CS1538 Homework #1[1]

Due by 11:59pm 2/1/2017

Problem Overview

You plan to send a robot to gather some materials (e.g., special plants) in an NxM area. The materials are more likely to appear in certain locations than others, and after being harvested, they will be replenished with certain probability. Your robot will have enough time to take T steps while moving in the environment (for this assignment, assume that T=25%*NM). You want to collect as many units of the materials as possible. You have a map of the area and you know the relevant probabilities for the occurrences of the materials, but you do not know whether the material is actually at a location until your robot visits there.

You are debating between different robot navigation strategies:

1) move toward the location where the material might appear (and reappear) with the highest probability and remain there for the rest of the time.
2) randomly choose a direction and continue until the robot bumps into a wall/obstacle, then randomly choose a new direction to explore.
3) randomly choose a direction at every step
4) always head toward a (randomly chosen) location where the material could appear.
5) **optional** (do not do this until you have fully completed the required portion of the assignment): some navigational strategies of your own design.

Use simulations to:

- determine whether one of the strategies is better than the others for the given map
- explore how changing environmental conditions might change your choice of robot navigation strategies (i.e., suppose the materials are more or less plentiful, suppose the robot can stay in the environment for a longer or shorter period, etc.).

Specifications for Simulation Model

The environment: The map is specified in an input text file in CSV format (N fields and M lines). Below is a toy example of a 5x5 area.

#,#,#,#,#
#,0,0,0,#
#,0,0.9,0,#
#,0,0,0,#
#,#,#,#,#

[1] This file and other support files will also be available in a Google Drive directory (after the initial commit to the repository on GitHub, small typos will be corrected here and any late-breaking additional files will be added here).

- The symbol # represents a wall or an obstacle that the robot cannot walk through.
- The value 0 represents a barren location (no material will ever grow here).
- Any value between 0 and 1 -- let's call this value p:
    - In the beginning of the simulation, this location has probability p chance of generating the material.
    - If the material did not get generated, it still has probability p chance of getting generated at the next time step.
    - If the material exists at time t, unless the robot harvested it at that time, it should still be there at time t+1.
    - If the material has just been harvested, that location has to "rest" for a period of round(p^2*100) steps before it can start to generate (with probability p) again. In other words, if a location can generate material with a probability of 0.3, when that location is harvested, it has to wait for 9 time-steps before it can start generating with probability 0.3 again.

In the assignment repository, there are two sample maps:

- toymap -- this is a smaller map that might be good for debugging etc.
- hw1map -- this is the main map to use for the experiment.

The Robot: Your robot always starts at location (1,1) -- i.e., the lower left hand corner; at each time step, it can move to one of the (non-wall/obstacle) neighboring squares -- a possible of at most 8 directions. If the robot moves into a square with some material, the material is automatically harvested.

Your simulation program should take two input arguments: the map, and a robot navigation strategy. It needs to create a simulated environment based on the map, following the material generation rules described above. Then add a robot to the environment and simulate its movement according to the specified navigation strategy. It should keep track of how many units of the materials the robot collected and stop the simulation when the time-step limit is reached.

You may need to write additional wrapper programs for running the experiment.

What to commit

- All of your source codes.
- A README file that contains the following information:
    - How to run your program (mention interpreter/compiler version if necessary)
    - List any additional resources, references, or web pages you've consulted.
    - List any person with whom you've discussed the assignment and describe the nature of your discussions.
    - (final commit) Describe any component of your program that is not working
- (final commit) A report that discusses the following:

- Overview -- Briefly summarize the goal of the experiment. Without running the experiment, which strategy do you expect to do better? Why?
- Experimental design -- assuming that your simulation program is working correctly, how would you use it to answer the questions asked in the overview section? Of the two questions asked, the second question is somewhat underspecified. Describe what kinds of environment conditions you plan on changing, and give some brief justifications for your choices.
- Analysis of the experimental results -- report the outcomes. Do you trust the numbers you got? How do the results compare with your initial guesses?
- Further discussions -- Discuss any trade-off between different robot navigation strategies (e.g., consider the ease of programming each strategy, the memory footprint of the robot, extra run-time needed to compute what to do next, etc.). Having performed the simulations, propose some alternative ideas for the robot navigation strategy. **Optional:** implement it and rerun the experiment to see how well it worked, then discuss the additional outcomes.

Grading Guideline

Assignments are graded qualitatively on a non-linear five point scale. Below is a rough guideline:

- 5 (100%): The program works as expected; there is a clear enough README for the grader; the report is complete and thorough -- insightful observations and correct analysis.
- 4 (93%): The program works as expected, but possibly with some minor flaws; there is a clear enough README for the grader; the report is complete with correct analysis.
- 3 (80%): The program works as expected, but possibly with some minor flaws; there is a clear enough README for the grader; the report is complete but cursory, or there are some minor inaccuracies in the report.
- 2 (60%): The program has major problems; there is a clear enough README for the grader; OR: the program is OK but the report is missing/incomplete/incorrect.
- 1 (40%): A serious attempt has been made on the assignment.

Bonus -- if you submitted an optional navigation strategy and discussed its outcome, you may receive up to 7% bonus points, depending on the quality of the strategy and the extra write-up.