Heap Data Structures: Implementation, Analysis, and Applications

# Abstract

This report explores the implementation and analysis of heap data structures, specifically focusing on Heapsort and priority queue operations. We present Python implementations, analyze their time and space complexity, and compare Heapsort with other popular sorting algorithms. A real-world application of priority queues is also demonstrated using a task scheduler model.

# Heapsort Implementation and Analysis

## 1. Implementation

Heapsort is implemented using a max-heap. The algorithm first builds a max-heap from the input list and then extracts elements from the heap one by one, maintaining the heap property throughout.

## 2. Time and Space Complexity

The time complexity of Heapsort is O(n log n) in the worst, average, and best cases. This is because building a heap takes O(n) time, and each of the n elements is extracted in O(log n) time. Space complexity is O(1) as it is an in-place sorting algorithm.

## 3. Comparison with Quicksort and Mergesort

Empirical comparisons were performed using randomly generated, sorted, and reverse-sorted lists. Heapsort showed consistent performance across all distributions, while Quicksort performed best on random inputs and Mergesort was stable across all.

# Priority Queue Implementation and Applications

## Part A: Implementation

The priority queue is implemented using a binary heap stored in a Python list. A Task class encapsulates task metadata such as ID, priority, arrival time, and deadline. A min-heap is used for task scheduling.

## Core Operations

The queue supports insert (O(log n)), extract\_min (O(log n)), decrease\_key (O(log n)), and is\_empty (O(1)) operations. Each operation maintains the heap property to ensure correct priority ordering.

# Conclusion

Heaps are fundamental data structures that support efficient priority-based operations. Heapsort offers a reliable O(n log n) sorting method, while priority queues are essential for scheduling applications. This assignment provided hands-on experience with both and validated their theoretical performance.