Problem Set 3

Noella James

02/11/2017

collaborators: none

1 Problem 3-3: Median Finding

Input: $l_1, s_1, e_1, l_2, s_2, e_2$ where l_1 is the first sorted array input. s_1 and e_1 are the start and end indices of l_1. l_2 is the second sorted array input. s_2 and e_2 are the start and end indices of l_2.
Output: The median of l_1 and l_2.

Algorithm 1 MEDIAN FINDING

procedure MEDIAN-FINDING($l_1, s_1, e_1, l_2, s_2, e_2$)
    if $e_1 - s_1 = 1$ AND $e_2 - s_2 = 1$ **then** $\triangleright$ Both arrays are single element arrays, or has reached recursively to this level.
        $med \leftarrow \frac{l_1[s_1]+l_2[s_2]}{2}$
        return med
    end if
    $left_{med} \leftarrow$ MEDIAN(l_1, s_1, e_1)
    $right_{med} \leftarrow$ MEDIAN(l_2, s_2, e_2)
    if $left_{med} = right_{med}$ **then**
        return $left_{med}$
    else if $left_{med} < right_{med}$ **then**
        $s_1 \leftarrow \lfloor \frac{s_1}{2} \rfloor + 1$
        $e_2 \leftarrow \lceil \frac{e_2}{2} \rceil$
        return MEDIAN-FINDING($l_1, s_1, e_1, l_2, s_2, e_2$)
    else $\triangleright$ $left_{med} > right_{med}$
        $e_1 \leftarrow \lceil \frac{e_1}{2} \rceil$
        $s_2 \leftarrow \lfloor \frac{s_2}{2} \rfloor + 1$
        return MEDIAN-FINDING($l_1, s_1, e_1, l_2, s_2, e_2$)
    end if
end procedure

Algorithm 2 MEDIAN

procedure MEDIAN(l, s, e) $\triangleright$ l is the array, s is the starting index, e is the end index
  $length \leftarrow e - s$
  if $length \% 2 = 0$ **then**
    $median \leftarrow l[\frac{length}{2}]$
    return $median$
  else
    $value_l \leftarrow l[\frac{length}{2}]$
    $value_r \leftarrow l[(\frac{length}{2}) + 1]$
    $median \leftarrow \frac{value_l + value_r}{2}$
    return $median$
  end if
end procedure

Lemma 1.1. *The MEDIAN-FINDING algorithm will always find the correct median of* 2 *sorted arrays.*

Proof. **Base Case:** n and m are both 1. The algorithm returns the median of the two values in the individual arrays.
Induction Hypothesis: The algorithm will return the median of all values of n and m given that $n \leq j$ and $m \leq k$ for arrays l_1 and l_2 respectively.
Induction: We prove that the algorithm works for values $j + 1$ and $k + 1$ for arrays l_1 and l_2 respectively. When arrays l_1 and l_2 are passed into the algorithm, they are immediately checked to see if they are single element arrays. If they are, we immediately return their median. However, if they're not, we calculate the medians of arrays l_1 and l_2 respectively. If the respective medians are equal, we return the median. However, if they're not, we recursively call the algorithm and shorten each array by half. Thus by the induction hypothesis where the algorithm will return the median of all values of n and m given that $n \leq j$ and $m \leq k$, this proves that the algorithm is correct since for $j + 1$ and $k + 1$ divided by 2 are less than j and k respectively. $\square$

1.1 Recurrence Relationship

$T(n, m) = T(\frac{n}{2}, \frac{m}{2}) + \theta(1)$
$T(1, 1) = \theta(1)$

Note 1.1. We assume that $n > m$ for solving the recurrence relationship.

$T(n, m) = T(\frac{n}{2}, \frac{m}{2}) + c$
$T(1, 1) = c$

$T(n, m) = T(\frac{n}{2}, \frac{m}{2}) + c = (T(\frac{n}{4}, \frac{m}{4}) + c) + c = ((T(\frac{n}{8}, \frac{m}{8}) + c) + c) + c = ((T(1, 1) + c) + \ldots + c) + c = c \log n = \theta(\log n)$

If $n < m$, the recurrence relationship will be $\theta(\log m)$.
The final recurrence relationship is $\theta(\log(max(n, m)))$.