Project 1
Computer Science 4013: Artificial Intelligence
Due 12:01AM Feb 12, 2011 (Note: This is Saturday!)

Introduction

The goal of this project is to have you learn how to enable an agent to move about a world in an intelligent manner. If you are a CS 4013 student, you will implement depth first search. If you are a CS 5013 student, you will implement depth first search and breadth first search. Although neither of these approaches will give you an optimal agent, that will come soon and these approaches will be very useful for the next project.

Pacman's goal is to eat as many pellets as possible while avoiding the ghosts. Pacman should also try to each the extras that appear (like the cherries). We will be grading you on a set of boards that you will have never seen so be sure your searches work on any board layout!

In order to quickly verify that your search techniques are working, you will need to order your actions. At any given state, your pacman agent should search actions in the order: UP, DOWN, RIGHT, LEFT.

Extra-credit opportunities

To keep grades within a fair range, all extra credit will be capped at 10 points. This means that no project can receive a grade higher than 110, no matter if they win the ladder, find great exploits, and are very creative.

Wanted dead or alive: bugs or exploits in the simulator

We are reasonably certain that you cannot exploit the simulator (say by directly affecting your points scored). However, any project has bugs and we want to know about them! If you find a bug or an exploit, you can receive extra credit according to the following scale:

- **5 points:** If you find an exploit and report it to us you can receive 5 points extra credit upon verification of the report. Note, we know of two exploits for which you cannot get extra credit as they are already discovered. Since both are both extremely difficult to fix and extremely difficult to implement, I'm not listing them here.

- **10 points:** If you find an exploit and give us a fix for it, you can receive 10 points extra credit upon verification of both the exploit and the fix.
- **1-3 points:** General simulator bugs are much more likely than exploits. Finding a bug and reporting it can get you 1 point. Fixing the bug and giving us the fix (you can't check it in directly but you can give it to us in the bug report) can get you three points. Both bug and fix must be verified for any extra credit to be awarded.

Competition ladder

This project has a class-wide competition ladder. Because the project is short (only one week), the ladder starts immediately and will run each night until the project is due. The first place player will receive one extra point for each night that that player wins the ladder up to a maximum of 5 points. The second place player will receive 1/2 extra point for each night that the player is in second place. No player can receive more than 5 extra points from the ladder and points will be distributed down the ladder accordingly should the maximum be reached.

The ladder will be scored by pacman's score which is **fill me in**. To receive extra credit, you must rank above the Random agent.

Wanted (alive please): creative individuals

Creativity is highly encouraged! To make this real, there are up to 10 points of extra-credit available for creative solutions. Some ideas here include good heuristics for dealing with ghosts, real-time planning where you plan in the background while executing your current best plan, or other forms of intelligent search. To get ideas on this, you may find the website `www.gamasutra.com` useful. If you choose to implement anything that you consider creative, please do the following:

- Document it in your writeup! I can't give extra credit unless I know you did something extra.
- Your search must still have DFS/BFS (depending on if you are 4013 or 5013) at the heart of it. If you are unsure if your search qualifies, come talk to me. Also, don't implement A^*! That will be the next project!
- Remember that by being creative I am referring to the algorithm and *not* to the ability to creatively download code. All project code must be written exclusively by you except for the sample players that we provide.

Implementation details

All of your source code must reside in your src/4x4 directory and be in your 4x4 package. You may name your files within this package anything that makes sense to you (remember that we are grading on coding style as well). To make the simulation know about your agent, you also need a `pacmaninit.xml` file. There is a sample pacmaninit.xml file in the examples.hello subdirectory. You will need to change this file to point to your new agent at name1234.MyAgent (or whatever you call your agent class). **Do not name it something other than pacmaninit.xml.** Also, you will need to update the entry for displayName to be something other than the default. Include your name in it for ease of grading.

The agent class contains a startAction(), endAction(), and initialize() method by default. startAction() is called each time an agent is about to begin an action and it must return a valid action for the agent to execute. endAction() is called after all agents have ended their actions but before the simulator goes to the next timestep. This may be left empty if you have no need for cleaning up after an action. initialize() is called when an agent is created (but not when it comes back to life from being killed). The example heuristic agent shows you how to access the internal state of the agent and of the environment.

Using methods from the Java SDK is acceptable and encouraged but downloading or using code from any other sources is not allowed. See the syllabus for more details on what is considered academic misconduct. As discussed below, any additional files you create should be turned in along with your main agent class.

Competing heuristics

For this project, you may play against the following heuristics:

- **Random** makes completely random moves. Not very bright but you must beat it for extra-credit on the ladder!
- **ClosePelletGrabber** is our naive pellet collecting agent. It moves to the nearest pellet and moves randomly if there are no pellets around it. While not terribly intelligent, it should show you how to access the state of the environment and how to make a more intelligent agent than random!

Project 1 details

For this assignment, you need to do the following:

1. Update your aiprojects code from project 0. If you got subclipse working, you can update your code from within eclipse using Team → Update. If not, use the command line or tortoiseSvn to do "svn update". If you did not get the code checked out for project 0, follow the instructions to check out the code in that writeup. The code has been updated so you will need to do this to complete project 1! Also be sure to refresh eclipse (right click on the project and select refresh). This will update the listing of files.

2. Change the worldconfig.xml file in examples.project1 to point to your agent in src/4x4. The detailed instructions for this are in project 0. Make sure to copy over a pacmaninit.xml in the src/4x4 directory so your agent knows how to start. Unlike project 0, you must name your xml file pacmaninit.xml for this project. To run your agent, use ant to run the target project1. Be sure to update the displayName entry to something other than the default! You must include your name it in somehow for ease of grading but you can make it funny/witty beyond that.

3. Create a valid board template for pacman. This will be turned in using submit (described below). The instructions for creating templates are in the docs directory.

4. Create an agent that makes moves using search as described below. Build and test your code using the ant compilation system within eclipse or using ant on the command line if you are not using eclipse (we highly recommend eclipse or another IDE!). To facilitate grading, you can add logging statements (but turn them off so the ladder isn't bogged down!) showing the process of your search.

   (a) CS 4013 students: You must create an agent that moves pacman using depth first search. Moves should always be investigated in the order: UP, DOWN, RIGHT, LEFT.

   (b) CS 5013 students: You must create an agent that can move using depth first search and breadth first search. To facilitate grading, you will turn in your DFS agent to Project1 DFS and your BFS agent to Project1BFS (detailed below).

5. Test your player(s) on a sample ladder. This will ensure that the agent runs on the CSN linux machines and that you have the agent correctly configured. To do this, we have created a submission script that will take your agent as input and run it against the heuristic agents several times and output the information to your terminal window. WARNING: It will output a LOT of text or either be ready to scroll or put the output to a file. To submit to this agent, do the following:

```
/opt/ai4013/bin/submit CS4013 Project1TestLadder *.java pacmaninit.xml
```

where you can simply list the java files instead of using *.java if you choose. Note that the xml file MUST be named pacmaninit.xml!

6. Submit your project on codd.cs.ou.edu using the submit script as described below.

   (a) Log into codd.cs.ou.edu using the account that was created for you for this class. Your username is your 4x4 and your default password is cs#4x4. Remember to change your password! Use ssh to login to codd.

   (b) Make sure your working directory on codd.cs.ou.edu contains all the files you want to turn in. All files should live in your 4x4 package. For example, if all of your code lives in `MyDFSPacmanAgent.java`, you would submit your code using the following command. The pacmaninit.xml file is required to run your client! Also be sure to turn in your board template.

   ```
   /opt/ai4013/bin/submit CS4013 Project1DFS \
   MyDFSPacmanAgent.java MyCoolBoard pacmaninit.xml
   ```

   If you have extra code in `Graph.java`, you would submit using the following command:

   ```
   /opt/ai4013/bin/submit CS4013 Project1DFS *.java MyCoolBoard pacmaninit.xml
   ```

   (c) Note that both CS 4013 AND CS 5013 students should submit to Project1DFS. CS 5013 students will ALSO need to submit their BFS agent to Project1BFS. For example, the following command will submit my BFS agent to this project.

   ```
   /opt/ai4013/bin/submit CS4013 Project1BFS \
   MyBFSPacmanAgent.java MyCoolBoard pacmaninit.xml
   ```

   (d) After the project deadline, the above command will not accept submissions. If you want to turn in your project late, use:

   ```
   /opt/ai4013/bin/submit CS4013 Project1DFSLate *.java pacmaninit.xml
   ```

   ```
   /opt/ai4013/bin/submit CS4013 Project1BFSLate *.java pacmaninit.xml
   ```

At 12:02AM on Feb 1, I will run a sample ladder. This ladder will not count for extra-credit points as the focus of this part of the assignment is to make sure that your accounts are setup correctly and that the ladder is able to run each of your players.

Point distribution

- 70 points for correctly implementing BFS/DFS. CS 4013 students will receive this credit for DFS and CS 5013 students will receive 35 points per search method (with the following partial credit examples also scaled in 1/2 for each method). A correct pacman agent will clear the board while avoiding obstacles.
  - 65 points if there is only one minor mistake. An example of a minor mistake would be having off-by one errors (where you miss a search node).
  - 55 points if there are several minor mistakes.
  - 50 points if you have one major mistake. An example of a major mistake would be failing to correctly mark nodes as visited so the search might infinite loop, negative path costs, or ignoring the ghosts.
  - 40 if there are several major mistakes.
  - 30 points if you implement a search other than BFS/DFS that at least moves the agent around the environment in an intelligent manner.
  - 20 points for an agent that at least does something other than random movements.
- 10 points for your board design: if you turn in a valid board (one that loads), you will receive full credit. You can test your board by editing the pacmanconfig.xml file to point to your new board.
  - 5 points if your template is almost valid and we can fix it very quickly
- 10 points: We will randomly choose from one of the following good coding practices to grade for these 10 points. Note that this will be included on every project. Are your files well commented? Are your variable names descriptive (or are they all i, j, and k)? Do you make good use of classes and methods or is the entire project in one big flat file? This will be graded as follows:
  - 10 points for well commented code, descriptive variables names or making good use of classes and methods
  - 5 points if you have partially commented code, semi-descriptive variable names, or partial use of classes and methods
  - 0 points if you have no comments in your code, variables are obscurely named, or all your code is in a single flat method.

- 10 points for your writeup. A full-credit writeup will describe your implementation of BFS/DFS. 5 of the 10 points will be given out for correctly describing the environment using the terms from chapter 2. We already said it is several of them but list the other environmental characteristics.
- As with the previous project, we will deduct 5 points from your total score if your password has not changed from the default (cs#4x4) password.