
DSC 40B - Quiz 02

Name:

Student ID:

About the quizzes:

- Quizzes in DSC 40B are *optional* and graded pass/fail.
- A score of 70% or higher earns a “pass” and 1.5 credits toward your final grade.
- If you don’t pass, no credits are earned, but it doesn’t hurt your grade.
- You have 30 minutes to complete the quiz.
- Unfortunately, we can’t answer clarifying questions during the quiz. If you think a question has a bug or is unclear, please let us know in a private post on Campuswire after the quiz, and we’ll take it into account when grading.

Problem 1.

Let

$$f(n) = \frac{n^4 - 3n^2 + 7}{\sqrt{n} + 5} \cdot \frac{\log n}{n^2 + 10n}.$$

What is $f(n)$ in asymptotic notation, in simplest terms?

- ☐ $\Theta(n^{1.5})$
 ☐ $\Theta(n^{1.5} \log n)$
 ☐ $\Theta(n^2 \log n)$
 ☐ $\Theta(n^{2.5} \log n)$
 ☐ $\Theta(n^4 \log n)$

Problem 2.

Let $f(n) = 3n^2 \log n + 50n$. True or False: $f(n) = O(n^3)$.

- ☐ True
☐ False

Problem 3.

Consider the function

$$f(n) = \begin{cases} n^3, & \text{if } n \text{ is even;} \\ n^2, & \text{if } n \text{ is odd.} \end{cases}$$

True or False: $f(n) = \Theta(n^3)$.

- ☐ True
☐ False

Problem 4.

Suppose **bar** and **baz** are two functions. **bar** takes $\Theta(n^3)$ time on an input of size n , while **baz** takes $\Theta(n)$ time on an input of size n . What is the time complexity of **foo**, below?

```
def foo(n):
    if n < 10_000:
        bar(n)
    else:
        baz(n**2)
```

- ☐ $\Theta(1)$
 ☐ $\Theta(n)$
 ☐ $\Theta(n^2)$
 ☐ $\Theta(n^3)$
 ☐ $\Theta(n^6)$

Problem 5.

Suppose $f_1(n)$ is both $O(n^2)$ and $\Omega(n)$, and suppose $f_2(n) = \Theta(n^3)$. What are the tightest possible bounds on the product $f_1(n) \cdot f_2(n)$?

- ☐ $\Theta(n^4)$
 ☐ $\Theta(n^5)$
 ☐ $O(n^5)$ and $\Omega(n^4)$
 ☐ $O(n^6)$ and $\Omega(n^3)$

Problem 6.

Suppose Algorithm A takes $\Theta(n^2)$ time, while Algorithm B takes $\Theta(n^3)$ time. True or False: on every input of size $n = 1,000$, Algorithm A takes less time than Algorithm B.

- ☐ True
☐ False

Problem 7.

What is the **best case** time complexity of the following function?

```
def foo(arr):  
    """arr` is a list containing n numbers."""  
    n = len(arr)  
    for i in range(n):  
        if arr[i] == max(arr):  
            for j in range(n):  
                print(arr[i] - arr[j])  
            return
```

- ☐ $\Theta(1)$ ☐ $\Theta(\log n)$ ☐ $\Theta(n)$ ☐ $\Theta(n \log n)$ ☐ $\Theta(n^2)$ ☐ $\Theta(n^3)$

Problem 8.

What is the **worst case** time complexity of the same function from the previous problem, reproduced below?

```
def foo(arr):  
    """arr` is a list containing n numbers."""  
    n = len(arr)  
    for i in range(n):  
        if arr[i] == max(arr):  
            for j in range(n):  
                print(arr[i] - arr[j])  
            return
```

- ☐ $\Theta(1)$ ☐ $\Theta(\log n)$ ☐ $\Theta(n)$ ☐ $\Theta(n \log n)$ ☐ $\Theta(n^2)$ ☐ $\Theta(n^3)$

Problem 9.

Suppose you are given an array of n integers that is **sorted** in increasing order. You are tasked with determining whether every element of the array is the same (that is, all of the entries have the same value).

What is the **tight** theoretical lower bound for the worst case time complexity of this problem?

- ☐ $\Theta(1)$ ☐ $\Theta(\log n)$ ☐ $\Theta(n)$ ☐ $\Theta(n \log n)$

Problem 10.

What is the **expected** time complexity of the following function? You may assume that `math.sqrt` takes $\Theta(1)$ time.

```
import math  
import random  
def foo(n):  
    # draw a number uniformly at random from 0, 1, ..., n-1 in  $\Theta(1)$  time  
    x = random.randrange(n)  
    if x < math.sqrt(n):  
        for i in range(n**2):  
            print(i)  
    else:  
        print('Lucky!')
```

- ☐ $\Theta(1)$ ☐ $\Theta(\sqrt{n})$ ☐ $\Theta(n)$ ☐ $\Theta(n\sqrt{n})$ ☐ $\Theta(n^2)$