Practice Exam 1

February 4, 2022

Instructions:

1. Exam 1 will cover the concepts from lectures 1-6 and homeworks 1(a)-1c. Absence of a topic from this practice exam does NOT imply an absence from the actual exam.

Problem 1. *(18 points)* Indicate whether the following statements are **true** or **false**.

a.) $4^{\log_2(n)} = \Theta(n^2)$

b.) $n^2 = \Omega(n^{\log_3(11)})$

c.) $n^n = \Omega(n!)$

d.) $n! = \Omega(2^n)$

e.) $2^{\log(n)^2} = \Theta(n)$

f.) $\log(n^3) = \mathcal{O}(\log(n))$

Problem 2. Solve the following number theory problems.

a.) Calculate $13^{13} \bmod 10$.

b.) Calculate the multiplicative inverse of 3 mod 121.

c.) True or False: If a has a multiplicative inverse mod b, then b has a multiplicative inverse mod a.

d.) Let p be a prime number, and let $c \equiv c' \mod (p-1)$. Show that, for any $a < p$, $a^c \equiv a^{c'} \mod p$.

Problem 3. Recall from class that we have discussed a dynamic programming approach to solve the Maximum Sum Subarray problem. Here we explore using a divide-and-conquer to solve this problem instead. As a reminder, a subarray of a list of elements $a_1, ..., a_n$ is a consecutive (no gaps) subsequence of the elements of the list. For example, 2,5,-4 is a subarray of 1,2,5,-4,-7,7 but 2,-4,7 is not.

Given an input $a_1, ..., a_n$, we are interested in finding the sum of the elements of the subarray whose elements sum to the maximum value out of all the possible subarrays. You do not need to return this subarray itself, just the sum of its elements. For example, the maximum sum subarray of 1,2,5,-4,-7,7 is 1,2,5, and our algorithm would return 9 (since $1 + 2 + 5 = 9$).

I.) A naive brute-force approach would simply loop through all the possible subarrays, calculating the sum each time. What is the runtime of this algorithm?

II.) Suppose you are given a list of integers $a_1, ..., a_n$ which are guaranteed to be such that the maximum sum subarray spans the middle of list (i.e. it contains at least one element from the first half and at least one element from the second half.)

Explain how to solve Maximum Sum Subarray for this case in $O(n)$ time.

III.) Give a divide-and-conquer approach for Maximum Sum Subarray with run-time $O(n \log n)$. (Note that, if the max sum subarray does not cross the middle of the list, it must lie either entirely in the first half or entirely in the second half.)

Problem 4. Answer the following problems which analyze func.
Let n be a power of 4. Suppose we have the following function:

```
def func(n):
    if n == 1:
        return
    for i in [1,2,3]:
        func(n/4)
    for i in range(n):
        for j in range(n):
            print("Hello")
```

I.) Let $H(n)$ denote the number of times of "Hello" is printed. What is the recurrence relation for $H(n)$?

II.) What is the tight upper bound for $H(n)$?