**Lab3: alpha-beta-pruning:**

|  |  |
| --- | --- |
| Question1: Given the following game tree, mark the value for each node using minimax. | C:\Users\yiliu\AppData\Local\Temp\SNAGHTMLd4429f.PNG |
| Question 2: Given the following game tree, mark the value for each node and the branches that have been cut. |  |
| Question 3: Given the graph from pacman maze, list the optimal action that Pacman AI agent will choose   1. North b)West   c) East d) South  Pic1)  Pic2)  Pic3) |  |
| Question 4: Design a utility function for pacman AI agent, write your answer on the right. |  |
| Question 5) Design a heuristic evaluation function for pacman AI agent. List the factors you want to add to the heuristic function (you may your observation from question 3 or add more features to help the agent to run well)  Write you answers on the right |  |
| **Coding an AI agent for Pacman (for your hw).**  File that has been changed:  File added: | **PacmanGameDemoAgent.py: (run this file)**  Line 10: Import PacmanAdversialGameAgent, PacmanAdversarialGameProblem, and so on  Lines 48-52: create ghost, create PacmanAdversarialGameProblem, then create pacmanAdvAgent and ghostAgent  Line 67 and 87, take turn to play  **Ghost.py** |
| File needed to check: You may need the following data for your utility function, terminate\_test, and heuristic eval. function | **PacmanGame.py**  contains data structure for your agent, such as capsulePos, foodPos, pacmanPos, ghostPos |
| File needed to modify: | **AdversarialGameAgent.py**  Design the class PacmanAdversarialGameProblem  Then design the class PacmanAdvGameAgent  The files also contains the following functions:   * **minmanhattanDistance**   + find and return the shortest distance among the list elements to a given position * **randomMove**: the method to move ghost * **betterThanRandomMove**: half of the time ghost will chase the pacman * **alphabeta**\_**search**: alpha-beta-pruning * **minimax**\_**decision**: minimax |
|  |  |

**HW-Chapter 5-1( you can discuss with your groupmember(2 persons in s group at most), due: 10/19):**

1. Finish all question if you have not done in the lab
2. Write a better evaluation function for pacman in the provided function *evalForPacman*

You may use any tools at your disposal for evaluation, including your search code from the last project. With depth 2 search, your evaluation function should clear the smallClassic layout with one betterThanRandomMove ghost more than half the time and still run at a reasonable rate (to get full credit, the score of the game should be averaging less than 160 points when he's winning).

Run your pacman game 10 times, record the scores each time when pacman wins, average scores and the number of wins when you submit your code (lose 3 points without submitting the results). Examples are :

**Scores for ten times: 112, 130, 115, 200, 235, 132, 145, 121, 151**

**Average score: 149**

**The number of wins:9**

Your evaluation function will get the points in the following way:

* If you win at least 4 times without timing out, you receive 4 points. Any agent not satisfying these criteria will receive 0 points.
* +3 for winning at least 5 times, +1 for winning all 10 times
* +1 for an average score of at most 200, +3 for an average score of at most 150 (including scores on lost games)

**Hints and Observations**

* You may want to use the reciprocal of important values (such as distance to food) rather than the values themselves.
* One way you might want to write your evaluation function is to use a linear combination of features. That is, compute values for features about the state that you think are important, and then combine those features by multiplying them by different values and adding the results together. You might decide what to multiply each feature by based on how important you think it is.