Complexity Analysis Report for PrioritySimulator – USEI08

Overview

The PrioritySimulator class is designed to simulate and schedule tasks based on priority. It operates on a list of articles and a collection of workstations, assigning operations to workstations based on their availability and processing time.

Time Complexity Analysis

1. computePrioritySimulation()

Functionality

This method performs the main task of assigning articles to workstations based on priority. The steps include sorting articles, iterating over each article, and further iterating over each operation within an article to assign it to the most suitable workstation.

Complexity Analysis

- Sorting Articles: Sorting the list of articles based on priority has a time complexity of O(n log n).
- Outer Loop: The outer loop iterates over each article, resulting in O(n).
- Inner Loop: For each article, the method iterates over the list of operations. Assuming n operations in the worst case, this contributes $O(n^2)$.
- Finding the Best Workstation: The findBestWorkstationByPriority() method is called within the inner loop, each time contributing O(n log n).

Total Complexity

The total complexity for computePrioritySimulation() is: $O(n^3 \log n)$

2. findBestWorkstationByPriority()

Functionality

This method identifies the best workstation for a given operation based on availability and processing speed. It first filters workstations to find those capable of handling the operation, then selects the optimal workstation by sorting them based on their processing time and availability.

Complexity Analysis

- Filtering Workstations: Filtering through the workstations has a complexity of O(n).
- Sorting Workstations: Sorting the suitable workstations has a complexity of O(n log n).

Total Complexity

The total complexity for findBestWorkstationByPriority() is: O(n log n)

3. getSuitableWorkstations()

Functionality

This method iterates over all workstations to identify those that are capable of performing a specified operation.

Complexity Analysis

- Iterating through Workstations: This operation involves iterating through all workstations, resulting in a time complexity of O(n).

Total Complexity

The total complexity for getSuitableWorkstations() is: O(n)

4. Display Methods

Methods

These include:
- displayScheduledOperationsTable()
- displayTotalTimePerArticleType()
- timeSpentPerOperation()
- displayWorkstationOperationTimes()

Each of these methods iterates through priorityOperationsScheduled to calculate and format output data related to scheduling and timing.

Complexity Analysis

- Iterating over Scheduled Operations: Each method involves iterating over priorityOperationsScheduled, performing operations or calculations for each scheduled entry. Given that there are n scheduled operations, each of these methods has a complexity of $O(n^2)$.

Total Complexity

The total complexity for each display method is: $O(n^2)$

Summary of Complexities

Method	Complexity
computePrioritySimulation()	$O(n^3 \log n)$
findBestWorkstationByPriority()	O(n log n)
getSuitableWorkstations()	O(n)
Display methods	$O(n^2)$

Conclusion on Time complexity

The PrioritySimulator class has a primary complexity driver in the computePrioritySimulation() method, with a complexity of $O(n^3 \log n)$. This analysis suggests that the performance bottleneck lies in the iterative process of assigning articles to workstations based on priority

Spatial Complexity Analysis

1. **computePrioritySimulation()**

- Sorting Articles: Sorting the articles requires additional space, typically O(n) for storage during the sorting process.
- Outer and Inner Loops: The outer loop iterates over articles, and the inner loop iterates over operations. These loops do not require additional memory beyond the input size.
- Finding the Best Workstation: The findBestWorkstationByPriority() method, invoked within the inner loop, may require temporary space for filtered lists of workstations.

Total Spatial Complexity: The total spatial complexity of computePrioritySimulation() is O(n).

2. findBestWorkstationByPriority()

- Filtering Workstations: Filtering workstations results in a temporary list, with O(n) space required.
- Sorting Workstations: Sorting the list of workstations also requires O(n) space.

Total Spatial Complexity: The total spatial complexity for findBestWorkstationByPriority() is O(n).

3. getSuitableWorkstations()

Spatial Complexity:
- Iterating through Workstations: No additional memory is required beyond the input list of workstations.

Total Spatial Complexity: The total spatial complexity for getSuitableWorkstations() is O(1).

4. Display Methods

Methods:
- displayScheduledOperationsTable()
- displayTotalTimePerArticleType()
- timeSpentPerOperation()
- displayWorkstationOperationTimes()

These methods store temporary variables during iteration but do not require significant additional memory beyond the input data.

Total Spatial Complexity:
The spatial complexity for each display method is O(1).

Summary of Spatial Complexities

Method	Spatial Complexity
computePrioritySimulation()	O(n)
findBestWorkstationByPriority()	O(n)
getSuitableWorkstations()	O(1)
Display Methods	O(1)