PACMAN PROJECT

AUTHORS

- Lê Trung Kiên - ltkien@apcs.vn - <1651021>
- Nguyễn Quốc Việt - nqviet@apcs.vn - <1651069>

INPUT

- The user will provide the maps with N M, position of elements and pacman starting position in a text file named "input.txt"
- User will be able to choose which level to execute the program from a console screen.

OUTPUT

- After the program finish, the user can view a graphical representation of how Pacman moves on the console screen.
- User can also view additional information in a file named "output.txt"
- The level chosen will also be written to a "level.txt" file

TOOIS

- We wrote our main business logic using C++ and output results to files
- For the console driver application we used Python instead. The user will mainly interact with this program.

EXECUTION

- Please refer to the README file to run and test correctly

IDEAS

Level 1

Algorithm

- Since we know everything on the map and the cost of moving is constant, we can use Breadth First Search to figure out the shortest way to reach the food.
- However, if the food is too far away, the cost overweights the profit, we will choose to stand still and not move at all.
- Execution time: below 0.05s

Pros

- Easy implementation
- Deterministic
- Solution is optimal

Cons

- High memory cost

Level 2

Algorithm

- Since the ghosts doesn't move, we treat them like normal walls and use Breadth First Search
- Execution time: below 0.05s

Pros

- Easy implementation
- Deterministic
- Solution is optimal

Cons

- High memory cost

Level 3

Algorithm

- Firstly, random a number between (mxn)/2 to mxn as the maximum steps Pacman will move
- Then we choose a random spot from a set of movable position around Pacman to move.
- However if there are below 2 choice to left, we will allow moving backward to get out of a deadend.
- Execution time: Could take up to 1 minute in a 40x40 board, and up to 10 seconds in a 30x30 board.

Pros

- Easy implementation
- Performs better when foods are gathered nearby.
- Faster then brute-forcing through every possible steps.

Cons

- Stochastic: We don't have any kind of prediction so it really depends on luck to get the highest score.
- May miss out in the case of "gold mines" (many foods next to each other but they are placed extremely far away).
- Run time depends on the randomed number of steps.

Level 4

- Can run but can't be drawn yet

Algorithm

- Pacman will perform BFS in 4 direction to find the closest food and move towards it

- Ghosts will use A star with Manhattan distance to Pacman to chase Pacman
- If Pacman sees a ghost within a Manhattan distance of 2 blocks away, it will not proceed in that direction.

Pros

- Easy implementation.
- Deterministic.
- Ghost chasing Pacman lowers the possible paths Pacman can take, which forces Pacman to travel more and get lower scores.

Cons

- Slow running time
- High memory usage
- Ghosts only chase and doesn't try to block Pacman's path to food