Data Tables

Data table for selection sort by GPA

Number of Students	Time Taken (ms)
100	0
500	1
1000	3
5000	20
10000	46
50000	524
100000	1298

Data table for merge sort by GPA

Number of Students	Time Taken (ms)
100	0

500	0
1000	0
5000	1
10000	2
50000	9
100000	16

Data table for sequential search by name

Number of Students	Time Taken (ms)
100	0
500	0
1000	0
5000	0

10000	1
50000	6
100000	13

Data table for binary search by age

Number of Students	Time Taken (ms)
100	0
500	0
1000	0
5000	0
10000	0
50000	1
100000	2

Discussion of Algorithm Performance

Selection sort by GPA

The selection sort algorithm sorts an array by repeatedly finding the minimum element from the unsorted part of the array and swapping it with the first element of the unsorted part. O(n^2) in the worst case.

The data table for selection sort by GPA shows that the algorithm performs reasonably well for small arrays (up to 1000 elements), but its performance declines quickly for larger arrays. Selection sort is not suitable for sorting large arrays in real-world applications or anything that is very large in size.

Merge sort by GPA

The merge sort algorithm sorts an array by dividing it into two halves, recursively sorting each half, and then merging the sorted halves. merge sort is O(n log n) in the worst case.

The data table for merge sort by GPA shows that the algorithm performs very well for all array sizes. For example, sorting an array of 100000 elements takes around 16 milliseconds. Therefore, merge sort is a good choice for sorting large arrays in real-world applications.

Sequential search by name

The sequential search algorithm searches an array by checking each element until the target element is found or the end of the array is reached. The time complexity of the sequential search is O(n) in the worst case.

The data table for sequential search by name shows that the algorithm performs very well for all array sizes. Therefore, sequential search is a good choice for searching arrays of any size in real-world applications.

Binary search by age

The binary search algorithm searches a sorted array by repeatedly dividing the search interval in half until the target element is found or the search interval is empty. Binary search is O(log n) in the worst case.

The data table for binary search by age shows that the algorithm performs very well for all array sizes, with search times of less than 2 milliseconds even for arrays of largest size (100000 elements).Binary search is a good choice for searching sorted arrays of any size in real-world applications.

Comparison of algorithms

From the data tables, we can see that merge sort is the fastest sorting algorithm, with very good performance for all array sizes. Selection sort, on the other hand, has poor performance for larger arrays and is not suitable for real-world applications. Therefore, merge sort is a better choice for sorting arrays of any size.

For searching, both sequential search and binary search have very good performance for all array sizes. However, binary search has a lower time complexity than sequential search, making it more suitable for searching large arrays. Additionally, binary search requires the array to be sorted, but sorting an array once can be more efficient than searching the same array multiple times using sequential search. Therefore, binary search is a better choice for searching sorted arrays of any size.

Overall, the choice of algorithm depends on the specific application and the size of the data. For small data sets, any algorithm may be suitable, but for larger data sets. In general, merge sort and binary search are good choices for sorting and searching, respectively, for large data sets.