# Introduction

This project involves working on a dataset of 10,000 films, implementing and analysing sorting and searching algorithms, as well as measuring their time complexity.

# Sorting Algorithm

**Algorithm Chosen: Merge Sort**

The following table compares the most common sorting methods.

A table with text and numbers

AI-generated content may be incorrect.

Figure 1 - Comparison of Sorting Algorithms (Code Project, 2021)

Merge Sort was chosen because it is an efficient and stable sorting method, especially for large datasets, which is the case. Its time complexity remains O(n log n) for best, average and worst case scenarios. Therefore, it seemed to be the best option.

Films were sorted by Title. If two or more films had the same title, their filmID was used as a secondary criterion for the sorting.

# Sorting Time Complexity Analysis

The Merge Sort algorithm was tested with datasets of the following sizes: 10, 100, 1,000, 5,000 and 10,000 films. Each size was tested three times and the average time (in nanoseconds) was calculated.

**Merge Sort Results:**

|  |  |
| --- | --- |
| **Number of Records** | **Average Time (ns)** |
| 10 | 1041.0 |
| 100 | 7736.0 |
| 1000 | 76805.0 |
| 5000 | 374972.0 |
| 10000 | 976861.0 |

# Searching Algorithm

**Binary Search Results:**

|  |  |
| --- | --- |
| **Number of Records** | **Average Time (ns)** |
| 10 | 6583.0 |
| 100 | 4166.0 |
| 1000 | 4111.0 |
| 5000 | 6875.0 |
| 10000 | 7194.0 |

# Search Time Complexity Analysis