Sample Programming Assignments

1 Review on Search

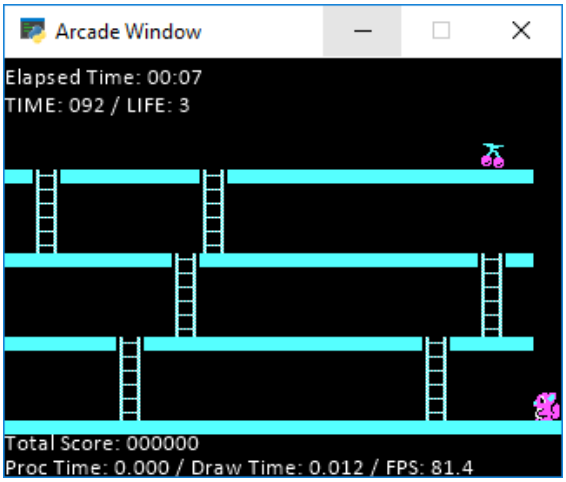
```
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 0],
 [4, 6, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 0],
 [1, 6, 1, 1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],
 [1, 6, 1, 1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],
 [4, 4, 4, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 4, 0],
 [1, 1, 1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 1, 0],
 [1, 1, 1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 1, 0],
 [4, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 4, 4, 4, 0],
 [1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 1, 1, 1, 0],
 [1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 1, 1, 1, 0],
 [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]]
```

1.1 Consider a grid world represented like above, defined as a 2D Python array. The encoding is like following. Assume that we are at $(10, 19)$ now. How can we use a search algorithm, e.g. DFS, to find a path from the initial location to the goal location (indicated as 8)?

- 0: Agent should always move left. This is where game information is printed so agents should not move into this area.
- 1: Agent can move either left or right. However, it cannot move to 0.
- 4: This is the platform cell. Agent cannot exist in this cell.
- 6: This is the ladder cell. Agent can move either up or down. However, it cannot move to 4.

1.2 Implement your DFS algorithm and test with the array above. Print out the found path. You can compare your solution with the provided `sample_dfs_search.py`.

2 Navigation with Search



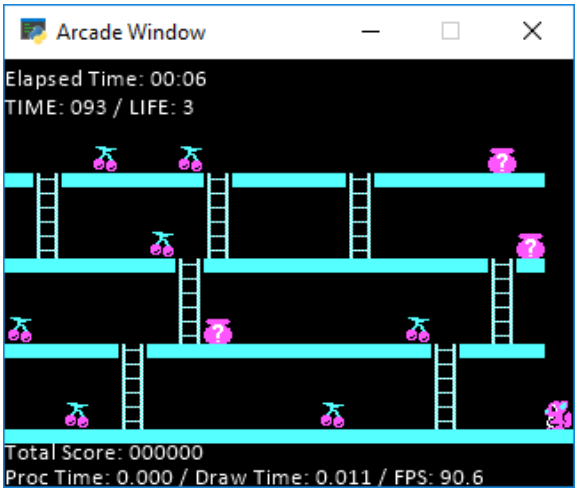
2.1 Assume that your agent needs to navigate a simple environment shown above. Your objective is to guide the agent to reach the goal (fruit). Among the algorithms we have learned, which one can be applied to to this problem?

2.2 We learned several different search algorithms in class. Which algorithm would you choose to solve this problem? Give a reason for your answer.

2.3 When implementing your suggestion above, which data structure/algorithm you would use? Why?

2.4 Implement your navigation algorithm. You may replace `game_data.py` with the provided `game_data_mod1.py` to start with. You may also need to modify the module inclusion code at the top of `game_core.py` and `game_object.py`.

3 Navigation with Planning

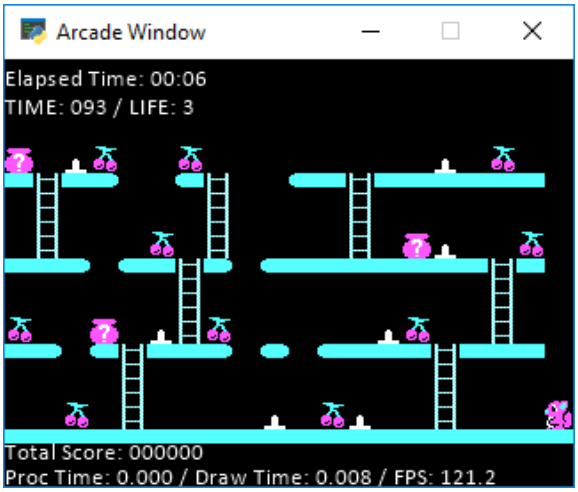


3.1 Now we have multiple targets to eat. Note that fruits always have smallest score and the bags with question marks have higher scores (either 500 or 1000; for the time being, assume that you know which one has the higher score than the other). Your first goal here is to guide your agent to eat all targets in the most efficient way. Discuss how you would approach this problem.

3.2 Implement your navigation algorithm. You may replace `game_data.py` with the provided `game_data_mod2.py` to start with. You may also need to modify the module inclusion code at the top of `game_core.py` and `game_object.py`.

3.3 Assume that your agent is allowed to make only a limited number of actions because of the time limit. What would be the best navigation strategy in such cases? Should the agent eat only the fruits? Should the agent aim for the bonus bags? Or mixture of these? In which criteria the agent should take into account when making decisions, in what way?

4 Obstacles and Adversarial Agents



4.1 Now we have the full game stage except the aversarial agents. What changes should be made from the algorithms/methods you implemented earlier? What additional factors we should consider in making those changes?

4.2 Implement your navigation algorithm. You may replace `game_data.py` with the provided `game_data_mod3.py` to start with. You may also need to modify the module inclusion code at the top of `game_core.py` and `game_object.py`.

4.3 Discuss how you would extend the implementation above so that it can deal with adversarial agents in the wild.