First Virtual Challenge Meet due by 11/8/2025

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
2) Big O Notation Tutorial - A Guide to Big O Analysis - GeeksforGeeks
3) Examples of Big-O analysis - GeeksforGeeks

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(n^2)

$O(\log\ n)$O(\log n)

3. 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
```

4. True/False: An algorithm with

$$O(n^2)$$

complexity is always slower than one with

$$O(n\log\ n)$$

complexity for all input sizes.

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.

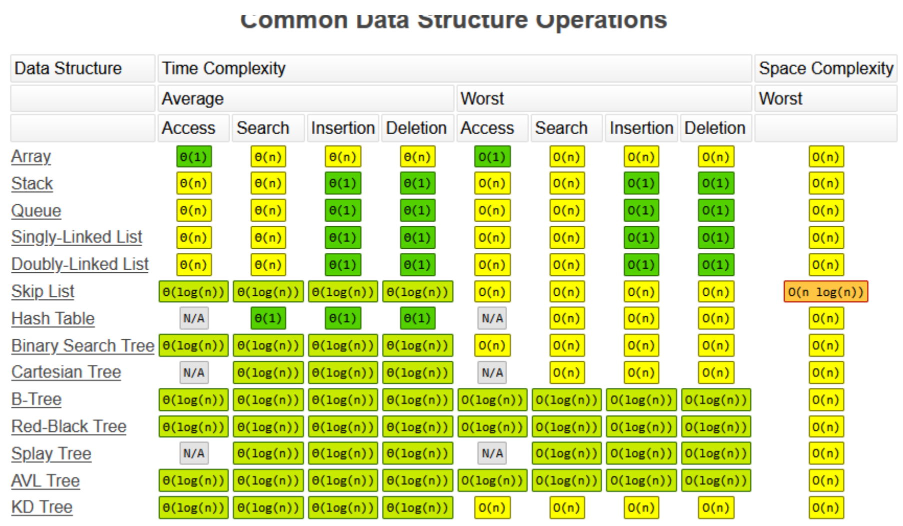
Question 5Analyze the following Java code that computes the factorial of a number recursively:

java

```
public int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

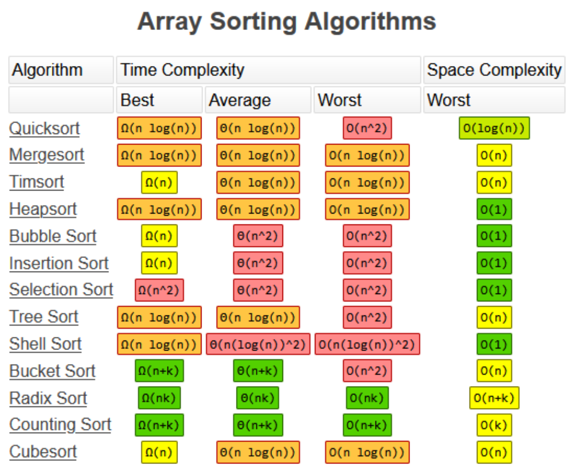
Question: What is the Big O time complexity of this code? Explain your reasoning.

Type I question: You are given the name of an algo and are asked to find average or worst case Big O



Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

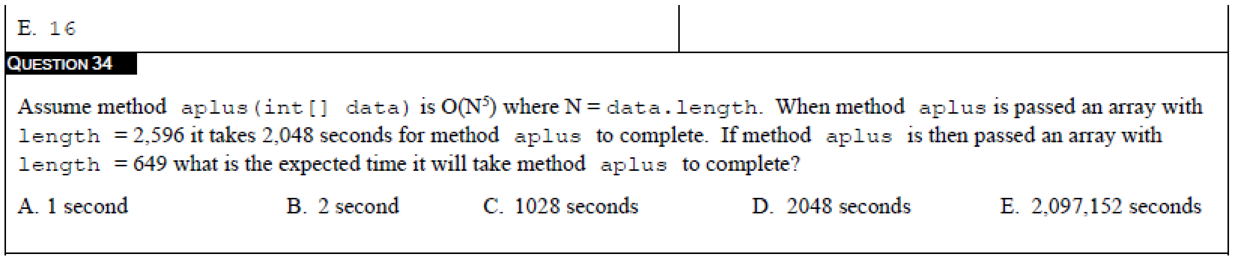


Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Type II Questions: You are given the code for an algo and are asked to find Big O.

Type III Questions: You are given runtime for an algo for a given n and its Big O and are asked to find the runtime for another n.



E. `16`

QUESTION 34

Assume method `aplus(int[] data)` is O(N^5) where N = `data.length`. When method `aplus` is passed an array with `length` = 2,596 it takes 2,048 seconds for method `aplus` to complete. If method `aplus` is then passed an array with `length` = 649 what is the expected time it will take method `aplus` to complete?

A. 1 second B. 2 second C. 1028 seconds D. 2048 seconds E. 2,097,152 seconds

A really nice video: https://www.youtube.com/watch?v=EmEroCSi95c