CMPSC 462: Data Structures and Algorithms (Fall 2022)

Project-2: Searching & Sorting

Hoang Nguyen
Section 001

Instructor
Dr. Vinayak Elangovan, Ph.D.
Department of Computer Science

Submitted On: 09/29/2022

TABLE OF CONTENTS

1. INTRODUCTION

The goal of this project is to perform search and sorting algorithms on an array/list and perform Time Complexity analysis on each function.

In this project I calculate the time elapses when the function run by creating a decorator using the perf_counter function of the time module

Note: size of 50,000 array take way too long so I only uses array size of 10,000.

2. BACKGROUND

Linear_search: this function ran through the array and in worse case it run until the end of the array which make it O(n) time complexity.

Binary_search: this function reduces the search size by half each iteration which would make the time complexity of O(logn)

Max_search: this function is like linear_search which iterate through the array one by one, so worst case it is O(n) time complexity

Min_search: this function works the same as Max_search so it is O(n) time complexity

Distinct_search: this funtion still iterate through the whole array one by one and therefore it's time complexity is O(n)

Bubble_sort: this function uses nested loops one loop through the whole array and another outside loop which loop until the array is sorted therefore it is n*n time complexity which is O(n^2)

Insertion_sort: this function also uses two nested loops, with each one of them loops through the entire length of the array, so it's time complexity is O(n^2)

Selection_sort: This function also utilize nested loops which loop through the entire array therefore it's time complexity is O(n^2)

Merge_sort: this function uses 3 loops consecutively as well as having two recersive call each time however it's input is reduced by half each recersive call, therefore it have the time complexity of O(nlogn)

3. SAMPLE OUTPUTS

```
SEARCHING

linear_search(pool, target = 161): time = 1.8994000129168853e-05 | index = 457
binary_search(sorted_pool, target = 161): time = 2.9749999157502316e-06 | index = 1630
max_search(pool): time = 0.00018693299989536172 | max = 1000
min_search(pool): time = 0.0001810809999369667 | min = 1
distinct_search(pool): time = 0.02249396200022602 | is_distinct = True | amount of dupes = 9000

SORTING

selection_sort(pool): time = 2.2967918900003497
insertion_sort(pool): time = 3.966418614999384
bubble_sort(pool): time = 8.289750423999976
merge_sort(pool): time = 0.11460706599973491

Final Tabulate Result:
time_linear = 8.54533336678287e-05
time_binary = 3.2263333196169697e-06
time_max = 0.00019163433353242
time_min = 0.0001812926666389103
time_distinct = 0.022699524666677462
time_selection = 2.3416229046664134
time_insertion = 4.072363446000054
time_bubble = 8.551475928999935
time_merge = 0.11500434333326363
```

for distinct search, the amount of duplicate number are too large therefore I only printed out the length of the amount of duplicates instead of everything single duplicate, maxium number is 1000 and the length of the pool is 10,000 therefore the leftover 9000 numbers have to be a duplicate which make sense.

4. TIME COMPLEXITY ANALYSIS

Since Big O notation relies on worst case scenerio the time might not be exectly align to it however the time different between all the differents functions does say something about the efficiency of the functions, functions with big O(n) have similar tabulated time and functions with O(n^2) are obviously slower than all the other functions.

5. CONCLUSION

I've learn a lot about the implementations of various differents algorithm as well as having an understanding of their performance and how does different implementation changes their run time in a different way even if it's big O notation are the same, for example: distinct search are slightly slower than max/min search even though both of them have O(n) because in my implementation distinct search requires two extra array and pushing items into those extra array also add a small extra time.

6. REFERENCES

Wikipedia contributors. (2022, August 8). Linear search. Wikipedia. Retrieved September 27, 2022, from https://en.wikipedia.org/wiki/Linear_search

Wikipedia contributors. (2022b, September 16). Binary search algorithm. Wikipedia. Retrieved September 27, 2022, from https://en.wikipedia.org/wiki/Binary_search_algorithm

GeeksforGeeks. (2022, September 1). Bubble Sort Algorithm. Retrieved September 27, 2022, from https://www.geeksforgeeks.org/bubble-sort/

Wikipedia contributors. (2022d, September 25). Insertion sort. Wikipedia. Retrieved September 27, 2022, from https://en.wikipedia.org/wiki/Insertion_sort

Wikipedia contributors. (2022d, September 25). Selection sort. Wikipedia. Retrieved September 27, 2022, from https://en.wikipedia.org/wiki/Selection_sort

Wikipedia contributors. (2022b, September 10). Merge sort. Wikipedia. Retrieved September 27, 2022, from https://en.wikipedia.org/wiki/Merge_sort#Top-down_implementation

7. APPENDIX

Your program codes

see attachments