Homework 2

Due Friday, February 9 at 5:00 PM

Prepare for trouble, and make it double
To protect the world from devastation,
To unite all people within our nation,
To denounce the evil of truth and love,
To extend our reach to the stars above
JESSIE! JAMES!
Team Rocket blasts off at the speed of light,
Surrender now, or prepare to fight
MEOWTH, that’s right!
- Jessie, James, and Meowth, Team Rocket, *Pokemon* (1996)

Handing In

To hand in a homework, go to the directory where your work is saved and run `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 `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. **You will lose points if you do not hand in your written work as a .pdf file.**

1 Written Problems

Problem 2.1

Argmax

1. **Silly premise:** Team Rocket is on the hunt for Pikachu. They want to capture Pikachu when the most people will be around to see their victory. Lucky for them, Jessie has a function $f$ that tells her when the most number of people are in the area to set the scene for Team Rocket’s announcement.

Write pseudocode for the function $\text{argmax}(L, f)$, where $L$ is an array and $f$ is a function. Your pseudocode for $\text{argmax}$ should find and return the element in the array $L$ for which $f$ is maximal. 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 assume that $L$ isn’t empty or null.
2. Describe the running time (give us a big-O relation) of \( \text{argmax} \) in terms of \( n \) (the size of the array \( L \)) and \( R \) (the worst-case running time of \( f \) on any element in \( L \)), and explain how you came to this conclusion.

**Problem 2.2**

**Big \( \Theta \) Notation**

Demonstrate that \( f(n) = 10 + 4 \cdot 2^{\log_4 n} + 6 \cdot 5^{\log_6 n} \) is \( \Theta(n^5) \). 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). Consider how you would show that a function is big-O or big-\( \Omega \) of another function by looking at what different integer values \( n \) can take on. 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 \( \Omega(g(n)) \). 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, \( a^{b \log_a n} = n^b \).

**Problem 2.3**

**Greedy vs. Dynamic Programs**

Let \( G \) be a graph (a collection of nodes and directed arcs) with \( n \) nodes, labeled 1, 2, \ldots, \( n \), and arcs as drawn in Figure 1 for \( n = 8 \). There are two types of arcs:

1. For \( 1 \leq i \leq n - 1 \), there is an arc \((i, i + 1)\).
2. For $1 \leq i \leq n - 2$, there is an arc $(i, i + 2)$.

Let $\text{cost}(i, j)$ be a function that takes as input two nodes connected by an edge in $G$ and returns the cost of the edge between them. See Figure 2 for an example of arcs labelled with their costs. Consider the problem of finding a low-cost path from node 1 to node $n$ (the cost of a path is the sum of the costs of the arcs on the path).

For the following problems you do not have to give detailed code. It suffices to clearly explain your algorithms in a few sentences or formulas. Your solution only needs to return the minimum cost, not the path itself.

1. Describe a greedy algorithm for solving the problem. A greedy algorithm is an algorithm that makes the best choice locally at each stage with the hope of finding a global optimum solution. Does your algorithm find the optimum for Figure 2? Will it always find the optimum? Prove or give a counterexample. (A counterexample would be a single instance of a graph in the specified form, and a set of weights for that graph (i.e., a picture like Figure 2 above), with the property that the path produced by the greedy algorithm is not optimal).

2. Describe a dynamic programming approach (like the one we used for solving seam-carving) to find an optimal solution. An algorithm that uses dynamic programming breaks the larger problem into smaller subproblems then uses the solutions to those subproblems to build up to solving the overall problem. See the lecture slides for more information.

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 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 maxseq_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. 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 python arraysearch_test.py from your hw2 directory.

Problem 2.4
arraysearch
Implement the following method:

• array_search takes in an int and an array (a.k.a. python lists) and returns True if the int is in the array. (i.e. array_search(3, [1, 3, 4]) -> True). 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.

• Do not, however, use the python tool 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 2.5
arraylessthan
Implement the following method:

• array_less_than takes in two arrays, p and q, and returns True if p and q have the same size and each element of p is less than its corresponding element in q, otherwise returns False. You can assume that the arrays will not be empty.
Examples:
array_less_than([0, 2, 2], [1, 3, 4]) -> True
array_less_than([1, 5, 3], [6, 8, 2]) -> False

Problem 2.6
The Pokémon Card Conundrum

Upon their successful capture of Pikachu, Team Rocket has decided to sell Pokémon Cards to celebrate the momentous occasion. After hearing about this incredible opportunity, you and your significant other (and perhaps your significant other’s significant other) have decided to invest in Team Rocket’s incredible Pokémon Card company. Lucky for you, you possess mystical powers and are able to predict the future market value of Pokémon Cards, and have decided to cash in on the opportunity in order to fuel your own Pokémon adventure and contribute to late stage capitalism.

Suppose you currently have 180 dollars to invest. You know how much the value of Pokémon Cards will fluctuate for \( n \) days, and you’ve written it down in an array of \( n \) numbers \( r[0] \ldots r[n-1] \), where \( r[i] \) represents the change in value for day \( i \).

You can choose the days on which you buy and sell Pokémon Cards. You can only buy and sell them once, so you must hold onto them for a number of consecutive days. Your goal is to maximize your profits. Use Python to implement a linear time algorithm for finding the maximum amount money you could have. Assume that Pokémon Cards don’t lose value on their own.

Example:
Let \( r = [-1, 2, 7, -8, 13, -2] \). To achieve maximum profits, you should buy on the second day \( (i = 1) \) and sell on the fifth day \( (i = 4) \). Therefore, when you sell out he will have 180 + 2 + 7 - 8 + 13 = 194 dollars. Your function should return 194.

Notes:

(a) Money is gained (or lost) on both the day you buy and sell Pokémon Cards. For example, if you buy on the 12th day and sell on the 14th and if \( r[11] = 10, r[12] = -3, r[13] = 4 \) then you will end up having 180 + 10 - 3 + 4 = 191 dollars.

(b) Your algorithm is not required to return the buying and selling days. You only need to return the maximum money that can be achieved by your investments.

(c) Remember that one option is to not buy at all, leaving you at 180 dollars. Another option is for you to buy and sell on the same day (your worth would be 180 + the dollar change for that day).