CAP5625 - Introduction to Robotics

Project 1: Knowledge Retrieval

Part 2: Implement search algorithm
Deadline: Sunday 8th October

For this part, we have created a Functional Object-Oriented Network (FOON) merging your labeling done in the previous part. Now, you have to implement search algorithms 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 .txt file)

FOON, sample goal objects and a kitchen file will be found in the provided starter code. The starter code and recorded video on this assignment will be found on Canvas module.

Tasks:

1. Implement two search algorithms.
    a. **Iterative Deepening Search:** Go through the class lectures to learn how this algorithm works. While you explore the nodes, you may find that there are multiple ways to prepare an object. Ideally, we need to explore all possible paths to find the optimal solution. But to make it simple, you can just take the first path that you find. You need to keep increasing the depth until you find the solution. A task tree is considered a solution if the leaf nodes are available in the kitchen.
    b. **Greedy Best-First Search**: To explore a node, instead of choosing a path randomly from various options, choose a path based on the heuristic function. You need to implement two different search method using the following two heuristic function.
        i. h(n) = success rate of the motion
        ii. h(n) = number of input objects in the function unit

        In case of heuristic 1, if you have multiple path with different motions, choose the path that gives higher success rate of executing the motion successfully. For example, a robot has higher success rate of *pouring* sliced onion compared to *slicing* a whole onion. The success rate of each motion is provided in the motion.txt file.

        For heuristic 2, if you find that *scrambled egg* can be prepared with either {egg, oil, cheese, onion} or {egg, oil, salt}, take the path that require {egg, oil, salt}. Because, in this path, you need fewer input objects.

2. Visualize the retrieved task trees and check if they make sense. You should use it to debug your program.
3. Save three task trees (one for iterative deepening search, one for heuristic 1 and one for heuristic 2) in three separate .txt files. If the goal node does not exist in FOON, print that "The goal node does not exist".

Output:

Three task trees saved in three separate .txt files.

How your program will be tested:

I will have my own kitchen file (different from what you will have) and a few objects to search. With this kitchen file, I will search the objects in FOON and check the retrieved task trees.

Submission:

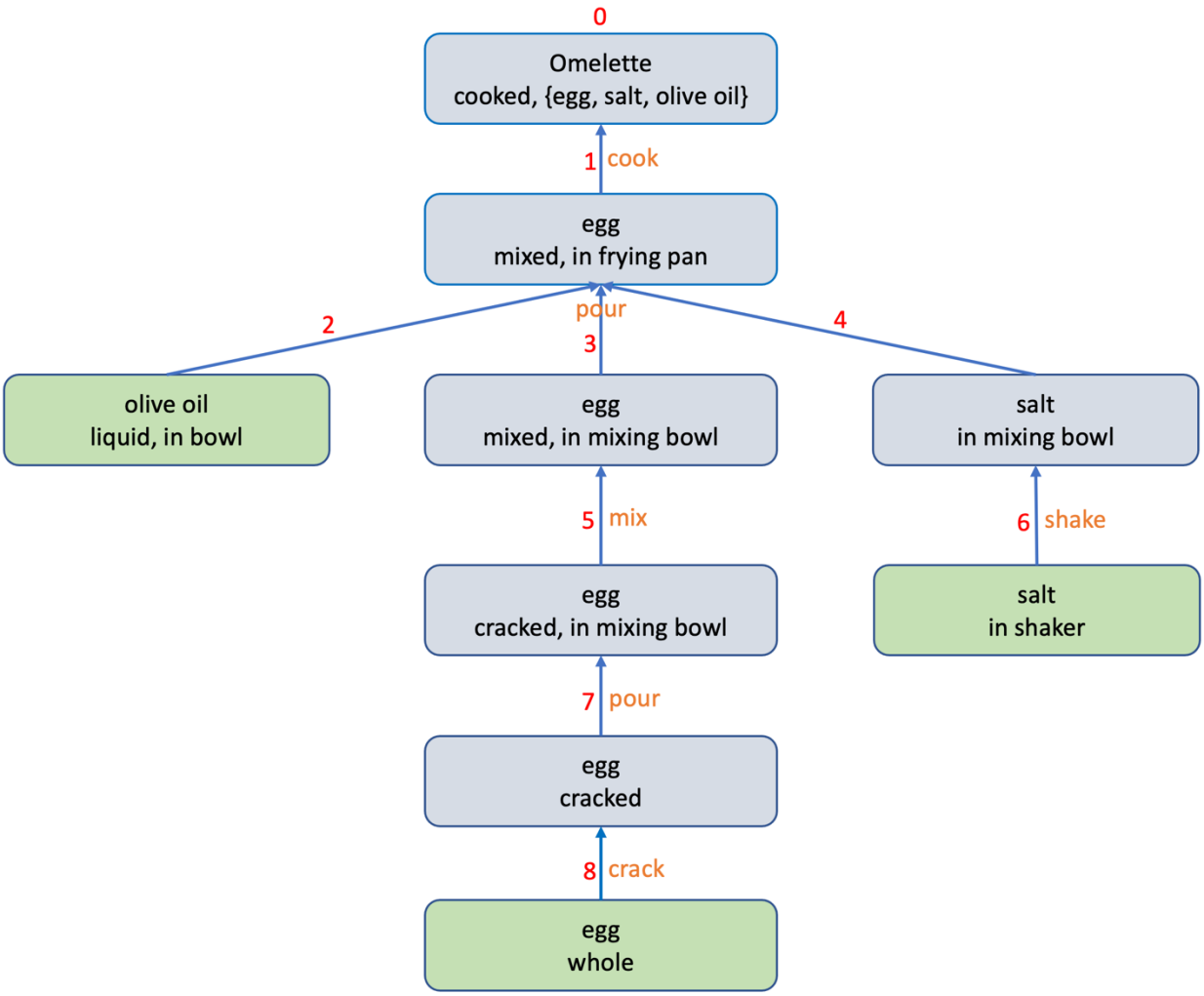
You can use any programming language for this project. Create a zip file including everything that are required to run your program. That is,

- FOON (.txt file)
- Your code
- Kitchen File
- Three retrieved task trees in three separate .txt files. (If there are 5 goal nodes given to you, then there will be 3x5 = 15 output files)
- 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 50% of your final project'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.