Homework 6

Moodle Submission Deadline: 2023/6/18 (Sunday) 23:59

Problem 1: Candy Crush [hw6_1.py]

This problem is about implementing a basic elimination for Candy Crush. Given a 2D integer array `board` representing the grid of candy, different positive integers `board[i][j]` represent different types of candies. A value of `board[i][j] = 0` represents that the cell at position `(i, j)` is empty. The given board represents the state of the game following the player's move. Now, you need to restore the board to a stable state by crushing candies according to the following rules:

(1) If three or more candies of the same type are **adjacent vertically or horizontally**, **"crush" them all at the same time** - these positions become empty.

(2) After crushing all candies simultaneously, if an empty space on the board has candies on top of itself, then these candies will drop until they hit a candy or bottom at the same time. (No new candies will drop outside the top boundary.)

(3) After the above steps, there may exist more candies that can be crushed. If so, you need to repeat the above steps.

(4) If there does not exist more candies that can be crushed (i.e., the board is stable), then return the current board.

You need to perform the above rules until the board becomes stable, then return the current board.

Sample Input/Output: **(read from candy_input.txt, output to candy_output.txt)**

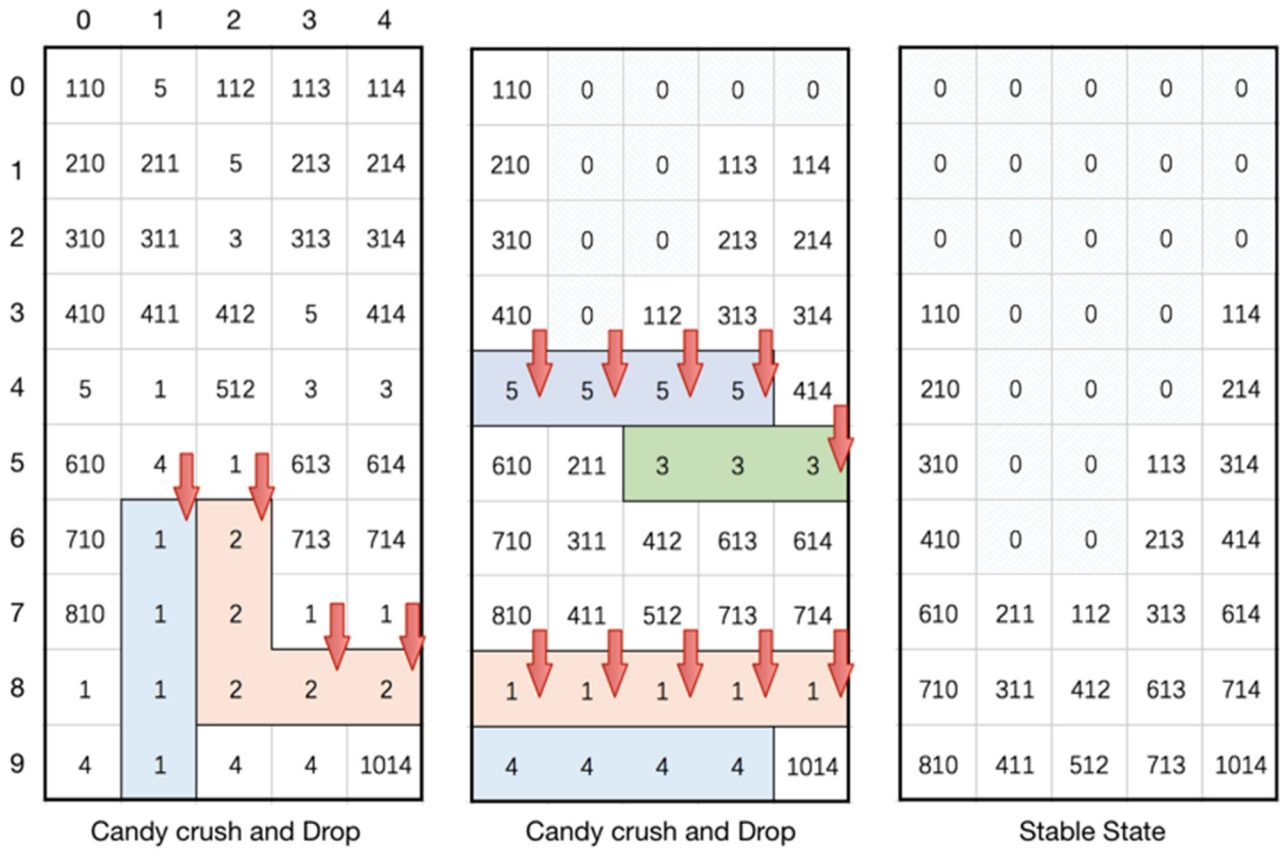
Example Input 1:

[[110,5,112,113,114],[210,211,5,213,214],[310,311,3,313,314],[410,411,412,5,414],[5,1,512,3,3],[610,4,1,613,614],[710,1,2,713,714],[810,1,2,1,1],[1,1,2,2,2],[4,1,4,4,1014]]

Example Output 1:

[[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[110,0,0,0,114],[210,0,0,0,214],[310,0,0,113,314],[410,0,0,213,414],[610,211,112,313,614],[710,311,412,613,714],[810,411,512,713,1014]]

Step-by-step Example 1:



Example Input 2:

[[9,9,7,9,9,9],[7,7,6,8,9,9],[5,6,5,6,8,8],[1,5,1,4,1,1],[2,1,4,1,1,1],[1,4,1,3,1,1],[1,1,2,1,3,1],[1,2,1,1,1,3]]

Example Output 2:

[[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[9,9,0,0,9,9]]

Problem 2: Interactive Candy Crush Game [hw6_2.py]

Based on problem 1, which contains only a round with multiple eliminations for Candy Crush, you have understood the basic actions of Candy Crush. Now you task is to construct a text-version interactive Candy Crush game, which is required to be run in console (command line) mode. You are no longer to read from files for the game. Instead, you need to generate the game – initializing all cells at the beginning in the board, and determining what candies to fall down until the board is full. Besides, a user is allowed to play with the game. She can decide which two adjacent candies to switch positions. The switch of two candies could lead to a series of elimination, like Problem 1. Instead of empty ("0") in each elimination, this time your game will have random candies falling down from the top of the board until the board is full of candies. Besides, you can determine your own scoring mechanism to score the user. More importantly, you need to have your own design to ensure the game can end at some time after multiple runs. More other functions (like allowing different difficulty levels, using real icon images to be the candies, allowing booms, etc.) are optional but very welcome. You are free to design the interactive manner to implement the following functions, which are the requirements in this problem. We give you large flexibility to design the details of each requirement.

(1) Before the game, the user is prompted to specify the width and the height of the board, and the number of candy types. Then the program generates the board accordingly.
(2) Use different numbers to denote various types of candies.
(3) Randomly determine candies at the beginning in the initial board.
(4) Allow the user to decide the two adjacent cells to switch.
(5) Switch may cause a series of eliminations.
(6) Randomly determine the next candies falling down from the top of the board after each run with possible eliminations.
(7) Show the board at each run of the game.
(8) Design your own scoring mechanism, and show the score in each run of the game.
(9) Ensure the game could end at some time after a number of runs operated by the user.

You may highly rely on ChatGPT for this problem. You are also welcome to implement by yourself. Your implementation may contain Python syntax or modules that you never learn in the class – you can learn by yourself or by ChatGPT/Google. That said, you can use any Python coding manners beyond what we teach you. Learning by yourself/ChatGPT is what we want you to practice in this problem. You can also think of this problem as your representative work that can be used in your future. So, try your best to let your implementation be as good as possible.

Need to write proper comments for each problem!

Important Notes

This is a homework for each **team**. Please submit your homework **by one team member.**

How to Submit Your Homework?

Submission in NCKU Moodle. Before submitting your homework, please zip the one submission file (**hw6_1.py**, **hw6_2.py**) in a zip file, and name the file as "學號 1_學號 2_hw6.zip". For example, if your 學號 of your team are H12345678 and H87654321, then your file name is:

"H12345678_H87654321_hw**6**.zip" or "H87654321_H12345678_hw**6**.rar"

When you zip your files, please follow the instructions provided by TA's slides to submit your file using NCKU Moodle platform http://moodle.ncku.edu.tw .

Have Questions about This Homework?

Please feel free to visit TAs, and ask/discuss any questions in their office hours. We will be more than happy to help you. Also remember that ChatGPT is also your TA.