Solved by SHOROUK GABR AWWAD 30002030

Problem 5.1 d

```
the lumoto case: 0.000308461
the hoare case: 0.000240452
the 3_median case: 0.000227821
```

```
the lumoto case: 0.000353425
the hoare case: 0.000268715
the 3_median case: 0.000265882
```

The 3_median pivot is considered to be the one with the least time complexity, for the reason that in order to get the pivot there is some sort of prior arrangement, which reduces the amount if time taken for the quick sort to go with.

The hoare pivot partitioning comes in the second rank as it uses two indices instead of only one. The swaps happening in the partitioning are in a random way to allow the two indices I,j to approach each other. So , both indices are working according to two diffenet comparisons to reach the same aim.

Lumoto pivot partitioning is considered to be the slowest among the three mechanisms, since it works with only one index that goes through the whole array and swaps the elements after comparing them to the pivot.

The second Question:

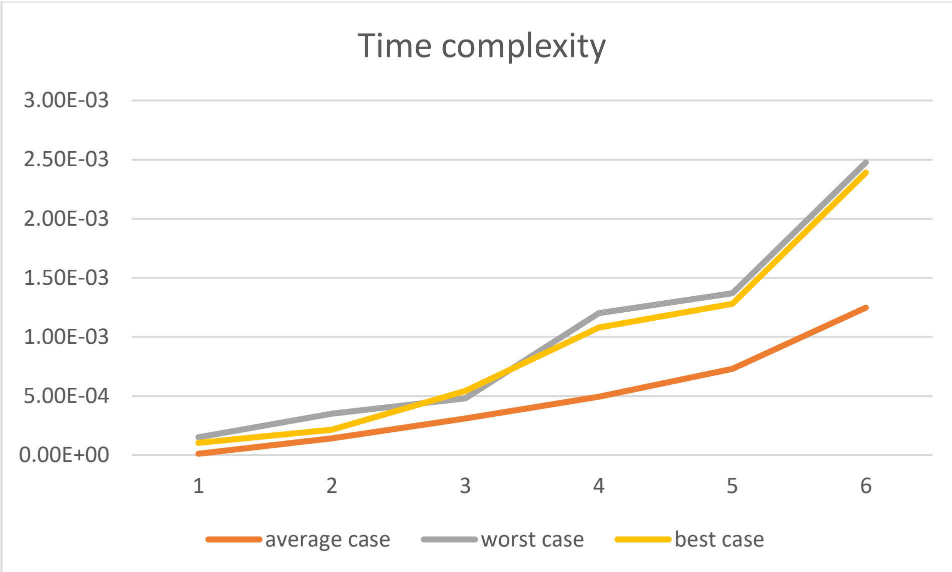
Problem 5.2b

Algorithm Pseudo code:

```
QuickSort (A,p,q)
Left_Index = p+1
Right_Index= q-1
Left_pivot
For i=0 to q-1
    If  A[i] < pivot1
        Swap  A[i] and arr[left_index]);
        Left_index++
      else if(A[i] >= pivot2){
```

```
        while(arr[right_index] > pivot2 and  i < right_index)
          right_index--
        swap(arr[i], arr[r_in]);
        right_index--;

        if(A[i] < pivot1){
          swap arr[i] and arr[left_index]
          left_index++
      i++

    left_index--Ɵ
    right_index++

    swap arr[p] and arr[l_in]
    swap arr[q] and arr[r_in]

    Left_Index = left_index;
    Right_Index= right_index;
      quicksort4(arr, p, Left_Index-1);
      quicksort4(arr, Left_Index+1, Right_Index-1);
      quicksort4(arr, Right_Index+1, q);
```

Solved by SHOROUK GABR AWWAD 30002030



In the quick sort algorithm, the worst case is when the array is already sorted whether according to an increasing pattern or a decreasing pattern.

In this case the time complexity is :

T(n) = T(n-2)+T(0)+T(0)+Θ(n)

In the best case, we suppose that the recurrence will split the array into three equal parts each time , and by using the recurrence tree we get T(n) = Θ(nlogn)

Solved by SHOROUK GABR AWWAD 30002030

assuming that the set already sorted and the two pivots are at the beginning.

Subject. Date.

$$T(n)$$

$$3T(n/3) \qquad 3T(n/3) \qquad 3T(n/3) \qquad 3 * n/3 = n$$

$$3T(n/9) \qquad 3T(n/9) \qquad 3T(n/9) \qquad 9\,n/9 = n$$

$\lg_3 n$

$$\text{So } T(n) = n + n + n \cdots\cdots n_{H = \lg n}$$

$$T(n) \in \Theta(n \lg n) \Rightarrow \text{the best case}$$

The worst case is when

$$T(n) = T(n-2) + T(0) + T(0) + \Theta(n)$$

$$T(n) = (n-2) * n/2 = \frac{n^2 - 2n}{2}$$

$$T(n) = \Theta(n^2)$$

Solved by SHOROUK GABR AWWAD 30002030

The third Question 5.3

5.3 : Decision tree

The proof of ($\log(n!) \in \Theta n\log n$)

while log n factorial is $= \sum_{k=1}^{n} \log(k)$

extracting the term

$\log(n!) = \log(1) + \log(2) + \log(3) + \log(4) + \dots + \log(n)$

$n\log(n) = \log(n) + \log(n) + \log(n) + \dots + \log(n)$

while $\log(n) > \log(1)$ and $\log(n) > \log(2) \dots \cancel{\log(n) >}$

$\dots \log(n) > \log(n-1)$

then $\log(n!) \leq n\log(n)$ then $\log(n!) \in O(n\log n)$ #

let $(n!)^2 = n! \times n!$

$(n!)^2 = (n)*1 + (n-1)*2 + (n-2)*3 \dots (1*n)$

$$\prod_{k=1}^{n} (n-k+1)*k = \prod_{k=1}^{n} \underbrace{(-k^2 + nk + k)}_{p(k) = -k^2 + nk + k}$$

by taking the derivative of $p(k)$

$p'(k) = -2k + n + 1$ let $p'(k) = 0$

$k = \frac{n+1}{2}$ while the Coeffecient of k^2 is negative

then $\frac{n+1}{2}$ represents a peak So for $k = 1$ & $k = n$

$p(k)$ is <u>not</u> at it's max

$$(n!)^2 = \prod_{k=1}^{n} p(k) \geq \prod_{k=1}^{n} p(k) \text{ at } k=1,n$$

$$(n!)^2 \geq \prod_{k=1}^{n} n \quad \text{so, } (n!)^2 \geq n^n$$

taking log

$2\log(n!) \geq n\log(n) \Leftrightarrow \log(n!) \in \omega(n\log n)$

Solved by SHOROUK GABR AWWAD 30002030

So,

while $\log(n!) \in O(n\log n)$

& $\log(n!) \in \Omega(n\log n)$

$\log(n!) \in \Theta(n\log n)$ $\square$