DSC 40B - Homework 02

Due: Wednesday, April 17

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope at 11:59 p.m.

Problem 1.

State the growth of each function below using Θ notation in as simplest of terms possible, and prove your answer by finding constants which satisfy the definition of Θ notation.

E.g., if $f(n)$ were $3n^2 + 5$, we would write $f(n) = \Theta(n^2)$ and not $\Theta(3n^2)$.

a)

$$f_1(n) = \frac{n^3 + 10n}{\sqrt{n} - 100}$$

b)

$$f_2(n) = \log(n^2 + 10)$$

c) Using properties of Θ, what is the asymptotic growth of $f(n)$ defined as

$$f(n) = f_1(n) \times f_2(n)$$

Problem 2.

Suppose $T_1(n), \ldots, T_6(n)$ are functions describing the runtime of six algorithms. Furthermore, suppose we have the following bounds on each function:

$$\begin{aligned}
&T_1(n) = \Theta(n^2) \\
&T_2(n) = O(n \log n) \\
&T_3(n) = \Omega(n) \\
&T_4(n) = O(n^2) \text{ and } T_4 = \Omega(n) \\
&T_5(n) = \Theta(n^3) \\
&T_6(n) = \Theta(\log n) \\
&T_7(n) = O(n^{1.5} \log n) \text{ and } T_7 = \Omega(n \log n)
\end{aligned}$$

What are the best bounds that can be placed on the following functions?

For this problem, you do not need to show work.

Example: $T_1(n) + T_2(n)$.

Solution: $T_1(n) + T_2(n)$ is $\Theta(n^2)$.

a) $T_1(n) + T_5(n) + T_2(n)$

Solution:

b) $T_4(n) + T_5(n)$

Solution:

c) $T_7(n) + T_4(n)$

Solution:

d) $T_3(n) + T_1(n)$

Solution:

e) $T_2(n) + T_6(n)$

Solution:

f) $T_1(n) \cdot T_4(n)$

Solution:

Problem 3.

In each of the problems below compute the average case time complexity (or expected time) of the given code. State your answer using asymptotic notation. Show your work for this problem by stating what the different cases are, the probability of each case, and how long each case takes. Also show the calculation of the expected time.

You may assume that `math.sqrt` and `math.log2` take $\Theta(1)$ time.

a)
```python
import random
import math
def boo(n):
    # draw a number uniformly at random from 0, 1, 2, ..., n-1 in Theta(1)
    x = random.randrange(n)
    if x < math.log2(n):
        for i in range(n**2):
            print("Very unlucky!")
    elif x < math.sqrt(n):
        for i in range(n):
            print("Unlucky!")
    else:
        print("Lucky!")
```

b)
```python
def bogosearch(numbers, target):
    """search by randomly guessing. `numbers` is an array of n numbers"""
    n = len(numbers)

    while True:
        # randomly choose a number between 0 and n-1 in constant time
        guess = np.random.randint(n)
        if numbers[guess] == target:
            return guess
```

In this part, you may assume that the numbers are distinct and that the target is in the array.

Hint: if $0 < b < 1$, then $\sum_{p=1}^{\infty} p \cdot b^{p-1} = \frac{1}{(1-b)^2}$.

Problem 4.

For each problem below, state the largest theoretical lower bound that you can think of and justify. Provide justification for this lower bound. You do not need to find an algorithm that satisfies this lower bound.

Example: Given an array of size n and a target t, determine the index of t in the array.

Example Solution: $\Omega(n)$, because in the worst case any algorithm must look through all n numbers to verify that the target is not one of them, taking $\Omega(n)$ time.

a) Given an array of n numbers, find the second largest number.

b) Given an array of n numbers, check to see if there are any duplicates.

c) Given an array of n integers (with $n \geq 2$), check to see if there is a pair of elements that add to be an even number.