# Heap Data Structures

Name

Institution Affiliation

# Heap Data Structures

Heapsort Implementation and Analysis

Implementation

Heapsort was implemented using a max-heap represented as an array. The following are the three main steps followed by the algorithm.

Building the max-heap.

Extracting the maximum element.

Maintaining the heap property after extraction.

Time Complexity Analysis

The time complexity for the process of building a max heap is O(n) because each element in the array is heapified only once. Each of the n elements requires O(log n) time during the extraction phase to maintain the heap property. In terms of overall time complexity, the heapsort is operating with O(n log n) time complexity for best, worst, and average cases. Lastly, for space complexity, the Heapsort requires O(1) auxiliary space, considering that all operations are performed in place. This makes it more space-efficient compared to other algorithms like Merge Sort.

Results comparison

Heapsort was compared with Quicksort, and Merge Sort understands its performance on different array sizes and distributions (Marcellino et al., 2021). On the random array, the quick sort was the best in terms of time efficiency, mostly because of the efficient petitioning strategy. However, the heapsort and merge sort showed consistent performance. On the sorted and reverse-sorted arrays, Heapsort and merge sort are able to maintain O(n log n) performance, but quicksort was slower because of its O(n^2) worst-case behavior.

What we can observe is that heapsort is more reliable when dealing with unpredictable input patterns, as indicated by its consistent performance in all input distributions.

Priority Queue Implementation and Applications

Max-heap was used to implement the priority queue in order to improve the efficiency of task management, where the tasks were represented as task objects (Bai et al., 2015). Some of the properties of these objects include task\_id, priority, arrival\_time, and deadline.

Core operations were insert for adding new tasks to the heap in O(log n), extract max for removing and returning the highest-priority task in O(log n), and is empty for checking if the heap was empty in O(1).

Test Results

The results indicate that the tasks were extracted in descending priority order, which indicates that the max heap implementation was correct and efficient. The priority queue was also able to handle task prioritization.

Testing Sorting Algorithms...

Size: 10, Random - Heap: 0.0000s, Quick: 0.0000s, Merge: 0.0000s

Size: 10, Sorted - Heap: 0.0000s, Quick: 0.0000s, Merge: 0.0000s

Size: 10, Reverse-Sorted - Heap: 0.0000s, Quick: 0.0000s, Merge: 0.0000s

Size: 100, Random - Heap: 0.0000s, Quick: 0.0000s, Merge: 0.0156s

Size: 100, Sorted - Heap: 0.0000s, Quick: 0.0000s, Merge: 0.0000s

Size: 100, Reverse-Sorted - Heap: 0.0000s, Quick: 0.0000s, Merge: 0.0156s

Size: 1000, Random - Heap: 0.0313s, Quick: 0.0213s, Merge: 0.0234s

Size: 1000, Sorted - Heap: 0.0285s, Quick: 0.0110s, Merge: 0.0155s

Size: 1000, Reverse-Sorted - Heap: 0.0104s, Quick: 0.0156s, Merge: 0.0156s

Size: 5000, Random - Heap: 0.2197s, Quick: 0.0781s, Merge: 0.1344s

Size: 5000, Sorted - Heap: 0.1922s, Quick: 0.0468s, Merge: 0.1345s

Size: 5000, Reverse-Sorted - Heap: 0.1922s, Quick: 0.0674s, Merge: 0.1249s

Testing Priority Queue...

Priority Queue (Max-Heap) Operations:

Tasks extracted in priority order:

Task ID: 2, Priority: 3

Task ID: 0, Priority: 4

Task ID: 1, Priority: 5

Task ID: 5, Priority: 13

Task ID: 4, Priority: 16

Task ID: 7, Priority: 70

Task ID: 6, Priority: 75

Task ID: 8, Priority: 78

Task ID: 9, Priority: 90

Task ID: 3, Priority: 98

# References

Bai, Y., Ahmed, S. Z., & Granado, B. (2015). ARC 2014: Towards a Fast FPGA Implementation of a Heap-Based Priority Queue for Image Coding Using a Parallel Index-Aware Tree. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, *9*(1), 1-16.

Marcellino, M., Pratama, D. W., Suntiarko, S. S., & Margi, K. (2021, October). Comparative of advanced sorting algorithms (quick sort, heap sort, merge sort, intro sort, radix sort) based on time and memory usage. In *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)* (Vol. 1, pp. 154-160). IEEE.