We aim at creating a new environment that captures a broader set of concepts: hierarchical planning, task and motion planning, POMDP, *etc.* The current version of the environment is simple, but it has already illustrated some ideas. In the future, as we are introducing more materials in the class, we can further extend this environment. We believe that this can help the students experience building a complex algorithm to solve a complex problem step by step, by combining all the knowledge learned from this class.

Setup. Tomas is hungry, and he is ordering a burger from a robotic food manufacturer. That is, they use robots to make burgers for customers. As a simplified version, we consider a discrete, deterministic, and fully-observed environment.
The environment involves a robot arm (we only consider the end gripper of the robot for simplicity), a table, and ingredients placed on the table. In this question, you will consider a 1-D version of the environment. That is, the table expands only in one dimension.
A burger is made of ground beef, greens, and two loaves of bread (one on the top and one at the bottom). Your goal is to build an agent that stack all ingredients in a specific order: the top bread first, then the ground beef, the greens, and finally, the bottom bread. Ground beef and the greens can not be directly put on the table. Instead, they need to be put in the containers or on the bread. Meanwhile, the environment is physically based: the containers and the loaves of bread can not float above the table. They must be either placed on the table, or on top of another bread/container.
The robot arm can move flexibly in the free space. However, if there are multiple loaves of bread or containers stacked together, the robot arm can only manipulate the bread/container at the top. It can pick up the containers, the bread, and also the ingredients in a container, and place them in a new position.

Written questions.

1. Define an environment that resembles the task described above. You can represent the position of each object as a tuple (x, h), where x is the position of the object on the table, and h is the height of the object. Your answer should include the state space, the action space, the successor function, and the goal test function.

Answer: See the description below.

2. Describe a possible heuristic function that can be used to accelerate your search procedure. Is your heuristic admissible? Is it consistent?

Answer: There are three relationships that need to be satisfied in the goal state: the top bread should be on the beef, the beef should be on the greens, and the greens should be on the bottom bread. One heuristic is to compute the number of unsatisfied relationships in the current state. Obviously, this heuristic is admissible and consistent. Another heuristic is based on the fact that the robot must move its arm to stack all ingredients. Thus, the number of movements (along the table axis) is always greater than s, where s is the smallest non-negative integer such that there exists some integer x such that the span $[x, x+s]$ contains all ingredients: the bread, the meat, and the greens. This heuristic is also admissible. It is also consistent because, at each step, the robot can only reduce the value of s by 1, by moving objects around.

Coding Questions

Starter Code: `https://github.com/zt-yang/6882-Robot-Kitchen`

We have provided a reference environment implementation. Implement an A algorithm with a uniform heuristic ($h \equiv 0$) and your custom heuristic to solve the task. You should run your algorithm in two settings. Both environments have the same state space, which contains the position of all objects.

Motion planning setting. In this case, the action spaces are: moving in four directions, picking up the object at the current gripper position, picking up the object in the container at the current gripper position (if there is a container at the current gripper position), and place the object at the current gripper position. The successor state function follows the physical rules defined above: all objects should be either placed on the table or on one of the other objects. The ground beef and the greens can not be directly put on the table: they can be placed either in the container, or on the bread / on each other.

Relational task planning setting. In this case, the action spaces are: pick up an object (one of the containers or the ingredients), and place an object on top of another object. For example, you can choose to pick the container of greens, and place it on the table. Next, you can grasp some greens from the container, and put them onto the bread.

1. Run your implementation of the A algorithm on both settings, record the number of expanded states in both cases, the running time, and the success rate.

   Answer: The results are:

Motion Planning				
Method	Success Rate (%)	Time (s)	# Node Expansions	# Env Steps
A* Uniform	0	5.00	1911	N/A
A* Custom	1	3.27	1549	26
Greedy Best-First Custom	1	0.08	76	17
Relational Task Planning				
Method	Success Rate (%)	Time (s)	# Node Expansions	# Env Steps
A* Uniform	1	2.14	519	7
A* Custom	1	2.26	519	7
Greedy Best-First Custom	1	3.75	1010	7

   Results are computed with a timeout of 5 seconds. We use the same heuristic for both settings. Thus, the performance of heuristic in the relational task planning is not ideal. The A* algorithm with a uniform heuristic fail to solve the planning problem. Thus, it has an N/A mark in its environmental steps.

2. Based on these data, describe the advantages and disadvantages of both algorithms.

   Answer: The motion planning setting (when the heuristic is both admissible and consistent) will produce the optimal policy for robot arm movements (that is, the total moving distance of the robot arm). However, it may get very slow when the world size (i.e., the size of the table) becomes large. On the contrary, the relational task planning may give a suboptimal solution to the original planning problem, but its speed is agnostic to the world size, making it suitable for large-scale problems.