Chance McCormick

COSC 3020

Lab_04 Time complexity

10/4/2019

1. If the array has 0 or 1 element the array is sorted, stop. T(1)
2. Go through all permutations until sorted array is found. T(n*n-1)
3. Test to find an array that is sorted. T(n)

Recurrance Relation:

$$T(n)=\begin{cases} 1 \; if \; n = 0 \\ 1 \; if \; n = 1 \\ (n * n - 1 \;) \; if \; n > 1 \end{cases}$$

Solve by substitution

T(n) = T(n*n-1)

=2T(n*n-1)

=3T(n*n-1)

…

$=1^{i}T(n*n-1^{i})$

For i = n

$=T(1) + n +(n*n-1)= 1 + n + (n*n-1) \in \ominus(n!)$

The best-case scenario would be to find the sorted array on the first permutation. It would have a time complexity of $\ominus(n)$.

The worst-case scenario is a time complexity of $\ominus(n!)$

If using a randomly generated permutation to find a sorted array the worst case scenario would be $\ominus(\infty)$. This is due to the fact that it is entirely possible to never find the sorted permutation if the permutations are randomly generated.