Program Structures and Algorithms

Spring 2024

NAME: Kartikey Vijayakumar Hebbar

NUID: 002276938

GITHUB LINK: https://github.com/kartikeyhebbar/INFO6205/tree/Spring2024/src

**Task: Assignment 6 (Hits as time predictor)**

**Relationship Conclusion:**

In most scenarios, Merge Sort tends to perform the greatest number of comparisons, with Quick Sort coming in next, followed by Heap Sort. Yet, as the dataset size grows, the gap in the number of comparisons between Merge Sort and Quick Sort narrows down.

When it comes to the frequency of swapping operations, Heap Sort stands out as the algorithm requiring the most swaps, with Quick Sort in the middle, and Merge Sort needing the fewest. Moreover, the disparity in swap operations between Heap Sort and Quick Sort becomes more pronounced with larger datasets.

Merge Sort uniquely employs the technique of combining two subsets during its merging phase, a process often referred to as hits or merges. This count escalates with the dataset's size, directly correlating to the merger operations needed.

Neither Heap Sort nor Quick Sort incorporate the hits mechanism in their sorting process.

The strategies of Merge Sort and Heap Sort include utilizing copies for the creation of temporary storage arrays throughout the sorting procedure. Among these, Merge Sort demands a more substantial number of copies compared to Heap Sort.

In examining Merge Sort, it's noted that the adjustment in performance times is minimal even as the dataset expands, showing a largely stable trend. Contrastingly, Heap Sort exhibits a more pronounced escalation in performance times with larger datasets. Quick Sort also shows an uptick in performance times as the dataset size increases, albeit at a milder rate compared to Heap Sort.

For Merge Sort, a rise in the metrics of hits, copies, swaps, and comparisons is observed alongside an expansion in dataset size, while the metric of fixes shows little fluctuation. In the case of Heap Sort, a significant surge in the swap metric is notable with increasing dataset sizes, whereas hits, copies, fixes, and comparisons maintain a steady count. With Quick Sort, an increase in both swaps and comparisons is observed as the dataset grows, while hits, copies, and fixes show negligible change.

These patterns suggest that swaps are a key indicator of performance time in Heap Sort, while comparisons serve as a main predictor for Quick Sort. Identifying a primary predictor for Merge Sort proves more challenging due to the even distribution of metrics.

In scenarios involving smaller datasets ranging from 10 to 40 elements, Merge Sort and Quick Sort demonstrate comparable efficiency, with Quick Sort slightly outperforming in metrics such as hits, copies, swaps, and fixes. Heap Sort, however, lags behind, showcasing inferior performance across these metrics.

As dataset sizes increase to between 80 and 1280 elements, Merge Sort and Quick Sort maintain similar efficiency levels, with Quick Sort marginally surpassing Merge Sort. Heap Sort's performance remains less favorable, with significantly higher metrics for copies, swaps, fixes, and comparisons.

Given these observations, Quick Sort emerges as the superior sorting algorithm for handling arrays of diverse sizes, consistently recording lower metrics across hits, copies, swaps, and fixes. Its performance times, both normalized and log-normalized, also surpass those of the other two algorithms, confirming its efficiency.

**Evidence to support that conclusion:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sort Method** | **Number of Elements** | **Hits** | **Copies** | **Swaps** | **Fixes** | **Compares** | **Normalized Time** | **log Normalized Time** | **Raw Time** | **log Number of Elements** |
| Merge Sort | 10 | 95 | 26 | 10 | 23 | 24 | 2.37 | 0.37474834 | 0 | 1 |
| Merge Sort | 20 | 266 | 80 | 20 | 95 | 66 | 2.2 | 0.34242268 | 0 | 1.301029996 |
| Merge Sort | 40 | 687 | 241 | 40 | 391 | 170 | 2.1 | 0.32221929 | 0 | 1.602059991 |
| Merge Sort | 80 | 1688 | 643 | 80 | 1580 | 417 | 2.11 | 0.32428245 | 0.01 | 1.903089987 |
| Merge Sort | 160 | 4017 | 1602 | 160 | 6361 | 992 | 2.14 | 0.33041377 | 0.01 | 2.204119983 |
| Merge Sort | 320 | 9306 | 3842 | 320 | 25522 | 2302 | 2.18 | 0.33845649 | 0.03 | 2.505149978 |
| Merge Sort | 640 | 21181 | 8962 | 640 | 102214 | 5343 | 2.22 | 0.34635297 | 0.07 | 2.806179974 |
| Merge Sort | 1280 | 47474 | 20481 | 1280 | 409225 | 11764 | 2.3 | 0.36172783 | 0.16 | 3.10720997 |
| Heap Sort | 10 | 185 | 0 | 28 | 47 | 39 | 2.86 | 0.45636603 | 0 | 1 |
| Heap Sort | 20 | 517 | 0 | 72 | 231 | 115 | 2.98 | 0.47421626 | 0 | 1.301029996 |
| Heap Sort | 40 | 1339 | 0 | 182 | 1028 | 306 | 2.92 | 0.46538285 | 0 | 1.602059991 |
| Heap Sort | 80 | 3301 | 0 | 441 | 4409 | 770 | 3 | 0.47712125 | 0.01 | 1.903089987 |
| Heap Sort | 160 | 7867 | 0 | 1037 | 18353 | 1859 | 3.04 | 0.48287358 | 0.02 | 2.204119983 |
| Heap Sort | 320 | 18281 | 0 | 2390 | 75148 | 4356 | 3.18 | 0.50242712 | 0.04 | 2.505149978 |
| Heap Sort | 640 | 41646 | 0 | 5417 | 304692 | 9990 | 3.12 | 0.49415459 | 0.1 | 2.806179974 |
| Heap Sort | 1280 | 93514 | 0 | 12109 | 1228378 | 22540 | 3.24 | 0.51054501 | 0.22 | 3.10720997 |
| Quick Sort | 10 | 84 | 0 | 15 | 19 | 25 | 2.7 | 0.43136376 | 0 | 1 |
| Quick Sort | 20 | 226 | 0 | 38 | 88 | 72 | 2.76 | 0.44090908 | 0 | 1.301029996 |
| Quick Sort | 40 | 580 | 0 | 97 | 389 | 192 | 2.43 | 0.38560627 | 0 | 1.602059991 |
| Quick Sort | 80 | 1430 | 0 | 235 | 1653 | 487 | 2.45 | 0.38916608 | 0.01 | 1.903089987 |
| Quick Sort | 160 | 3414 | 0 | 554 | 6869 | 1186 | 2.46 | 0.3909351 | 0.01 | 2.204119983 |
| Quick Sort | 320 | 7955 | 0 | 1279 | 28111 | 2804 | 2.16 | 0.33445375 | 0.03 | 2.505149978 |
| Quick Sort | 640 | 18173 | 0 | 2907 | 113935 | 6484 | 2.56 | 0.40823996 | 0.08 | 2.806179974 |
| Quick Sort | 1280 | 40894 | 0 | 6507 | 458729 | 14730 | 2.64 | 0.42160392 | 0.18 | 3.10720997 |

A graph with lines and numbers

Description automatically generated

A graph showing a line of different colors

Description automatically generated with medium confidence

A graph showing the number of elements

Description automatically generated

A graph showing the growth of a graph

Description automatically generated

A graph showing the number of elements

Description automatically generated

A graph showing the number of elements and different numbers

Description automatically generated