Report on the Project: Max-Heap (Student Evaluation by Bakdaulet)

1. Algorithm Description and Implementation

In this project, the Max-Heap data structure was implemented. It allows efficient extraction of the maximum element from a dataset. The project includes the following core operations: - Insert (insert): adds an element while maintaining the heap property. - Extract Maximum (extractMax): removes the largest element and restores the heap. - Increase Key (increaseKey): increases the value of a node and restores heap order. - Build Max-Heap (buildMaxHeap): constructs a valid Max-Heap from an unordered array.

The implementation includes performance tracking for comparisons, swaps, execution time, and memory usage.

2. Complexity Analysis

Time Complexity: - insert(value): O(log n) - extractMax(): O(log n) - increaseKey(index, newValue): O(log n) - buildMaxHeap(): O(n)

Space Complexity: - O(n)

3. Code Quality Overview

The code is clean, readable, and follows modular design principles. Proper error handling is included, ensuring stability under invalid operations. The codebase is well-documented and logically organized into packages: algorithms, metrics, cli.

4. Performance and Metrics

Performance was measured on datasets of size 100, 1000, 10000, and 100000. Metrics include the number of comparisons, swaps, and execution time.

Results: - Execution time grows logarithmically with input size. - Comparisons and swaps increase slowly as input grows. - Max-Heap performs consistently even with large datasets.

Charts and Reports: - Metrics were stored in a CSV file. - Based on this data, a chart (Graphic.png) was generated to visualize time vs input size.

5. Recommendations

1. Memory optimization through dynamic array resizing with a calculated growth factor. 2. Add logging for heapify call counts. 3. Implement interactive visualization for educational use. 4. Test on edge cases such as arrays with identical or sorted elements.

6. Conclusion

The Max-Heap project is correctly and efficiently implemented. The structure performs well and meets complexity expectations. The code is modular, extendable, and maintains theoretical performance guarantees. This project successfully demonstrates strong algorithmic understanding and software design discipline.