uOttawa

CSI 3105 - Design and Analysis of Algorithms

Fall 2022 Section A

Professor Jean-Lou De Carufel

Assignment 1

Kien Do (ID: 300163370)

1. **Answer**

   See scanned hand-written page.

2. **Answer**

   If $n = 2$ and $A[2] - A[1] > 0$, return the result of $A[2] - A[1]$. $\xleftarrow{O(1)}$

   Otherwise,

   Copy the elements of array A into new array B. $\xleftarrow{O(n)}$

   Sort array B using merge sort. $\xleftarrow{O(n\log(n))}$

   Create two variables $numOne$ and $numTwo$. Let variable i be 0 and variable j be 1. $\xleftarrow{O(1)}$

   Iterate through array B, incrementing i and j by 1 after each iteration. In each iteration, calculate $B[j] - B[i]$. If $B[j] - B[i] > 0$ and $B[j] - B[i] < minimum$,

   - reassign $minimum$ with $B[j] - B[i]$,

   - reassign $numOne$ with $B[i]$,

   - reassign $numTwo$ with $B[j]$.

   Stop the iteration when j equals n. $\xleftarrow{O(n)}$

   Create variables $indexOne$ and $indexTwo$.

   Go back to array A, iterate through it, then set $indexOne$ as the current index in the iteration if the current number in array A equals $numOne$ and set $indexTwo$ as the current index in the iteration if the current number in array A equals $numTwo$. Upon first assignment of $indexOne$ or $indexTwo$, do not reassign a new index to $indexOne$ or $indexTwo$ again. $\xleftarrow{O(n)}$

   Return $indexOne$ and $indexTwo$, where $indexOne$ and $indexTwo$ represent the indices of array A such that $A[indexTwo] - A[indexOne] > 0$ and $A[indexTwo] - A[indexOne]$ is the minimum.

   Time complexity:

   Recall that the time complexity of an algorithm is the sum of the time complexity of every operation. Ignoring the constant $O(1)$ time complexities, we have that,

   $$T(n) = O(n\log(n)) + O(n) + O(n)$$

   where $T(n)$ is the non-constant time complexity of the algorithm.

   We have that,

   $$\begin{aligned} T(n) = n\log(n) + n + n &\leq n\log(n) + 2n \\ &\leq n\log(n) + 2n\log(n) \\ &\leq 3n\log(n) \\ &= O(n\log(n)) \end{aligned}$$

   Therefore, the algorithm runs in $O(n\log(n))$