**Assignment 4: Heap Data Structures: Implementation, Analysis, and Applications**

Sih Chi Wang

University of the Cumberlands

Fall - Algorithms and Data Structures (MSCS-532-A01) - First Bi-term

Dr. Vanessa Cooper

September 15, 2024

**Overview**

This study discusses the implementation, analysis, and comparison of heap data structures, with a particular emphasis on the use of Heapsort and Priority Queue operations. Python is used to carry out the implementation, which is then followed by carrying out an in-depth investigation of the effectiveness of the algorithms and the practical applications of those algorithms.

**Heapsort Implementation and Analysis**

**Implementation**

The implementation of the Heapsort algorithm is carried out using Python. Included in the processes are the construction of a max-heap, the sorting of the array by the recurrent extraction of the maximum element, and the maintenance of the heap attribute.

**Python Implementation**

A screen shot of a computer

Description automatically generated

**Analysis of Implementation**

**Space Complexity**

Due to the fact that it sorts in situ without needing extra storage that is proportionate to the size of the input, the space complexity of the heapsort algorithm is O(1). It is important to note that the space complexity of the heap data structure itself is O(n), where n is the number of items that are included inside the heap.

**Comparison**

We carried out empirical experiments on a variety of input sizes and distributions in order to evaluate the performance of Heapsort in comparison to Quicksort and Merge Sort. When compared to Quicksort, the performance of heapsort is generally superior across all distributions; nevertheless, it may be slower in typical circumstances.

Quicksort is often quicker in most circumstances owing to its superior performance on average. However, if optimizations such as random pivots are not performed, its worst-case performance might deteriorate to O(n^2), which is known as the worst-case scenario. The Merge Sort algorithm consistently achieves a performance level of O(n log n) in both the average and worst circumstances, while having a higher space complexity rating of O(n).

Observations

The performance of heapsort is steady at O(n log n), although there is a possibility that it will be slower than Quicksort since it does fewer comparisons than Quicksort does. In fact, Quicksort is often quicker than other methods, yet it might have bad performance in the worst way possible. It is possible to get steady performance using Merge Sort, but it comes with increased space overhead.

**Priority Queue Implementation and Applications**

**Part A: Priority Queue Implementation**

**1. Data Structure**

We use an array to represent the binary heap for simplicity and efficiency. The choice of array is justified by its straightforward implementation and the efficient access it provides for heap operations.

Python Implementation

