Intro to AI – Spring 2019

Homework 3

Due: Monday, April 29th, 11:59 pm

Description

In this programming homework, you will write a program that uses Q-learning algorithm to determine the best path to a goal state.

Format

Similar to the format described in the class, you will work with a 3*4 board. Each of the 12 squares has a unique index, as shown on the corner left of the two example boards in Figure 1. There are four special squares on the board. These four squares are: start, goal, forbidden, and wall squares. The remaining 8 squares are empty ordinary squares. The starting square (shown with the letter S) is fixed and always at square 1. The location of the goal, forbidden, and wall squares are determined from the input. An agent has four possible actions of going to the north, east, south and west. The board is bounded from the sides.

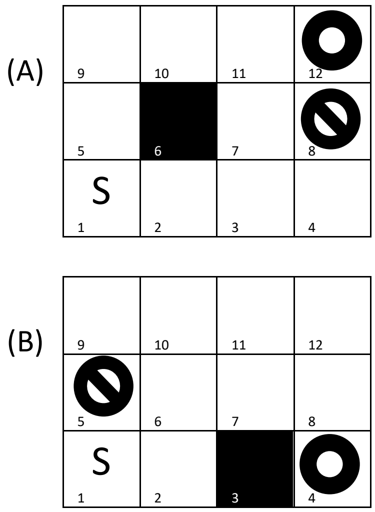

Figure 1 – Two of the possible formats for the board. In (A) the input starts with "12 8 6" and in (B) starts with "4 5 3".

Input

The input to your program consists of three numbers, one character, and possibly an additional number [# # # X (#)]. The first three numbers show the location of the goal, forbidden and wall squares respectively. Figure 1 shows two possible inputs and the corresponding board configuration based on each of those inputs. The remaining items in the input, determine the output format. The fourth item is either character "p" or "q", and if it's "q", there will be an additional number at the end. Item "p" refers to printing the optimal policy (Π*), and "q" refers to the optimal Q-values (Q*).

Implementation

You should use Q-learning to calculate the best action for each square to reach to the goal square. At the beginning, all of the Q-values are set to zero. The (hypothetical) agent should start from the S square. The agent iteratively updates the Q-values for each state, by following the main formula discussed in the class:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[R(s, a, s') + \gamma \max_{a'} Q(s', a')]$$

In this problem, the living reward for every action (each step) is r=-0.1. The discount rate is γ = 0.5, and learning rate α = 0.1. The reward for performing the exit action (the only available action) in the goal square is +100, and for the forbidden square, is -100. You don't need to assign and calculate Q-values for the goal, forbidden and wall states. The agent cannot enter or pass through the wall square. For the purpose of exploring the board, use an ε-greedy method with ε = 0.1. This means that with the probability ε, the agent acts randomly, and with the probability 1-ε, it acts on current policy.

Convergence

If needed, you can consider two-digit precision (#.##) for checking for conversion (i.e. for checking if any Q-value changes). You can also set a maximum number of iterations to 10,000. After conversion, you can set $\varepsilon = 0$.

Output

If the input contains "p", your program has to print the best action that should be chosen for each square or in other words print Π*. To do this, in separate lines print: each state's index and the action. Here is an example:

Input:

```
12 8 6 p
```

Output (possible):

```
1 ↑
2 →
3 ↑
4 ←
5 ↑
7 ↑
9 →
10 →
11 →
```

If the input contains "q" following a number n, the program has to print the four Q-values associated with each of the four possible actions in the state that has index n. Here is an example:

Input:

```
12 8 6 q 11
```

Output (possible):

```
↑ 0
→ 99.23
← 90.45
↓ 78.39
```

Programming Language

As previously discussed you can use any language that the TA approves. Name the submitted file “hw3”, so that a program in Python can be run as follows:

```
$ python hw3.py <input>
```

User Interface

You can use a visual user interface (like textbox and buttons) for input and output. You can (and are encouraged to) show the board and the agent’s movement and learning process. Make sure it’s clear how the program operates and how the output can be interpreted. The steps taken to reach the goal have to be clear from the output. Show Π and Q-values clearly.

Grading

Total points will be 100. You will receive 30 points for printing correct output (10 for Π, and 20 for Q-values). The remaining 70 points will be given to your implementation.

Final Note

Note that this homework might take longer than the previous programming homework for you to finish. The late sublimation policy will be strictly followed.