Determining the Optimal k Value for a Hybrid Sorting Algorithm

Introduction

The purpose of this report is to provide a detailed description of the process involved in finding the optimal value of k for a hybrid sorting algorithm that combines merge sort and binary insertion sort. The optimal k value allows the algorithm to efficiently balance the performance benefits of both sorting methods, resulting in improved overall sorting performance.

To begin the process, we first consider the strengths and weaknesses of each algorithm. Merge sort is known for its ability to handle large datasets and for its speed. However, it can become inefficient when dealing with smaller datasets. On the other hand, binary insertion sort is faster and more efficient when sorting smaller datasets.

The hybrid sorting algorithm we are using switches from merge sort to binary insertion sort when the size of the dataset falls below a certain threshold, represented by the value of k. The goal of this report is to find the optimal value of k that will allow us to take full advantage of the strengths of both algorithms and achieve better performance.

> ⚠️ **Note on Data Collection**
>
> Please be advised that the data presented in this report was collected on a 16-inch, 2021 MacBook Pro with an Apple M1 Pro processor and 16 GB of Memory, running macOS Ventura 13.2.1 and virtualising Ubuntu 22.10. It is important to note that the performance results may vary depending on the system and configurations. Additionally, the algorithm runs were performed single-threaded.

Methodology

1. A sorting function was implemented that accepts an array of data, its size, and a k value as parameters. This function uses merge sort for sorting larger subarrays and switches to binary insertion sort for subarrays of size k or smaller.
2. To find the optimal k value, the sorting time and data move count were measured for k values ranging from 1 to 100. For each candidate k value, a copy of the original dataset was made and the hybrid sorting function was used to sort the data.
3. The sorting time and data move count were measured for each candidate k value. The sorting time was recorded using the system clock, while the data move count was captured by modifying the sorting algorithm to track these operations.
4. For each iteration, the sorting time and data move count were compared with the best values observed so far. If a new best k value was found for sorting time or data move count, the respective best k value was updated. This process was repeated until all k values in the range were evaluated.
5. Finally, the best k values for sorting time and data move count were determined separately and reported.

Results

After performing multiple iterations and examining various candidate k values, we found the optimal k values for each criterion:

Field	**Fastest k value**	Sorting time (s)
Field1	20	6.168160
Field2	22	3.235541
Field3	21	3.742201

Field	**Optimal k value**	**Data move count**
Field1	10	920,930,930
Field2	10	921,001,735
Field3	10	921,005,812

Conclusion

This report detailed the methodology used to find the optimal k value for a hybrid sorting algorithm that combines merge sort and binary insertion sort. The optimal k value allows the algorithm to efficiently balance the performance benefits of both sorting methods, resulting in improved overall sorting performance. The results show separate optimal k values for sorting time and data move count, which can be chosen based on specific performance requirements.