What is the time complexity of iterating through each element of a standard array of size n?

[redacted]

Searching for a specific element within an unsorted linked list of size *n* has what worst-case time complexity?

[redacted]

What is the time complexity of the best-case scenario for finding a target value in a binary search tree of size *n*?

[redacted]

Main Event (Time and Space Complexity)

```
def find_pairs(nums, target):
    pairs = []
    for i in range(len(nums)):
    for j in range(i + 1, len(nums)):
    if nums[i] + nums[j] == target:
        pairs.append((nums[i], nums[j]))
    return pairs
```

What is the time complexity of this function?

What is the space complexity of this function?

[redacted]

Scenario: You need to store user information where fast lookup by username is crucial. Consider these options:

- **A: Hash table**
- **B: Balanced binary search tree**
- **C: Unsorted array**
- **Which option generally offers the best time complexity for username lookups? Explain.**

[redacted]

True or False: Algorithm A with complexity O(n^3) is always slower than Algorithm B with complexity O(n^2 * log n).

[redacted]

Describe a scenario where using an array for data storage might be preferable to a linked list, even though some common operations are technically slower on arrays.

[redacted]

Consider a dynamic array implementation that starts with an initial capacity and doubles its capacity whenever it runs out of space. Analyze the following:

- The worst-case cost of a single insertion.
- The amortized cost of a series of N insertions, starting from an empty array.

[redacted]