CSE 222
Data Structures
Report

Yavuz Selim İkizler
1901042617

Time Complexity Analysis

Merge Sort:

The best case scenario is when the input array is already sorted or has sorted runs. In this case, the merge sort algorithm takes O (n) time to merge the subarrays and O (log n) levels of recursion. Therefore, the overall time complexity is O (n log n)

The worst case scenario is when the input array is in reverse order or has alternating elements. In this case, the merge sort algorithm takes O (n) time to merge the subarrays and O (log n) levels of recursion. Therefore, the overall time complexity is also O (n log n).

The average case scenario is when the input array is randomly ordered and does not have any sorted runs. In this case, the merge sort algorithm takes O (n) time to merge the subarrays and O (log n) levels of recursion. Therefore, the overall time complexity is also O (n log n).

For merge sort algorithm the input does not effetcs the time complexity. İt effects only the operation number.

Selection Sort:

For selection sort average , best case scenerio and worst case scenerio does not depends on the input because it always must go trough every element of the list to find smallest or largerst element.Since it search the element in nested loop it takes O(n^2) time.

Insertion Sort:

Insertion sort’s time complexity depends on the input it puts the sorted elements in order one by one if the input is already sorted it takes linear time O(n) don’t need to compare the other elements . Otherwise for the worst and average case it takes O(n^2) time since it search the list through nested loop.

Bubble Sort:

For the best case scenerio buble sort pass through all the elements if its already sorted.It does not do any swap operation therefore it takes linear time (O(n)) however for for worst case the algorithm needs to make n-1 passes through the array and perform n-i-1 swaps in each pass and for average case n/2 swap needs, where i is the current pass number. Therefore, the overall time complexity is O (n^2),

Quick Sort:

Best-case performance for this strategy would be achieved by having the 'count' of the final item of the array (pivot) as the median of the counts. As quicksort works best when it partitions the array in the middle, getting the median is the best option.In this case the algorithm takes O(nlog(n)) time.

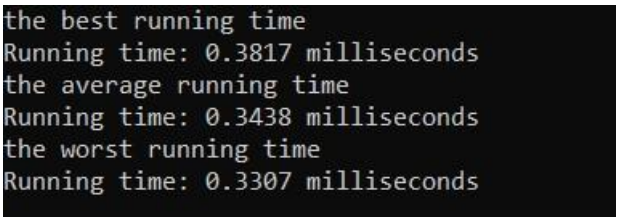
The worst case scenario is when the pivot element chosen in each partition step is the smallest or the largest element of the subarray so the list is sorted in descented or ascenting order. In this case, the algorithm divides the array into one subarray of size n-1 and another subarray of size 0 and recurses on the larger subarray. Therefore, the overall time complexity is O (n^2), where n is the number of elements in the array.

The average case scenario is when the pivot element chosen in each partition step is close to the median of the subarray. In this case, the algorithm divides the array into two subarrays of size n/2 and recurses on each subarray. Therefore, the overall time complexity is also O (n log n), where n is the number of elements in the array.

Running Time

Merge Sort:

Selection Sort:



Insertion Sort:

```
the best running time
Running time: 0.345 milliseconds
the average running time
Running time: 0.4836 milliseconds
the worst running time
Running time: 0.5789 milliseconds
```

Bubble Sort:

```
the best running time
Running time: 0.311 milliseconds
the average running time
Running time: 0.6229 milliseconds
the worst running time
Running time: 0.5162 milliseconds
```

Quick Sort:

```
the best running time
Running time: 0.2978 milliseconds
the average running time
Running time: 0.3256 milliseconds
the worst running time
Running time: 0.7256 milliseconds
```

Algorithm Comparison

According to the analysis results mentioned above, in terms of time complexity, Insertion Sort and Bubble Sort are suitable for small datasets that are already close to being sorted. However, they become slow for large datasets and unstable datasets. Selection Sort is generally a slow algorithm in terms of time complexity and is not suitable for large datasets. On the other hand, QuickSort and MergeSort are better algorithms in terms of time complexity compared to the others. They are suitable for unordered datasets. Comparing QuickSort and MergeSort, MergeSort is a more stable algorithm and suitable for all types of datasets. However, QuickSort can be slower for datasets that are already close to being sorted and is not as stable as MergeSort.

D:

In the quicksort algorithm, the process of partitioning is a key step in dividing the input list into two sublists. During partitioning, a pivot element is selected from the list, and the other elements are rearranged such that elements smaller than the pivot are placed before it, and elements greater than the pivot are placed after it. This step involves comparing each element with the pivot and swapping them if necessary.

However, quicksort does not perform a strict comparison during the partitioning process. It only compares elements with the pivot to determine their relative ordering, without considering their original positions in the list. As a result, when there are multiple elements with equal keys, their relative order may change after the partitioning step.