GTU DEPARTMENT OF COMPUTER ENGINEERING

CSE-222 SPRING 2023 HOMEWORK REPORT

ONUR ATASEVER

210104004087

a)

Analysis of InsertionSort:

For comparision of each element of map, if map is already sorted each element of map controls only previous element. So best case is map is sorted and it is $T(n) = \Theta(n)$

In worst case map is reverse sorted. And each element should control all previous elements. If index is 1 it has 1 element to control, if index is 2 it has 2 elements to control, if index is n-1 it has n-1 elements to control. Total number of operation is summing of them. And it is $(n-1)(n) / 2$. So worst case it $T(n) = \Theta(n^2)$

Average case depends on the distribution of inputs. And comparision number depends on number of inverse ordered element relation. And it is $T(n) = O(n^2)$

Analysis of Bubble Sort:

In Bubble Sort algorithm, it checks adjacents as doubles. In best case map is already sorted and it compares adjacents one time. And there is n-1 comparison. So $T(n) = \Theta(n)$

In the worst case map is reverse sorted. It means each element should be compared with rest of map's elements. First element should be compared with n-1 elements, second one n-2 … And number of operation is $n(n+1)/2$.
So $T(n) = \Theta(n^2)$

Average case depends on the distribution of inputs. And comparision number depends on number of inverse ordered element relation. And it is T(n) = O(n^2)

Analysis Of Selection Sort:

Selection sort algorithm finds minimum value and puts it the beginning of the list.

The best case is sorted map. Eventhough map is sorted, it will find minimum value of map each time. It will traverse all map elements. To find minimum value it will pass n elements. And to find second minimum element it will pass n-1 elements… So T(n) = ϴ(n^2)

In the worst case map is reverse sorted. But the rest of actions are same with best case scenario. It will find minimum value of map each time. It will traverse all map elements. To find minimum value it will pass n elements. And to find second minimum element it will pass n-1 elements… So T(n) = ϴ(n^2)

The average case is state of being partial sorting. And average case is between best and worst case so T(n) = ϴ(n^2)

Analysis of Merge Sort:

Merge Sort algorithm divide list 2 pieces each time and it combines them by sorting. It does not matter whether map is sorted. For each situation there are log(n) steps while dividing and for each steps there are n copy operaitons. So for dividing n*log(n) operations. And there ares n*(log(n)) operations for combination.

So Tbest(n) = Tworst(n) = Taverage(n) = n*(log(n))

Analysis of Quick Sort:

The best case for Quick sort is having same number of elements for both subarrays. If it happens, it means there is log(n) levels of recursion. And for each level n actions is performed. So T(n) = Ɵn*log(n)

And the worst case is being pivot is smallest or biggest element of map. So we decrease 1 size of map for each recursive call until size is 1. So number of recursive calls is n-1. And for each level n actions are performed.
So T(n) = O(n^2).

The average case depends on pivot. Running time will be bounded by a constant x*n*logn. So T(n) = O(n*log(n)) b)

Input = "'Hush, hush!' whispered the rushing wind."

Running time for Bubble Sort in nanoseconds: 935200

Running time for Insertion Sort in nanoseconds: 182100

Running time for Selection Sort in nanoseconds: 200600

Running time for Quick sort in nanoseconds: 110200

Running time for Merge Sort in nanoseconds: 121800

Input: “Buzzing bees buzz.”

Running time for Bubble Sort in nanoseconds: 167600

Running time for Insertion Sort in nanoseconds: 93900

Running time for Selection Sort in nanoseconds: 90700

Running time for Quick sort in nanoseconds: 51700

Running time for Merge Sort in nanoseconds: 64000

Best-Average-Worst Case For Bubble Sort

Input: “caaxxxbbbbyyyyy” Already sorted. Best Case

Running time for Bubble Sort in nanoseconds: 8000

Input: “yyyyybbbbxxxaac” Inversely sorted. Worst Case

Running time for Bubble Sort in nanoseconds: 75700

Input: “xxxaayyyyycbbbb” Partial sorted. Average Case

Running time for Bubble Sort in nanoseconds: 48000

Best-Average-Worst Case For Insertion Sort

Input: “caaxxxbbbbyyyyy” Already sorted. Best Case

Running time for Insertion Sort in nanoseconds: 14900

Input: “yyyyybbbbxxxaac” Inversely sorted. Worst Case

Running time for Insertion Sort in nanoseconds: 52500

Input: “xxxaayyyyycbbbb” Partial sorted. Average Case

Running time for Insertion Sort in nanoseconds: 47600

Best-Average-Worst Case For Selection Sort

Actually there is not best average or worst cases for selection sort because for all cases T(n) = O(n^2). The following results also prove this.

Input: “caaxxxbbbbyyyyy” Already sorted

Running time for Selection Sort in nanoseconds: 53300

Input: “yyyyybbbbxxxaac” Inversely sorted

Running time for Selection Sort in nanoseconds: 46500

Input: “xxxaayyyyycbbbb” Partial sorted.

Running time for Selection Sort in nanoseconds: 53500

Best-Average-Worst Case For Merge Sort

Actually there is not best average or worst cases for Merge Sort because for all cases T(n) = O(n*logn). The following results also prove this.

Input: “caaxxxbbbbyyyyy” Already sorted.

Running time for Merge Sort in nanoseconds: 47800

Input: “yyyyybbbbxxxaac” Inversely sorted.

Running time for Merge Sort in nanoseconds: 53900

Input: “xxxaayyyyycbbbb” Partial sorted.

Running time for Merge Sort in nanoseconds: 47200

Best-Average-Worst Case For Quick Sort

Input: “xxxaabbbbcyyyyy” Pivot is median. Best Case

Running time for Merge Sort in nanoseconds: 41400

Input: “caabbbbxxxyyyyy” Pivot is smallest. Worst Case

Running time for Merge Sort in nanoseconds: 64500

Input: “aabbbbcxxxyyyyy” Pivot is not median, smallest or largest. Average Case

Running time for Merge Sort in nanoseconds: 44200 c)

According to b option for small and sorted inputs Bubble sort is fastest. For partial sorted inputs fastest is Quick sort then Merge Sort.For inversely sorted inputs, It seems Selection sort is best choice. Of course all this observations can change because size of inputs can change and also for all sort algorithms best-average-worst case scenarios are not same.

According to a option to decide which is most logical, we should estimate the order of map. If it is already sorted Insertion Sort is fastest. If map is reverse sorted, Quick sort is best choice. Usually we do not know whether it is sorted or unsorted. So we should go on above average case. According to average cases Quick Sort and Merge sort is T(n) = O(nlogn). If number of input is smaller Quick sort is best choice. And We have atmost 26 inputs. So Quick sort is faster. Because eventhough it's number of comparision is more than merge sort's, it partially sorts array according to pivot value. This is reason why it is faster.

d)

Bubble Sort and Insertion Sort can keep the input ordering but Quick Sort and Selection Sort can not keep. In selection sort It find minimum value and It swaps with first value of subarray. Maybe first value and second value was equal but because of swapping order is broken. In quick sort there is swapping which depends on pivot. It looks smaller number than pivot from leftmost and It looks largest or equal number than pivot from rightmost. Since largest or equal situation begins from rightmost, for some swapping order can change. For example a=3, b=4, c=3, d=3. Pivot is a=3, and it swaps b and d. In Bubble Sort and Insertion sort, order does not change because swapping is between adjacents.