Middle East Technical University Department of Computer Engineering

CENG 242

Programming Language Concepts

Spring '2013-2014

Hw3

Due date: 14 April 2014, Monday, 23:55

1 Introduction

Last night you had a dream and in that dream you were the head of a big game company. The dream really had influence on you and this morning you called your best friend Ricky and told him that you want to found a company which produces games. Furthermore you already come up with the idea of your first game. The game is a basic grid game (a good idea to start game programming). In this game you will have a 15 x 15 grid and in this grid you will have agents. You will have two type of agents; hunters and preys. Normally prey agents will be controlled by users and hunter agents will be automatically controlled by the computer. However in this base version both type of the agents will be controlled by the computer. Both of the agents will have their own strategy and you want to see what happens if both type of the agents use these strategies.

2 Specifications

- The environment to be simulated is a 15 x 15 grid world and it has several obstacles located at grid cells that a number of mobile agents should avoid.
- Two types of mobile agents, namely hunters and preys, exist in the environment. Any grid cell can have at most one of the obstacle, prey or hunter objects.
- There are five possible actions for hunters and preys: stay still, move one grid cell in one of the 4 different directions (left, right, up or down). Agents move concurrently. This moves are numbered from 0 to 4, namely stay still (move 0), left (move 1), right (move 2), up (move 3), down (move 4).
- Prey agent's goal is to escape from hunters. Actually more efficient algorithms can easily be developed but for this version you will code a simple artificial intelligence for prey agents. At each time step t they (preys) will move according to the following formula: (fibonacci(t)) modulo 5. Meaning that they will calculate the t'th fibonacci number and take the modulo 5 of it. The resulting number will be between 0 and 4. They will (all of the prey agents) move according to this number at that time step. For example if we are at 5th time step (time steps are starting from 1) and the 5th fibonacci number is 5 then the modulo 5 of it is 0. In this case the prey agents will stand still (because move 0 is defined as stand still) in that time step. Furthermore if the chosen cell has an obstacle in it then the prey will stand still. Another thing to pay attention is, if the choosen action by preys lead that particular prey to out of the grid then that prey should stand still at that time step.

- Hunter agent's goal is to capture the prey agents. They capture the agent when they are in the same cell with a prey agent. Their strategy is again simple. It is step by step defined below:

  1. Each of the hunter calculates it's own distance (with manhattan distance) to the all of the preys and chooses the prey with the minimum distance to itself. If there are more than one preys with the same distance then the one with the lowest row number is choosen. If also the row numbers are equal then the prey with the lowest column number is choosen.
  2. From all of the neighboring cells of the hunter (1 right,1 left,1 up and 1 down cells) manhattan distances to the choosen prey is calculated. (You don't have to calculate manhattan distances from the neighbor cells that have obstacle in them. Furthermore you don't have to calculate manhattan distances from the neighbor cells that are out of the grid. Because hunter agents will not choose these cells ever.)
  3. After manhattan distances from the neighbors are calculated the hunter should choose the cell with the minimum distance to the prey. If there are more than one cell with the same distance then the one with the lowest row number is choosen. If also the row numbers are equal then the cell with the lowest column number is choosen.
  4. If there is no conflicts with the other agents then the hunter will move to the choosen cell. (The conflict situation of the agent's is explained below.)

- Before making the move for both hunters and preys make sure that 2 agents (two preys or two hunters) are not moving into the same cell. If such situation is seeming to happen at that time step, move the agent with the lowest row number to that place and make the others (which are trying to move into the same cell) stand still. If the row numbers are also equal then move the agent with the smallest column number and make the others stand still.

- Hunter agents have initial energy levels, given with e. Energy level of a hunter is decremented by 1 for any movement action (left, right, up or down), by 0.2 for stay still. If a hunter captures a prey then the energy of that hunter will be increased by 3.

- The captured preys and the hunters which finish their energy won't exist in the game grid anymore. You should delete them from your game grid.

- The game finishes in two situations. The first one is the case that all of the prey agents are captured. The second situation is the case that the energy of all of the hunters are finished. In these cases the game finishes and you should print the final game table (this will be explained below).

- No wrong inputs will be given.

3 Functions to be implemented

In this homework you will implement a module named hw3. In this module you should export two of your functions (to provide abstraction). That function names are `print_game_table` and `simulate_the_game`.

- `print_game_table environment_list`

  This function is to print the current status of the game table (or the grid). `environment_list` is a list of triples. At each triple the first element is 'h','p' or 'x'. 'h' represents hunter,

'p' represents prey and 'x' represents obstacle. The second and third numbers are representing the row and colunm numbers. (In our case the grid's row and column numbers are starting from 0). You will print hunters as 'H', preys as 'P' and obstacles as 'X' for all the other cells you should print '-'. For a 4 x 4 grid for the following function call will work as the following:
`print_game_table` [('p',0,0),('h',0,3),('p',1,1),('x',2,0),('h',2,2),('x',3,0)]

```
P - - H
- P - -
X - H -
X - - -
```

- `simulate_the_game environment_list e until_when`

  This function is the main function which will do all the work described at the specifications part. It will print the game table as a result after the simulation is over (game is over). `environment_list` is same as the `print_game` function. 'e' is the initial energy of the hunters it is a Float. `print_when` parameter is an Integer. It shows how much time steps your game should last. If it is given -1 then you should play the game until it is over (according to the rules given in the specifications part) if it is given any other positive number (smaller than the maximum time step that the game requires) then you should not wait until the game finishes you should return after the specified amount of time steps. (Time steps are starting from 1).

4 Manhattan Distance

In this homework when you will calculate distance you will always use Manhattan distance. What is Manhattan Distance then ? This distance calculation is inspired from well-known gridlike street geography of the New York borough of Manhattan. The calculation is pretty easy.
The distance between two points measured along axes at right angles. In a plane with p1 at (x1, y1) and p2 at (x2, y2), the Manhattan distance is |x1 - x2| + |y1 - y2|.
To understand better; assume that we have the following grid and want to calculate the distance between obstacle at (3,0) and prey at (1,1):

```
P - - H
- P - -
X - H -
X - - -
```

The distance is |3-0| + |1-1| which is 3. As another example the distance between the hunter at (0,3) and the prey at (1,1) is |0-1| + |3-1| which is 3.
Note: When calculating Manhattan Distance if there is an obstacle in the way this changes nothing still the distance will be the same. (Calculate the Manhattan Distance without thinking about the obstacles.) Obstacles are only important in this homework when they are at the neighbor cells and that time we have rules for both hunters and preys on what to do in that case.

5 Review

In this part a small example will be traced to make you understand the homework better. For this small example we will use a 4 x 4 grid again. To make you understand where the row and column number starts we put numbers on the grid. (just to make you understand, normally they will not exist)
Assume that the input is the following:

```
simulate_the_game [('p',0,0),('h',0,3),('p',1,1),('x',2,0),('h',2,2),('x',3,0)]
100 1
```

This means that you will evaluate only for one step (the first step) and print the resulting environment. Initially the environment is like the following: (n and the numbers are just given to illustrate where the row and colunm numbers are starts)

```
n 0 1 2 3
0 P - - H
1 - P - -
2 X - H -
3 X - - -
```

Then your code should do the following (The order of these steps may change according to your implementation but the outcomes are unique):

1. For the prey's move's we are at first step so calculate fibonacci(1). The outcome is 1, so we will employ move 1 which is left. For the prey at (0,0) we can not go to left so we will stand still. For the prey at (1,1) there is no problem so for now we can say that it will move to (1,0).

2. For the hunter at (0,3):

   a. First calculate the manhattan distance to prey at (0,0). The distance is 3.

   b. Calculate the manhattan distance to prey at (1,1). The distance is 3.

   c. The distances from the hunter to both prey's are equal so we chose the one with the lowest row number which is (0,0).

   d. Now we will calculate distances from the avaible neighbor cells of hunter to the prey at (0,0). First we calculated the manhattan distance from cell (0,2) to prey at (0,0). The distance is 2.

   e. Now we calculated the manhattan distance from the cell (1,3) to prey at (0,0). The distance is 4.

   f. From these two cells the hunter chooses (0,2).

3. For the hunter at (2,2):

   a. First calculate the manhattan distance to prey at (1,1). The distance is 2.

   b. Calculate the manhattan distance to prey at (0,0). The distance is 4.

   c. The distances from the hunter to prey at (1,1) is minimal so hunter chooses this prey.

   d. Now we will calculate distances from the avaible neighbor cells of hunter to the prey at (1,1). First we calculated the manhattan distance from cell (1,2) to prey at (1,1). The distance is 1.

   e. Now we calculated the manhattan distance from the cell (2,1) to prey at (1,1). The distance is 1.

   f. Now we calculated the manhattan distance from the cell (2,3) to prey at (1,1). The distance is 3.

   g. Now we calculated the manhattan distance from the cell (3,2) to prey at (1,1). The distance is 3.

   h. The distance from cell (1,2) and (2,1) is equal so we took the one with lowest row number. So, hunter chooses to go (1,2) cell.

4. We check if there are any conflicts (meaning if there are agents trying to move into the same cell) and the answer is no for this case. Also none of the preys are captured.

5. So we can update our game table.

The new game table (the final game table because only one step is wanted) is the following:
```
n 0 1 2 3
0 P - H -
1 P - H -
2 X - - -
3 X - - -
```
Note that if we were to continue both of the hunter's energy should be updated as 99 for the next move.

6 Regulations

1. **Programming Language:** Haskell
2. **Late Submission:** You have 3 days out of 10 (total lates for all homeworks) for the late submission.
3. **Cheating: We have zero tolerance policy for cheating.** People involved in cheating will receive zero from all homework (hence get NA from this course).
4. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.
5. **Evaluation:** Your program will be evaluated automatically using black-box technique so make sure to obey the specifications. Your codes will also be visually investigated and code analysis tool will be used to catch similarities between submissions. For that reason, never share your code with others.

7 Submission

Submission will be done via COW. Create a tar document named hw3.tar.gz which contains a single file named hw3.hs that contains your code.
Note: The submitted archive should not contain any directories! Failing to obey this specification will cause deduction in your grades.