

Find unique Element Merge Sort Approach

Input: A sorted sequence of N ints where one and only one element is unique; the other elements are repeated an arbitrary amount of time.

Output: the unique element

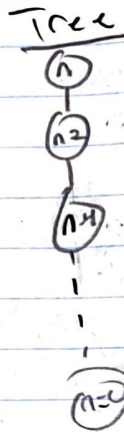
Merge_Find ($A = (a_1, a_2, \dots, a_n)$)

- 1 If $n = 1$
- 2 return $A[0]$
- 3 elif $(a_1 = a_2)$
- 4 return (Merge_Find($a_3 \dots a_n$))
- 5 else
- 6 return a_1

Time complexity Merge-Find($A(a_1, a_2, \dots, a_n)$)

$$T(n) = T(n-2) + O(1)$$

n
—
 $n-2$
 $n-4$
 \vdots



$O(1)$
|
|
|
|
|
|

$O(n)$

$O(n)$ because we do a constant amount of work $\frac{1}{2}$ times but we don't care about coefficients so $O(n)$

Proof

Theorem: If A is a non-empty sorted list of n integers where n is not infinity and there is one and only one unique element in A the Merge-Find(A) will return the unique element.

Proof: we will proceed with Induction

Base case: $n=1$

If $n=1$ then the unique element is the only element in the list and we return the first element. this happens on line 1 when

there is a check of n . If $n=1$ the first element of A is returned.

Induction Hypothesis: Assume the algorithm works for n items, prove that it works for $n+1$

On line 3 there is a check if $a_1 = a_2$ this checks for unique elements. if not unique, we slice from list and continue from $a_3 \dots a_n$ else we return a_1 .