Homework 1: Big O and Unit Testing

Due: 11:59pm EST February 7, 2021

Acknowledgements: This homework assignment contains problems from an assignment originally written by Alice Paul. Additionally, problem 2 was adapted from a problem sourced from [external source linked in solution document]; content on that site is licensed under the Creative Commons license, and as such, so is our adapted problem.

Concepts: Big-O analysis, pseudocode, lists, unit testing

Theory

1. (5 points) Explain how you would implement list concatenation and analyze the runtime using big O analysis. This is not asking for Python commands, but a description of how you would create and modify arrays to concatenate 2 lists. Be sure to define any variable(s) you use in your analysis. For example, you may let a be the resize constant for a list.

2. (10 points) Unimodal Search: An array $A[1, \ldots, n]$ is unimodal if it consists of an increasing sequence followed by a decreasing sequence. In other words, the array is unimodal if there is an index $m \in \{1, 2, \ldots, n\}$ such that:

   - $A[i] < A[i+1] \quad \forall \quad 1 \leq i < m$, and
   - $A[i] > A[i+1] \quad \forall \quad m \leq i < n$

   If you are unfamiliar with the "$\forall$" symbol, it simply means "for all". You can refer to this link for an explanation of this symbol and other mathematical symbols.

   (a) (5 points) Give an algorithm (in pseudocode) to compute the maximum element of a unimodal input array $A[1, \ldots, n]$ that runs in $O(\log n)$ time. Explain (in one or two sentences) why your algorithm runs in $O(\log n)$ time.

   (b) (5 points) Prove the correctness of your algorithm by identifying the loop invariant, initialization, maintenance, and termination. *Hint: any sub-array (of at least length 2) that contains the maximum value is also unimodal.* We are not asking for a rigorous proof (that will come later in the course), but rather an informal one that follows the appropriate structure. To give you a sense of what we are looking for, here is how the insertion sort algorithm is proved in the textbook:

Algorithm 1 Sort the array A using the insertion sort method (shown is pseudocode using 1-indexing).

```
1: Function InsertionSort(A)
2: j ← 2
3: while j ≤ length(A) do
4:     key ← A[j]
5:     i ← j − 1
6:     while i > 0 and A[i] > key do
7:         A[i + 1] = A[i]
8:         i = i − 1
9:     end while
10:    A[i + 1] = key
11: end while
12: return A[a]
```

- **Loop invariant:** At the start of each loop, $A[1, \ldots, j-1]$ is sorted.
- **Initialization:** To show that the loop invariant *holds before the first iteration*, it can be trivially stated that because j is initialized to 2, the array $A[1, \ldots, j-1] = A[1]$ is sorted.
- **Maintenance:** The algorithm works by shifting values in $A[1, \ldots, j-1]$ to the right as necessary such that the value originally at $A[j]$ ends up in the correct position, resulting in $A[1, \ldots, j]$ being sorted by the end of each loop. As such, incrementing j for the next loop preserves the loop invariant.
- **Termination:** The loop terminates when $j = \text{length}(A) + 1$. Substituting this value of j in the loop invariant shows us that the array $A[1, \ldots, \text{length}(A)] = A$ is sorted.

Importantly, make sure that you justify any statements that you make in your loop invariant.

Practice

Unit testing is a part of good programming practice. As part of this course, you will use the `pytest` package, which can be installed using pip or Anaconda. You can find the `pytest` documentation at `http://doc.pytest.org/en/latest/assert.html`, and the provided template code shows you how you should structure your tests.

3. (10 points) Write a function that implements your algorithm from Q2 and a corresponding test function. Be sure to include appropriate documentation in your code (e.g. docstrings and any necessary comments) and to use the template Python file provided by the teaching team.

   Your tests should cover both nominal and edge-case (e.g., an array of length 2) scenarios. Your tests and code can assume that the array passed to the function is valid; i.e., that it has a length ≥ 2 and is, in fact, unimodal.