Homework 2
Due Friday, May 28 at 11:59 PM

Fun Sea Creature fact: Dolphins rest half of their brain at one time!

Installing and Handing In
Accept the GitHub assignment [here](#). Homework is handed in through Gradescope. For written portion, submit a PDF and match the questions accordingly. For the coding portion, submit the .py files.

Overview
Homework is handed in through Gradescope. For written portion, submit a PDF and match the questions accordingly. For the coding portion, submit the .py files.

- We will not accept paper handins. You must submit all written work as a PDF. For information on how to make PDFs of your work, see the very handy [PDF guide](#). Please make sure to NOT include any identifying information, e.g. name or banner info on your handin, as we are anonymizing grading - we appreciate your help with this.

- Please follow proper pseudocode formatting guidelines. See our [pseudocode standards](#). If you use \LaTeX, use the `newalg` or `algorithmic` packages or our CS16 `pseudo` environment to format your pseudocode. For even more wizardry, check out the Docs section of our website for a \LaTeX handout and tips on improving your pseudocode.

1 Written Problems

Problem 2.1

Argmax

1. Write pseudocode for the function \texttt{argmax}(L, f), where \( L \) is an array and \( f \) is a function. Your pseudocode for \texttt{argmax} should find and return the element in the array \( L \) for which \( f \) is maximal. For example, given \( L = [1, 3, 4, 2] \) and \( f(x) = x^2 \) then \texttt{argmax}(L, f) should return 4 because \( 4^2 > 1^2, 3^2, 2^2 \). If there are two elements that maximize \( f \), return the first. You can assume that elements of \( L \) are valid inputs for \( f \), and that \( f \) outputs real numbers. You can also assume that \( L \) isn’t empty or null.
2. Let’s use the variable $R$ to refer to the worst-case runtime of $f$ on an element in $L$. Let’s use the variable $n$ to reference the size of array $L$. Give us the big-O runtime of $\arg\max$ in terms of $n$ and $R$. Explain how you came to this conclusion.

Problem 2.2

Big $\Theta$ Notation

Demonstrate that $f(n) = 10 + 2^{3\log_2 n} + 5^{8\log_5 n}$ is $\Theta(n^8)$.

**Reminders**: Remember that big-$O$ is an upper bound, big-$\Omega$ is a lower bound, and big-$\Theta$ is a *tight* bound (i.e. upper and lower). Meaning, to prove that $f(n)$ is $\Theta(g(n))$ you must prove both that $f(n)$ is $O(g(n))$ and that $f(n)$ is $\Omega(g(n))$. Consider how you would show that a function is big-$O$ or big-$\Omega$ of another function by looking at what integer values $n$ can take on. Make sure you prove this using the formal definition of big-$\Theta$, not just an intuitive explanation.

**Hint**: You may want to simplify your function! This fun property of logarithms might be helpful, $a^{\log_2 n} = n^\frac{1}{\log_2 a}$.

2 Python Problems

In this case, we have checked for valid input for you, but make sure to do so on your own in the future! Remember, empty lists are valid input!

How To Test and Run Your Code

For this homework and all subsequent python coding assignments, you will be required to hand in a set of test cases for each python problem. **You must write your own test cases in addition to the example ones provided. The example tests do not count as your own.** You should use these tests to confirm that your algorithms work as expected, but they will also be graded according to how comprehensive they are. We will grade your tests by running them against broken implementations of the problems. The more errors your tests catch, the better. When writing tests, try to think of every possible edge case and every kind of input that could be passed into your functions. Keep in mind though that writing many tests which are similar will not earn a better score. Quality over quantity!

We have provided three stencil test files in which you should write your tests: `arraysearch_test.py`, `arraylessthan_test.py`, and `maxword_test.py`. These files have a few example tests filled in to show you how to write your own. Define new functions for your tests, naming them descriptively according to what they are testing for. Fill in these functions with `assert` statements.

```python
assert
```
takes in a conditional and a string. If the conditional is true, it continues. If it is false, your code stops executing, an error is thrown, and the assert statement string is printed to the console. It is fine for your testing functions to contain multiple assert statements as long as they are all related (they should logically fall under the same descriptive testing function name). As a rule of thumb, you should write a new function for each different case you are testing for. Make sure to follow the instructions in the stencil and add the name of each function you write to the list in get_tests().

Examples:
assert max(1, 2) == 2, 'Test 1 failed' will pass
assert max(1, 2) == 1, 'Test 2 failed' will fail, causing your code to terminate and 'Test 2 failed' to be logged

Running your tests: To run your code and your tests, you should run the test files rather than the files in which you wrote your code. For example, to run your arraysearch code, you should run arraysearch_test.py by typing python3 arraysearch_test.py from your hw2 directory. Additionally, follow this guide to install pytest if you do not already have it.

Problem 2.3
arraysearch
Overview
In arraysearch.py implement the array_search method, which finds whether an int is contained within an array.

Input/Output
You can assume your function will be tested on anything that fits within the below definition for Input, but will not be tested on inputs that don’t fit within the below input definition.

Input: An int and an array of numbers.

Output: A boolean that represent whether the int is contained within the list of integers.

Example
array_search(3, [1, 3, 4]) -> True
array_search(3, [1, 2, 4]) -> False

Details
• You may not assume that the array is ordered.
• Although not necessary, you are allowed to use Python indexing and “slices” (as in array[2:4]) if you wish.

• You are not allowed to use python’s x in(array). While this is an awesome thing to know about (and you should definitely look at the documentation for it and use it in the future), it would make this problem trivial.

Problem 2.4
arraylessthan

Overview

In arraylessthan.py implement the array_less_than method, which finds whether each element of one list is strictly less than the corresponding element in another list.

Input/Output

You can assume your function will be tested on anything that fits within the below definition for Input, but will not be tested on inputs that don’t fit within the below input definition.

Input: Two arrays of numbers, let’s call them arrays p and q, where p is the first argument of array_less_than, and q is the second.

Output:

• If both input arrays contain the same number of elements, return a boolean that represents whether each element of p is strictly less than the corresponding element of q.

• If the number of the elements in the input arrays are not the same, return False.

Example

array_less_than([0, 2, 2], [1, 3, 4]) -> True
array_less_than([1, 5, 3], [6, 8, 2]) -> False

Problem 2.5
maxword

Overview

In maxword.py implement the max_word method, which, given a sentence, finds the most used word in the sentence.
Input/Output

You can assume your function will be tested on anything that fits within the below definition for Input, but will not be tested on inputs that don’t fit within the below input definition.

**Input:** A string.

**Output:** Returns the number of occurrences of the most common word in the string.

Example

```python
max_word('"hello world!"') -> 1
max_word('"the quick brown fox jumped over the lazy dog"') -> 2
```

Details

- You do not have to take into account punctuation or capital letters. All words will be separated with spaces.
- Your solution must run in O(n) time
- In order to break up the string into individual words, you should look into Python string manipulation. In particular, you will need to use the built-in Python function `string.split()` (doc linked [here](#)). Look into what this function does!
- Using the built-in Python `dict` datastructure will be very helpful. A `dict` (short for dictionary) is a data structure that allows you to store an object or value in relation to another object or value. We call this “mapping” a key to a value. As an example, you can think of a real life dictionary as a mapping between words and their definitions. In an upcoming lecture, we will go into further detail on how dictionaries really work. Here is how you instantiate and use a dictionary in Python:

```python
my_dictionary = {}
my_dictionary['hello'] = 1  # 'hello' is the key, 1 is the value
my_dictionary['world'] = 2
print my_dictionary['hello']  # prints 1
print my_dictionary['world']  # prints 2
```

For this problem, think about what you should use for keys and what you should store in the dictionary

- You will also need to loop through your dictionary once you have filled it in order to find the most common word(s). Read this documentation to find out how to traverse your dictionary.