**Project 2 (And HW7): Shut the box**

Shut the box is a traditional pub game. The game is played with two dice (with numbers 1-6 on them) and a box with numbers 1 to 9. In this game, the goal is to continuously remove numbers from the box so that the summation of numbers left in the box is as small as possible. If all the numbers are removed, it is called 'shut-the-box' for the player. The game was invented for multiple players, but it can also be played solo. In this project, we will play this game solo.


Detailed rules:

The game starts with 9 numbers (1 to 9) in the box.

In each turn:

1. The player rolls two dice, and gets a summation of the two dice. As an example, say the player rolls 2 and 4, then the summation is 6.
2. The player then picks any numbers that are still in the box (can pick multiple), whose summation equals the summation of dice. Those numbers are removed from the dice – In the example that the player rolls 2 and 4, the player can pick any combinations that are still in the box and adds up to 6, that is (1,2,3), (1,5), (2,4), or (6).
3. Continue until the player cannot pick any numbers to equal the summation

After the game is finished, the player gets a reward that equals all numbers removed from the box.

**Questions:**

a) (1 pts) Play some games with shut_the_box_interactive.py. Log your play in the report!
b) (1 pts) There are 11 different summations of two dice (from 2 to 12). Is the probability of these summations equal? Why?
c) (2 pts) A state in the game can be represented as a subset of {1, 2, 3, 4, 5, 6, 7, 8, 9} which represents the current numbers left in the box, together with the current roll (summation of two dice, from 2 to 12).

   • For example, {1,2,3,4,5,6,9} and 12 represents a state when 1,2,3,4,5,6,9 are in the box, 7,8 are removed, and you rolled two 6 which summed up as 12.

   For this definition, what is the total number of states here?

For the following part, let us use a tuple of two items (set of numbers left, current dice summation) to represent a state.

d) (2 pts) Now let's consider the transformation between states. For a state, you will take one action – pick the numbers to remove (or give up). Before you can take the next action, you rolled the dice. So, the transition should consider both steps.

   List all the possible states that can be transformed from ({1,2,3,4}, 2) in the report.

e) (2pts) We now implement a value iteration algorithm to find the optimal policy.
   - Complete Agent.value_iteration function in shut_the_box.py – most parts are already completed for you. The work left for you is to use the helper functions to implement the value iteration steps.
   - Run the code and answer the following questions. Put the answers in your report:
     - What is the utility for [1,2,3,4,5,6,7,8,9] and dice roll 12?
     - What is the utility for [1,3,4,5,6,7,8,9] and dice roll 12?
     - What is the utility for [1,3,5,6,7,8,9] and dice roll 12?

f) (2 pts) Finally, find out what is the best action for each dice roll summation (2,3,4, …, 12) at the beginning?
   - Complete the Agent.policy function in shut_the_box.py
   - Run the code and answer the following questions. Put the answers in your report:
     - What is the best action for [1,2,3,4,5,6,7,8,9] and dice roll 12?
     - What is the best action for [1,3,4,5,6,7,8,9] and dice roll 12?
     - What is the best action for [1,3,5,6,7,8,9] and dice roll 12?

**What you will submit:**

- (10 pts) shut_the_box.py – Make sure it is completed and runnable!
- (10 pts) A report doc – will be considered as HW7.

**Extra credit (2 pts):**

Consider the variants of the game with 1-10, and 1-12 in the box. What are the utility and best policy for each dice roll summation (2-12) at the beginning step, respectively? Report them in your doc.