07 Module Quiz

- Due Oct 18 at 11:59pm
- Points 15
- Questions 10
- Available until Oct 18 at 11:59pm
- Time Limit None
- Allowed Attempts Unlimited

Instructions

- **Purpose:** To *review and reinforce your understanding* of key concepts from the week. They are designed as a low-stakes way to assess your mastery and identify topics to revisit before exams.
- **Task:** At the end of each module, complete the quiz in Canvas. You may take each quiz as many times as you like before the deadline. Each attempt may show different questions, drawn from a larger question bank. Your highest score by the deadline will be recorded.
- **Criteria:** Full credit is earned by completing at least 80% of the weekly quizzes on time. Quizzes are unproctored and open-resource, but intended for individual review. Use them to reflect on what you've learned and where you may need more practice.

This quiz was locked Oct 18 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	175 minutes	15 out of 15

ⓘ Correct answers are hidden.

Score for this attempt: 15 out of 15
Submitted Oct 18 at 8:12pm
This attempt took 175 minutes.

Question 1
1 / 1 pts
What is the **primary goal** of **algorithm analysis**?

- (●) To evaluate the time efficiency of the algorithm when a problem size grows.
- ( ) To ensure the algorithm terminates in a finite amount of time.
- ( ) To determine the correctness of the algorithm considering all test scenarios.

Question 2
1 / 1 pts

Which of the following are considered **constant-time operations** when analyzing an algorithm's time complexity with respect to a problem size **n**? *(Select all that apply)*

- [x] Comparing two references in a conditional statement (e.g., `if (someReference == null)`).
- [x] Performing a multiplication operation (e.g., `product = a * b`).
- [x] Accessing an element in an array by index (e.g., `array[i]`).
- [ ] Searching for an element in an unsorted array using a loop.

Question 3

1 / 1 pts

An algorithm always performs the following actions:

1. Create a 1-dimensional array of length N in 3 total basic operations
2. **Fill *EACH element*** of the array, using 2 basic operations per index

What is the **Big-O** notation for this algorithm, if N is the `length` of the array?

- (•) O(N)
- ( ) O(1)

Question 4

1 / 1 pts

The runtime complexity of an algorithm which performs 2*N + 1 basic operations is _____.

- ( ) quadratic
- ( ) constant
- (•) linear

Question 5

1 / 1 pts

What is meant by the "**worst case**" for an algorithm when analyzing its runtime complexity?

- ( ) The input which makes the algorithm do the ***least*** work.
- ( ) The ***most frequently provided*** input to the algorithm.
- (•) The input which makes the algorithm do the ***most*** work.

Question 6

2 / 2 pts

Which of the following algorithms has the **MOST efficient** runtime complexity **T(N)**?

- ( ) $T(N) = 2^N+200$
- ( ) $T(N) = 2N^3+50N^2+1000$
- (•) $T(N) = 100N+500$

- T(N) = $3N^2+100N+5$

Question 7

2 / 2 pts

What is the Big-O worst-case runtime complexity of the following code when the problem size N grows with no limit? Where the problem size `N` represents the number of elements in the array `candidates`.

```
public void printNames(String[] candidates) {
    for (int i = 0; i < 10; i++)
      System.out.println(i + ":" + candidates[i]);
}
```

- O(N^2)
- O(1)
- O(10^N)
- O(N)

Question 8

2 / 2 pts

Complexity can be described using a complexity class descriptor, or using the Big-O notation for that complexity class. Match each descriptor with its corresponding Big-O notation below:

Linear time algorithm

O(N)

Quadratic time algorithm

O(N^2)

Constant time algorithm

O(1)

Logarithmic time algorithm

O(log(N))

Cubic time algorithm

O(N^3)

Exponential time algorithm

O(2^N)

Question 9

2 / 2 pts

What is the runtime complexity of the following `thirdDegreePolynomialFunction()` method, assuming that the problem size is the input parameter `n`?

```
public static double thirdDegreePolynomialFunction(int a, int b, int c, int n) {
  if(a == 0 && b != 0)
    return b * Math.pow(n, 3) + c * n; // return b * n^3 + c*n
  if(a == 0 && b == 0)
    return c * Math.pow(n, 3); // return c * n^3
  return a * Math.pow(n, 3) + b * Math.pow(n, 2) + c * n; // a* n^3 + b* n^2 + c*n
}
```

- O(n^2)
- O(n)
- O(1)
- O(n^3)

Question 10

2 / 2 pts

What is the the **worst case** runtime complexity of the following `mystery()` method, assuming that the problem size n is the **length** of the input array `data`?

```
  public static boolean mystery(String[] data) {
    for (int i = 0; i < data.length; i++) {
      int j = 0;
      while (j < i) {
        if (data[j].equals(data[i])) {
          return true;
        }
        j++;
      }
    }
    return false;
  }
```

- O(1)
- O(n)

- $O(n^2)$

Quiz Score: 15 out of 15