Computer Science I — CSCI-141
What Happens in Vegas... — Lab 7

10/20/2020

1 Problem

On a trip to a Las Vegas casino, you are presented with the following card game, somewhat like a simplified variation of the card game solitaire. The game works as follows:

There are k cards in the deck. There is a card with the number 1 on it, a card with the number 2 on it, and so forth, up to the card with the number k on it. These cards are shuffled face down, and then you flip them over one at a time.

Your goal is to get as many cards as possible onto a *victory* pile. The *victory* pile starts out empty. The cards must be placed onto the *victory* pile in ascending numerical order. The first card placed on the *victory* pile must be the card with the number 1 on it, followed by the card with the number 2 on it, etc.

You are allowed two *discard* piles. Any time you flip a card from the deck and you can't (or choose not to) put it immediately onto the *victory* pile, you place it face up onto one of the *discard* piles.

At any point in the game, you are allowed to take the top card off of either of the *discard* piles, and move it onto the *victory* pile, if it is the appropriately numbered card to do so.

You are not allowed to move the top card from one *discard* pile to the other *discard pile*.

The game is over when you have flipped through all k cards in the deck, and you have moved as many cards as you can onto the *victory* pile.

Vegas is charging you \$5 to play the game, and paying out \$1 for every card you get onto the *victory* pile. For this lab, you will use data structures discussed in class and come up with a strategy for how to best use your *discard* piles to optimize your performance.

You will write a program to simulate the play of the game and determine whether you'd get rich or go bankrupt in the long run.

2 Problem-solving Session (15%)

You will work in a group as determined by your instructor. Each group will work together to complete the following activities.

1. Use whatever strategy comes to your mind and play out the game for the case where $k = 8$, and the cards from the deck are flipped over in the following order: $\{5, 4, 6, 7, 2, 8, 3, 1\}$. That is, you flip over the 5, then the 4, etc.

   Show how the contents of your *discard* piles and *victory* pile change as the game progresses, and indicate the final number of cards in the *victory* pile as well as the final state of the *discard* piles.

2. Instead of shuffling the deck in the usual way, we will generate a random order to deal the cards as follows. All of the cards $1, 2, ..., k$ will be put sequentially into a data structure to hold them. To determine the next card to be played, we will cycle through the cards, repeatedly taking one off of the top and replacing it on the bottom. We will do this a random number of times. When we stop cycling through the cards, the one on the top at that point is removed and returned as the next card to be played.

   What data structure will be useful to allow you the functionality you need to implement this type of random dealing?

3. Write code for a function that takes an instance of this data structure, and returns the next card dealt (modifying the data structure in the process to remove this card). You may assume that any relevant modules have already been imported. Make sure each card has an equal chance of being selected.

4. What data structure will be useful to represent the *discard* piles?

5. Describe a strategy for utilizing your *discard* piles to achieve a good result in the game.

3 Post-PSS Implementation (10%)

For your post-PSS work, create a file `vegas.py` containing the following functions:

- A function `init_deck` that creates and returns a new deck of cards of a given size, using an appropriate data structure.

- A function `random_draw` that takes an argument of a deck of cards and chooses and returns a randomly-selected card as described in problem solving. Note that this function should modify the deck so that the chosen card is no longer present in the data structure.

- A function `play_game` that takes an argument of a deck of cards and uses `random_draw` to deal out all of the cards one at a time, printing the card dealt each time.

  Note that for your final submission, instead of printing each card, this function will implement your chosen strategy to play the game, using appropriate data structures.

- A `main` function that prompts the user for the number of cards in the deck, creates a deck of the given size, and calls `play_game` appropriately.

Upload your `vegas.py` to the post-PSS dropbox in MyCourses by the due date given.

4 Implementation (75%)

Each student will *individually implement* and submit their own solution to the problem as a Python program named `vegas.py`. This program will implement your chosen strategy and simulate the play of a large number of games (with no user input except at the beginning) to estimate your long-term winnings. It must adhere to the following requirements:

- You must use the `node_types.py`, `cs_stack.py`, and `cs_queue.py` files provided with this week's lecture notes as part of your solution. These files should not be submitted (the SLIs have their own copy for grading). **You may not use Python's `list` for this assignment. You will receive no implementation credit if you use `lists`.**
- Your program must have a `main()` function which does the following:
  - Prompt for input corresponding to the number of cards to use in the deck. Also prompt for input corresponding to the number of games to play in the simulation. (It's ok if your program crashes on faulty input. We will not test your program on faulty input.)
  - Call whatever necessary function(s) to setup and play the game the requested number of times.
  - At the conclusion of the simulation, output the following statistics:
    - The average number of cards on the *victory* pile (this should be output as a floating point number).
    - The maximum number of cards ever achieved on the *victory* pile.
    - The minimum number of cards ever achieved on the *victory* pile.
- Your code must not prompt for additional input beyond that indicated in the previous bullet.
- For full credit, you must employ a strategy for utilizing the *discard* piles that results in an average *victory* pile size of 4 or more when tested with a card deck of size 10. Recall that it costs \$5 to play the game, and you get \$1 back for each card on the victory pile, so you can make money if your average *victory* pile size is at least 5 when tested with a card deck of size 10. Since we can't actually give you money, you can earn a 5% grade bonus by implementing a strategy that achieves this level of performance.

5 Grading

- 15%: The program uses `cs_stack.py`, `cs_queue.py` and `node_types.py`.
- 25%: The program simulates the game play correctly.
- 25%: The program employs a strategy that has an average *victory* pile size of 4 or greater when playing the game with 10 cards. Bonus: 5% bonus for a strategy that achieves an average *victory* pile size of at least 5 when playing the game with 10 cards.
- 10%: The program correctly prompts for input, simulates the game the requested number of times, and displays the requested output information.
- Up to -10%: Improper style and/or improper or missing documentation.

6 Submission

Zip your `vegas.py` file into a file called `lab07.zip` and submit this to the **myCourses dropbox** by the due date for this lab.