Homework 7

P1

(a) Code implementation in vs code
(b) Code implementation in vs code
(c) Code implementation in vs code
(d) Each version of Quicksort is different due to how it is selecting and furtherly places the pivot. Swap count(how many elements are swapped), the partition balance(how well the pivots splits the array) and Recursion depth(how many recursive calls are needed) are being affected.

Lomoto Partition:

- typically worse than Hoare due to an excessive number of swaps.
- In random data, swaps seem less costly, Lomoto can still perform well
- In worst-case scenarios, it suffers due to unbalanced partitions

Hoare Partition:

- Expected to be faster than Lomoto due to fewer swaps
- More balanced partitions mean fewer recursive calls, improving efficiency
- In our results, Hoare is slightly slower than Lomoto Partition because if the data is completely random, the while loops inside Hoare's partitioning take longer to find the correct elements to swap

Median-Of-Three:

- Designed to avoid worst-case scenarios
- More overhead due to median calculation, which slightly increases the execution time
- In general the median selection doesn't help much, making it only slightly faster than Lomoto in general cases. In our case this general rule is not followed which may be due to the fact that performing two comparisons and up to two swaps before the start of the partition is extra work which may add a small overhead that is not always justified when dealing with fully randomized data.

P2

(a)code implementation in vscode

Homework 7
ADS

Problem 7.2

(b) The three-way Quicksort uses two pivots to divide the array into three subarrays (In the Best Case => 3 subarr are equal and $\simeq n/3$)

1st part ($\leq$ pivot 1) $\simeq n/3$ elements

2nd part (pivot 1 < 2nd part $\leq$ pivot 2) $\simeq n/3$ elements

3rd part (> pivot 2) $\simeq n/3$ elements

The recursive call occurs three times => $3T(n/3)$

The partioning step:

- we choose two pivots and we scan the array once to split them into three subcategories. Because the array is scanned once => $O(n)$ time complexity

So, for best case (~~subarr equal~~ (subarrays of equal values =>, $\simeq n/3$ elements each) the recurrence relation is: $T(n) = 3T(n/3) + O(n)$;

$a = 3$, $b = 3$; $f(n) = O(n)$

$n^{\log_b a} = n^{\log_3 3} = n^4 = n$

$n^{\log_b a}$? $O(n)$

$f(n) = O(n)$ and $n^{\log_b a} = n^1 \Rightarrow f(n) = \Theta(n^{\log_b a})$

(Case 2 of the Master's theorem $\Rightarrow T(n) = \Theta(n^{\log_b a} \log n)$

$= \Theta(n \log n)$

Worst case occurs when partitioning is highly unbalanced (e.g.: pivots are chosen poorly (smallest and highest element => almost all fall under one partition => problem is not reduced to a smaller number of steps).

Now, instead of getting three equal parts ($\simeq n/3$ each) we will have one partition of size $n-2$ and the other two are empty (because we got pivot 1 as the smallest element and pivot 2 as the largest) $\Rightarrow$

$\Rightarrow T(n-2)$

Because the ~~n~~ elements of the array are scanned once $\Longrightarrow$ $O(n)$

$\Rightarrow$ recursion formula for worst case :

$T(n) = T(n-2) + O(n)$;

$T(n) = T(n-2) + \overset{\text{"constant"}}{c'} n$

Induction Hypothesis:

We guess that

$T(n) \leq cn^2$

linear polynomial of deg 2

Base case : $T(n) = O(1)$, which satisfies $\overset{\uparrow}{T(n)} \leq \overset{\uparrow}{cn^2}$

Induction step: We assume that : $T(k) \leq c k^2, \forall k < n$

We now need to proof this for n:

$T(n) = T(n-2) + O(n)$

By the induction hypothesis: $T(n-2) \leq c(n-2)^2$

$T(n) \leq c(n-2)^2 + O(n)$

$T(n) \leq c(n^2 - 4n + 4) + O(n)$

$T(n) \leq cn^2 - 4cn + 4c + O(n)$

$O(n)$ is at most ~~n~~ multiplied by some constant p:

$O(n) \leq pn$, we rewrite

$T(n) \leq cn^2 - 4cn + 4c + pn$

$T(n) \leq cn^2 - (4c - p)n + 4c$

For $T(n) \leq cn^2$ to hold, we need the extra terms (the residual one's) to be ≤ 0

$$-(4c-p)n + 4c \leq 0 \quad / (-1)$$

$$\begin{cases} (4c-p)n - 4c \geq 0 \\ (4c-p)n \geq 4c \end{cases}$$

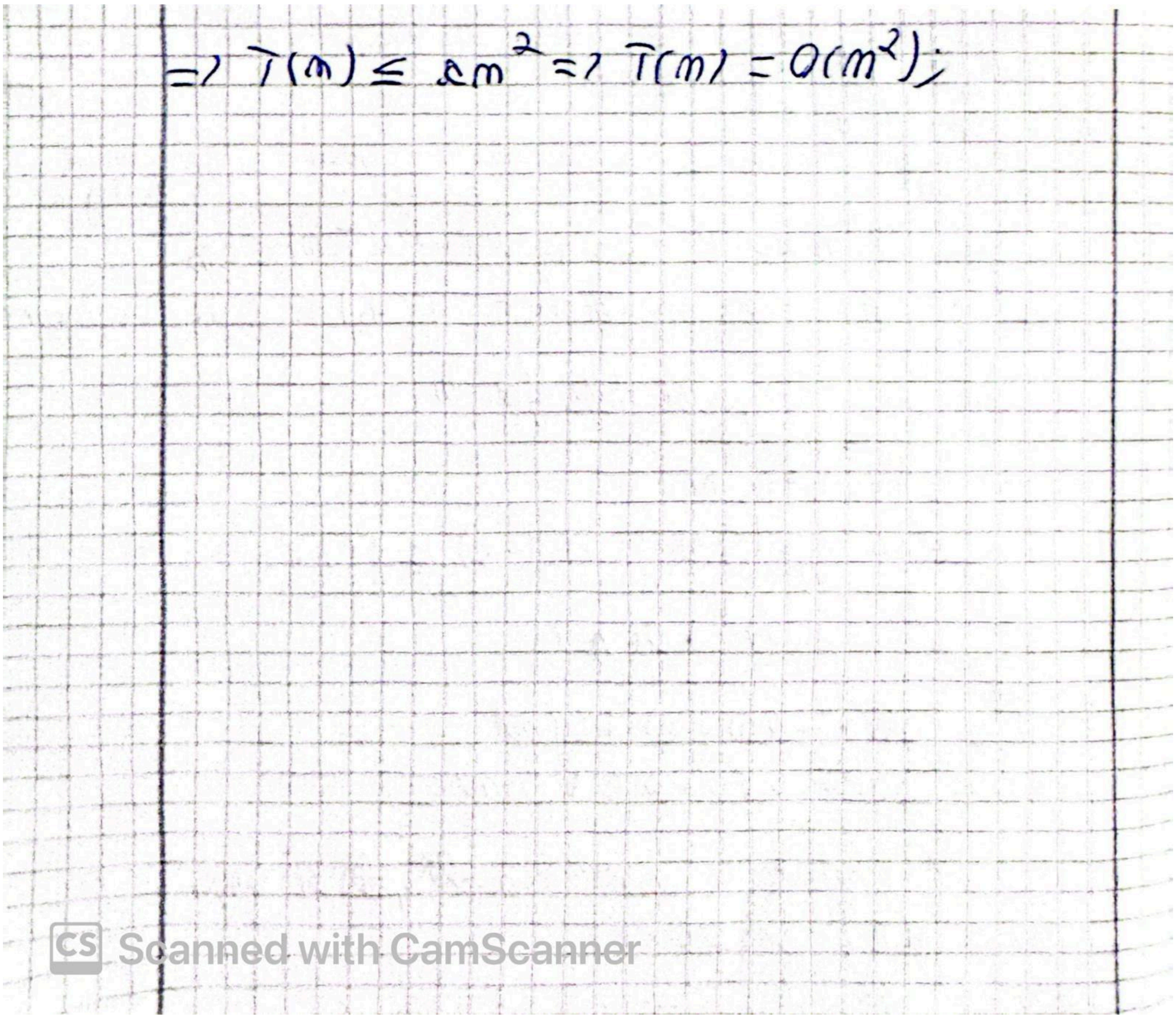
$$4c - p \geq \frac{4c}{n}$$

As $n \to \infty \Rightarrow \frac{4c}{n}$ approaches 0 for sufficiently large n

$\Rightarrow 4c - p \geq 0 \Leftrightarrow 4c \geq p \Rightarrow c \geq \frac{p}{4}$

For $c \geq \frac{p}{4}$ our assumption works $\Rightarrow$ The induction holds

$\forall n \geq 1, c \geq p/4$



$\Rightarrow T(n) \leq 2n^2 \Rightarrow T(n) = O(n^2);$

(c) see code implementation in vs code