Big O class <https://x.com/i/grok/share/MaSM25NjQkIYrXlP9xo7dcPvh>

Good web sites:

1. BigO cheat sheet: [Big-O Algorithm Complexity Cheat Sheet (Know Thy Complexities!) @ericdrowell](https://www.bigocheatsheet.com/)
2. [Big O Notation Tutorial - A Guide to Big O Analysis - GeeksforGeeks](https://www.geeksforgeeks.org/dsa/analysis-algorithms-big-o-analysis/)
3. [Examples of Big-O analysis - GeeksforGeeks](https://www.geeksforgeeks.org/dsa/examples-of-big-o-analysis/)

Quiz: Questions:

1. Multiple Choice: What does Big O notation describe?
   * A) The exact runtime of an algorithm
   * B) The upper bound of an algorithm’s runtime
   * C) The lower bound of an algorithm’s runtime
   * D) The average runtime of an algorithm
2. Multiple Choice: What is the time complexity of this code?

python

def sum\_array(arr):

total = 0

for num in arr:

total += num

return total

* + A) ( O(1) )
  + B) ( O(n) )
  + C)

O(n^2)

O(\log n)

1. Short Answer: Analyze the time complexity of the following code and explain your reasoning:

python

def find\_duplicates(arr):

seen = set()

for num in arr:

if num in seen:

return True

seen.add(num)

return False

1. True/False: An algorithm with

complexity is always slower than one with

complexity for all input sizes.

5. Wrap-Up and Q&A (5 minutes)Objective: Reinforce key concepts and address student questions.Content:

* Recap: Big O focuses on worst-case, dominant-term growth rates.
* Emphasize: Use Big O to compare algorithms, not to predict exact runtimes.
* Real-world relevance: Efficient algorithms improve performance in large-scale systems (e.g., search engines, databases).
* Answer student questions and clarify any confusing points.

Homework/Extension:

* Assign students to analyze the time complexity of a sorting algorithm (e.g., quicksort or insertion sort) and write a short explanation.
* Encourage exploring space complexity as a follow-up topic.

Question 1Consider the following Java code that calculates the sum of an array:

java

public int arraySum(int[] arr) {

int sum = 0;

for (int i = 0; i < arr.length; i++) {

sum += arr[i];

}

return sum;

}

Question: What is the Big O time complexity of this code? Explain your reasoning.

Question 2Analyze the following Java code that prints all pairs of elements in an array:

java

public void printPairs(int[] arr) {

for (int i = 0; i < arr.length; i++) {

for (int j = 0; j < arr.length; j++) {

System.out.println(arr[i] + ", " + arr[j]);

}

}

}

Question: What is the Big O time complexity of this code? Explain your reasoning.

Question 3Examine the following Java code that searches for an element in a sorted array using binary search:

java

public int binarySearch(int[] arr, int target) {

int left = 0;

int right = arr.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target) {

return mid;

}

if (arr[mid] < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1;

}

Question: What is the Big O time complexity of this code? Explain your reasoning.

Question 4Consider the following Java code that generates a multiplication table up to n:

java

public void multiplicationTable(int n) {

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

System.out.println(i + " \* " + j + " = " + (i \* j));

}

}

}

Question: What is the Big O time complexity of this code? Explain your reasoning.

Question 5Analyze the following Java code that computes the factorial of a number recursively:

java

public int factorial(int n) {

if (n == 0 || n == 1) {

return 1;

}

return n \* factorial(n - 1);

}

Question: What is the Big O time complexity of this code? Explain your reasoning.

AnswersQuestion 1 AnswerTime Complexity: O(n)  
Explanation: The code contains a single for-loop that iterates through the array of length n exactly once. Each iteration performs a constant-time operation (addition and assignment). Therefore, the time complexity is linear, O(n), where n is the length of the array.Question 2 AnswerTime Complexity: O(n²)  
Explanation: The code has two nested for-loops, each iterating n times (where n is the array length). For each iteration of the outer loop, the inner loop runs n times, resulting in n \* n = n² total iterations. The operations inside the inner loop (printing) are constant time, so the overall time complexity is O(n²).Question 3 Answe