Farm Simulation Task

Daniel Paul Pena

# Instructions

* **The program requires Python3, will not work correctly with Python2**
* Have the three files in the same folder.
* In Run.py configure the size of the farm and the time for each action.
* Run Run.py

# Design decisions

* In each turn, the agent can do any of the possible actions or wait for the next day. This means that i.e. planting 2 cells and then waiting for the next day would be three consecutives actions in three different turns. In this way the agent can perform all the actions that it desires in each day. The total number of actions in one turn (counting invalid actions) will be n\*n\*3 + 1 (for each cell 3 actions + wait for the next day).
* The nodes of the Search Tree don’t contain an instance of the simulator, just the action from the father node that leads to them. For each iteration of the Monte Carlo Tree Search, a copy of the Simulator instance is used and applies all the actions until reach to the selected Leaf Node, then will expand and simulate the run.
* When expanding a node it will consider all the possible actions and create one node for each action, even if that action is not possible, if at any point the program try to reach a node with an invalid action it will delete it and select another node. This is done to avoid precalculating the possible moves during the expansion, which required to create multiple copies of the Simulator, generating an important performance issue.
* When simulating the run the policy is not totally random, it will only select the action of waiting for the next day if there is no any other action available.
* The score for each run is calculated as Total food harvested / number of cells \* 15. This way the score is keep approximately between 0 and 1 (can exceed in some cases) and the value of c = √2 is adequate independently of the size of the farm.

# References

* Reinforcement Learning by Sutton and Barto. Chapter 8
* <https://www.baeldung.com/java-monte-carlo-tree-search>