Homework 10
Due Friday, April 28 at 3:00 PM

“I exclusively eat breakfast, write data structures and algorithms, and watch Shia Labeouf movies.” - Shia Labeouf

Handing In
To hand in a homework, go to the directory where your work is saved and run 
\texttt{cs0160\_handin hwX} where X is the number of the homework. Make sure that your written work is saved as a .pdf file, and any Python problems are completed in the same directory or a subdirectory. You can re-handin any work by running the handin script again. We’ll only grade your most recent submission. To install stencil Python files for a homework, run \texttt{cs0160\_install hwX}. Please leave room between questions and in the margins on your pdf so that your grader can leave feedback on your work. \textbf{You will lose points if you do not hand in your written work as a .pdf file.}

1 Written Problems

Problem 10.1

Racing Horses

Ignorant Bet Person (IBP) is watching 2 horses (A and B) race with members of the breakfast club, Special Kei (SK), Captain Crunch (CC), Tony the Tiger (TT), and Shia LaBeouf (SL). True to the name, IBP wants to bet on which horse will win each race, but sadly knows nothing about the horses that are racing. IBP instead decides to bet based on the advice the other 4 breakfast club members provide.

Ideally, in an offline situation, IBP would only listen to the breakfast club member who knows the most about the horses. However, and once more true to the name, IBP doesn’t know who that is! Instead, to deal with this online situation, IBP decides to bet using a weighted majority vote based on the advice given by the other breakfast club members. IBP initially considers each of the other members equally with respective weights of 1. When a member gives bad advice, IBP weighs all of their subsequent advice exactly one third of the previous weight. If there is a tie, IBP will always bet on horse A winning the race. Given the table below of advice and actual winner, fill in IBP’s bet for each race. The first bet has been filled out for you.

How much weight do each of the experts have after the end of the fourth race?
Problem 10.2

Computability

In class, we talked about classifications of computability (P, NP, NP-Hard, NP-Complete).

Part 1:

For the following problems, state which complexity class each of the problem falls under, and explain your answer in one to two sentences. For problem 3, in three or four sentences, describe the algorithm to solve this problem.

1. Sorting a list of numbers

2. Creating a seating arrangement of \( n \) people at \( k \) tables, such that no two people who hate each other are seated at the same table. You can assume that you know how two arbitrary people \( k_A \) and \( k_B \) feel about each other.

3. Determining whether it’s possible to assign each of a graph’s vertices one of two colors, such that no two vertices that share an edge have the same color.

Part 2:

In class, we talked about P and NP as if they were known to be different complexity classes. In reality, it’s not known whether P = NP, although in most cases we assume P \( \neq \) NP.


Consider that RSA encryption (one of the most common encryption techniques) is in NP. In one to two sentences, explain what it would mean with regards to this if it was proven that P = NP.
Problem 10.3

Bitcoin

We learned about bitcoin in class.

Tell us about an interesting bitcoin transaction in a paragraph or less. At least one reference/link is required.

Problem 10.4

Algorithmic Fairness

We learned about machine learning and bias in algorithms during class.

If someone writes a machine learning algorithm that is biased in its results (it’s incredibly difficult to avoid all bias...), who do you think is responsible? And why? Some examples of individuals involved may be the folks who provided the data that the algorithm was trained on, the programmer who wrote the algorithm, the owner of the computer on which the computations were performed, the company or organization that the programmer works for... but there are many more!

There’s no correct answer to this problem - we’re interested in your moral intuitions.

2 Python Problems

Problem 10.5

Functional Programming Practice

In class we talked about two higher order functions called map and reduce. Solve each of the following problems using only python’s built in map or reduce (it is up to you to decide which one is appropriate for the problem). The functions you pass into map or reduce must be anonymous functions. Your solutions should go in the stencil functional.py. You might have to think about each of these for a while, but the solution to each one is a single line of code (besides the error handling).

Part 1: apply_all

Fill in the function apply_all in the stencil. This function takes in a list of unary (one argument) functions and a number. It returns a new list with each of those functions applied to that number. In other words, if you pass in the list of
functions \( [f(x), g(x), h(x)] \) and the number \( n \), apply_all should produce \( [f(n), g(n), h(n)] \).

Example

- \( \text{apply\_all}([\lambda x: x+1, \lambda x: x+2, \lambda x: x+3], 4) \rightarrow [5, 6, 7] \)

Part 2: compose

Fill in the function compose in the stencil. This function takes in a list of unary functions and a number \( n \). Compose should compose all of the functions in the list and apply to \( n \) to produce a single number. The inner-most function in the composition will be the first function in the input list, and the outer-most function in the composition will be the last function in the input list. In other words, if you pass in the list of functions \( [f(x), g(x), h(x)] \) and the number \( n \), compose should produce \( h(g(f(n))) \).

Example

- \( \text{compose}([\lambda x: x+1, \lambda x: x+2, \lambda x: x+3], 4) \rightarrow 10 \).

Part 3: list_compose_steps

Fill in list_compose_steps, which should take in a list of unary functions and a number \( n \). It should return a list with each of the intermediate values produced by compose. In other words, if you pass in the list of functions \( [f(x), g(x), h(x)] \) and the number \( n \), list_compose_steps should produce \( [n, f(n), g(f(n)), h(g(f(n)))] \). Please note that using append will not work in your lambda expression because append doesn’t return anything. To append some number \( x \) to a list, you should write list+[x] instead of list.append(x).

Example

- \( \text{list\_compose\_steps}([\lambda x: x+1, \lambda x: x+2, \lambda x: x+3], 4) \rightarrow [4, 5, 7, 10] \).

Keep in mind that you should never call compose and that you can only use map or reduce. Also remember that the number \( n \) should be the first element in your output list.

All of these should raise an InvalidInputException if any of the inputs are None.
Testing

Write your test cases in `functional_test.py`. A few examples have already been filled in for you.