Complexity Analysis Report for produceFlowDependencyListing Method – USEI07

1. Overview

The produceFlowDependencyListing method processes a list of articles and their operations to build and display a flow dependency map across workstations. This method performs the following main steps:

1. Collects operations associated with each article.
2. Sorts these operations by their order.
3. Builds a flow dependency map that tracks transitions between workstations.
4. Sorts and displays the workstation dependencies.

Each step involves a different level of computational complexity, and together they determine the method's overall time and spatial complexity.

Time Complexity Analysis

1: Outer Loop - Iterating Over articleList

This loop iterates once per article in articleList. Let n represent the total number of articles.
Complexity: $O(n)$

2: Inner Loop - Collecting Operations for Each Article

For each article, the method iterates over operationsScheduled, checking each operation to see if it is associated with the current article.
Since there are up to n operations in total, this nested loop results in a complexity of $O(n^2)$.
Complexity: $O(n^2)$

3: Sorting Operations for Each Article

After collecting operations, the method sorts them by their order, using a sorting algorithm.
Since sorting is performed for each article, this step adds $O(n^2 \log n)$ to the overall complexity.
Complexity: $O(n^2 \log n)$

4: Building the Flow Dependency Map

Following the sort, the method builds a flow dependency map by iterating over the list of sorted operations. This involves an $O(n)$ loop for each article.
Since we have n articles, this step contributes $O(n^2)$ to the time complexity.
Complexity: $O(n^2)$

5: Displaying Results Sorted by Dependency Count

Finally, the method iterates over flowDependencyMap, which could contain up to n unique workstations. For each workstation, it sorts its dependencies, which can involve up to O(n log n) complexity per workstation.
This step's complexity in the worst case is O(n log n).
Complexity: O(n log n)

3. Overall Complexity Summary

After analyzing each step, we observe that Step 3 (Sorting Operations for Each Article) has the highest complexity at O(n^2 log n). This term dominates the overall complexity, so the time complexity of the produceFlowDependencyListing method is: O(n^2 log n)

4. Final Complexity Analysis Summary Table

Step	Description	Complexity
1. Outer Loop	Iterates over articles	O(n)
2. Collecting Operations	Finds operations for each article	O(n^2)
3. Sorting Operations	Sorts collected operations	O(n^2 log n)
4. Building Flow Dependency Map	Tracks workstation flows	O(n^2)
5. Displaying Results	Sorts dependencies	O(n log n)

5. Conclusion on Time Complexity

The produceFlowDependencyListing method has a time complexity of O(n^2 log n), primarily driven by the sorting step for operations associated with each article. This complexity suggests that for large input sizes, the method could become computationally intensive, particularly as the number of articles (n) grows.

Spacial Complexity Analisys

The spatial complexity of the produceFlowDependencyListing method reflects the memory used to store and manipulate data as each step processes articles and operations.

1. **Outer Loop - Iterating Over articleList**
   The method iterates through articleList, requiring a constant amount of space per article.
   Spatial Complexity: $O(1)$

2. **Collecting Operations for Each Article**
   This step temporarily stores operations for each article in a collection, where each article may have up to n operations. The memory for storing these collections scales linearly with n.
   Spatial Complexity: $O(n)$

3. **Sorting Operations for Each Article**
   Sorting the operations requires temporary space proportional to the number of operations. Since sorting occurs for each article, this contributes $O(n^2)$ space in the worst case.
   Spatial Complexity: $O(n^2)$

4. **Building the Flow Dependency Map**
   The flow dependency map structure holds each workstation's dependencies. If each workstation potentially depends on every other one, the map could require up to $O(n^2)$ space to store all possible transitions.
   Spatial Complexity: $O(n^2)$

5. **Displaying Results Sorted by Dependency Count**
   This step sorts and displays the dependencies using data already stored in the map, requiring only a constant amount of extra space.
   Spatial Complexity: $O(1)$

Conclusion on Spatial Complexity

Considering each step, the most significant spatial complexity arises from the flow dependency map and the temporary storage for sorting operations and collecting dependencies. The **overall spatial complexity for the produceFlowDependencyListing method is** $O(n^2)$