamma = 1), no living reward, and no noise, use the Value Iteration Algorithm with 100 iterations to update the optimal values for each state and print the result [only for the first 10 iterations] in the following format:

```
iter 1:
state (1,1,1)  V = (some value)     Best Action: Ai (where i is some number 1-4)
state (1,1,2)  V = (some value)     Best Action: Aj
...
state (5,5,4)  V = (some value)

iter 2:
state (1,1,1)  V = (some value)     Best Action: Ai (where i is some number 1-4)
state (1,1,2)  V = (some value)     Best Action: Aj
...
state (5,5,4)  V = (some value)
```

If two actions are tied for best, you can select one at random or always choose the one with the smallest index.

B. Code Output:

```
Generating path from initial state to the goal

1 1 4 1

2 1 4 2

4 1 4 3

4 1 1 2

4 3 1 2

4 5 1 4

4 5 4 1

5 5 4 3

Path generated
```

C) If you start from state (1,1,4) and follow the optimal policy you found in part B, does it follow the same path you proposed in part A?

C. Yes, the code follows the same path. (More or less)

Only difference is that the code makes single step moves first instead of two step moves.

D) Repeat part B with the same assumptions, except for gamma = 0.8. Compare the results with that from part B. Do they match?

D. Code Output:

```
Generating path from initial state to the goal
1 1 4 1
2 1 4 2
4 1 4 3
4 1 1 2
4 3 1 2
4 5 1 4
4 5 4 1
5 5 4 3
Path generated
```

Even though the values for each cell change, the path itself does not change.

The actions that lead to rewards have way higher value than those that do not. (As compared to part B)
Basically, the disparity between the values of high reward moves and low reward moves is larger.

E) Repeat part B with the same assumptions, except for gamma = 0.2. Compare the results with that from parts B and D. Do they match?

E. The values for most states are approximately -0.62499968. Some states have positive value. Other states lead to rewards.
The path planner does not converge at all for the value of gamma = 0.2
The agent is stuck in cell (1,1) and it keeps rotating indefinitely.

F) **(Optional for 4420)** Repeat part B, but this time with noise = 0.2, and gamma = 0.9 and no living reward. With a noise of 0.2, every time you take an action, the result will be the expected action with probability 0.8 (80%), but 20% of the time, the robot will instead take a different action (taken randomly out of possible unexpected actions, with equal probability).

For example, if we are in state (4,1,4), so location (4,1) and facing right, and we take action A_1 (moving one cell forward), the result will be:

A_1: (5,1,2)	with probability 0.8
A_2 is impossible	
A_3: (4,1,1)	with probability 0.1
A_4: (4,1,2)	with probability 0.1

Compare the results with that of the previous parts and explain your observations.

F. Code Output:

```
Generating path from initial state to the goal
```

```
1 1 4 2

3 1 4 2

5 1 4 3

5 1 1 2

5 3 1 3

5 3 3 2

3 3 3 4

3 3 1 2

3 5 1 4

3 5 4 2

5 5 4 3
```

Path generated

The path is the same not the same as the previous path.

I believe that due to the addition of probability, the values are inflated, and the agent has a bit of uncertainty added to its path.

P.S. I'm a beginner in python. As such, any tips/pointers/approaches that can help optimize my code are much appreciated!

The values of the initial 10 iterations are printed by the terminal/console when the question1.py code is run.