Project 1: Knowledge Retrieval

Part 3: Implement Monte Carlo Tree Search (MCTS) Algorithm

For this part, you need to implement Monte Carlo Tree Search algorithm so that it can extract a *task tree* to prepare any dish existing in FOON. A task tree has the exact same structure of a subgraph.

Input:

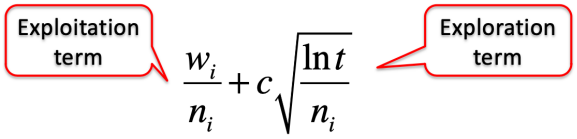
Your program should have the following input:

1. FOON (A .txt file)
2. A goal object to search (An object name and its state)
3. List of ingredients and utensils available in the kitchen (A kitchen file)
4. Success rates in the motion.txt file

Task:

Implement the MCTS based on the following steps

1. (25%) For each object i, simulate k random execution of subgraphs that producing object i; count how many were successful out of the k executions; select the function unit with the highest success counts.
    a. Decide the success or failure of a motion randomly using the success rates in the motion.txt file
    b. End a simulation with a motion failure is detected or the whole tree is successfully executed
2. (25%) Implement exploitation and exploration
    a. Exploitation: Keep track of average success rate for each child (functional unit) from previous searches; prefer child that has previously lead to more success
    b. Exploration: Allow for exploration of relatively unvisited children (moves) too
    c. Combine these factors to compute a "score" for each child; pick child with highest score at each successive node in search

    where w_i = number of success after object i,
    n_i = number of execution simulations to reach the object i,
    t = total number of task tree simulations
3. (25%) Recursively build task tree, where each round consists of:
    a. Selection: Starting at root, successively select best child nodes using scoring method until leaf node L reached
    b. Expansion: Create and add best (or random) new child node, C, of L
    c. Simulation: Perform a (random) task execution from C
    d. Backpropagation: Update score at C and all of C's ancestors in search tree based on execution results
4. (10%) Given a goal object g, run MCTS with simulation k=1,000, then generate the best task tree to make object g based on MCTS.

Outputs:

The task tree obtained using MCTS saved in a separate .txt file.

How your program will be tested:

We will have our own kitchen file and a few objects to search. With this kitchen file, we will search the objects in FOON and check the obtained task trees.

Submission:

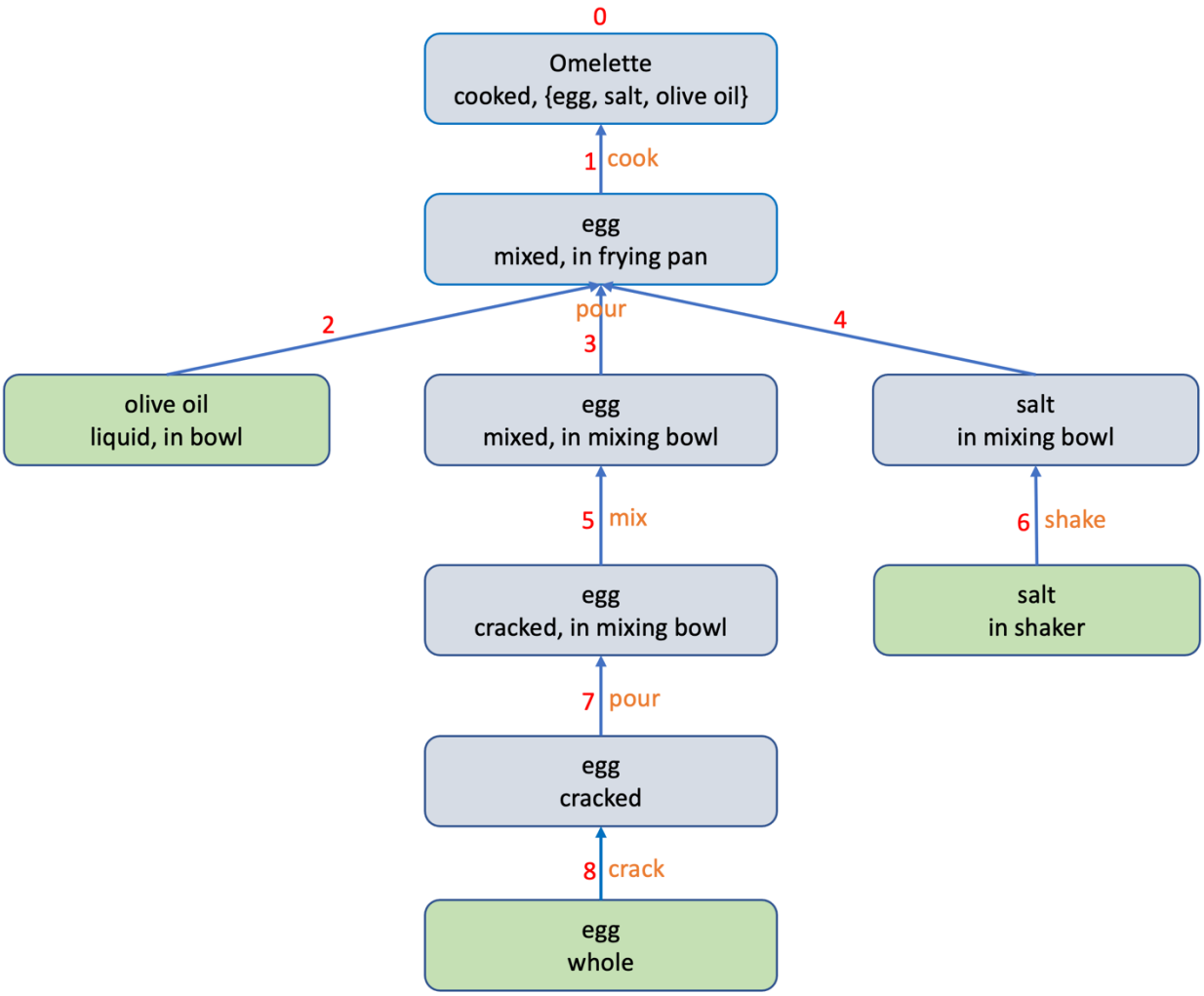
You can use any programming language for this project. Create a zip file including everything that are required to run your program. That is,

- FOON (.txt file)
- Your code (include a test script)
- Kitchen File
- The obtained task tree in a .txt file.
- A readme file with the instructions about how to run your program. Also mention if any package need to be installed.

Name the zip file with your name (e.g. james.zip). Submit it on Canvas.

This part of the project is worth 20% of project 1's grade.

Here is an example of how the search works in FOON.



Input

Object: Omelette

State: cooked, contains {egg, salt, olive oil}

Steps

- The input object is a dish we want to prepare. The node representing that object is referred as the goal node. The first step is to find the functional unit that has the goal node as an output object. In the above figure, node 0 is the goal node. The search starts from the goal node.
- Now, we need to search for the input objects that are required for the goal node. This search may vary depending on the algorithm we are using. If we explore the graph in a BFS manner, we will find the object in the order shown in the figure.
    - At node 3, you need to have egg mixed in a mixing bowl. If we find that there are more than one way to get the egg mixture, you can choose the first way you discover in case of iterative deepening. But, when you are using heuristic, you need to check all the candidate functional units and calculate the cost of the heuristic function. Then, you can decide which path to choose.

- For each node, we need to check if the ingredient already exist in the kitchen (check the kitchen.txt file). For example, in node 4, we need the salt in the mixing bowl but the kitchen does not have that. So, we further explore and discover node 6. Since, we have “salt in a shaker” in the kitchen, we stop there. Similarly, we stop at node 2 and 8 because they are available in the kitchen.