Report Lab 2

LE Quoc Hung

08/10/2024
C Programming and Unix

Introduction

In lab 1, we already compared three slow sorting algorithms: bubble sort, insertion sort, and selection sort. Today, we study two fast sorting algorithms, quick sort and heap sort. Both have a common feature: the recursion. Which of the two is better? Why does the recursion help the two algorithms run faster than the slow ones? We will conduct some experiments with the programming language C.

Methodology

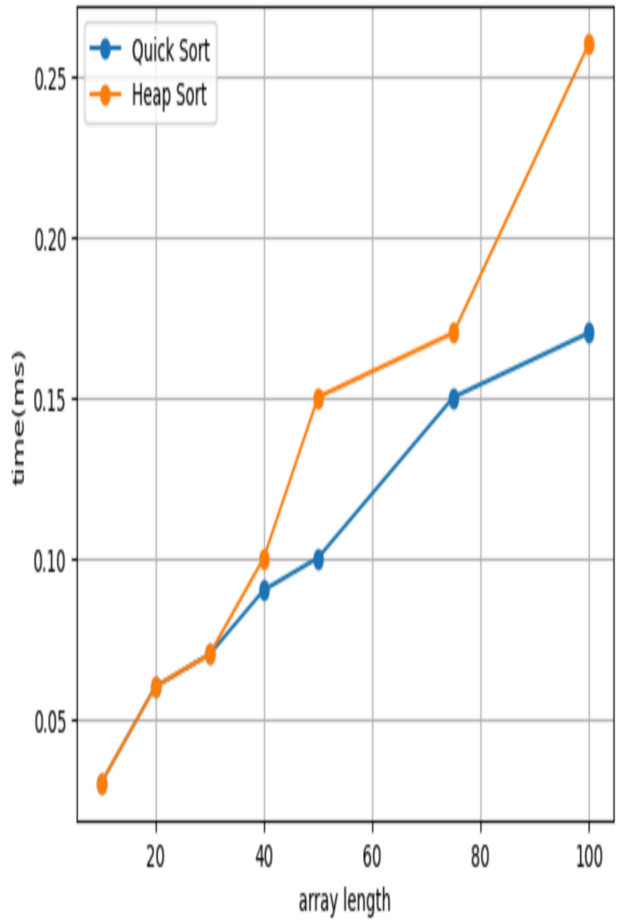
We compile the C file named **sorting.c** (after being updated) by gcc, change 3 lines consequently, and see the results.

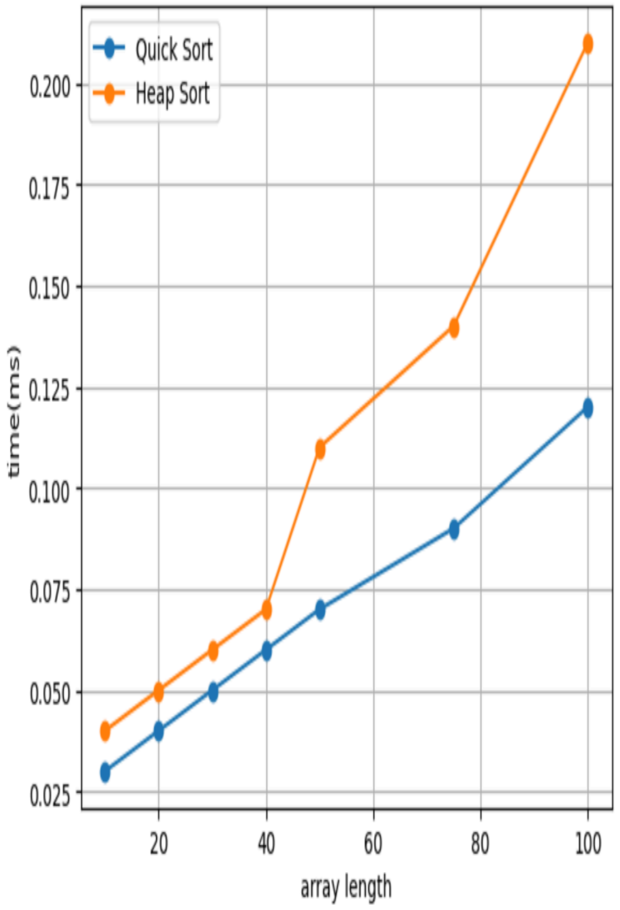
Link to my Github: https://github.com/hunglqdz/c-unix

Results

For array filled with random numbers (left figure):

Array length	10	20	30	40	50	75	100
Quick Sort	0.03	0.06	0.07	0.09	0.1	0.15	0.17
Heap Sort	0.03	0.06	0.07	0.1	0.15	0.17	0.26





For array filled with random numbers but in reverse (descending) order (right figure):

Array length	10	20	30	40	50	75	100
Quick Sort	0.03	0.04	0.05	0.06	0.07	0.09	0.12
Heap Sort	0.04	0.05	0.06	0.07	0.11	0.14	0.21

Conclusion

In both cases (in terms of the order of random numbers), the quick sort algorithm has lower execution time than the heap sort algorithm. One more thing, while three slow sorting algorithms have time complexity **$O(n^2)$**, the fast ones have time complexity **$O(n\log_2(n))$**. The reason why the recursion helps the two algorithms run faster is that it breaks down the array into smaller sub-arrays. The problem size with each recursive call is then reduced.