Intro to Algorithms, COMP-160, Homework #3

Benjamin Tanen, 02/15/2016

1. You are given two arrays, A_1 and A_2, each containing n numbers in sorted order. Show how to compute the median of all the numbers in A_1 and A_2, in $O(\log n)$ time.

   You can go about finding the median of all numbers in A_1 and A_2 by comparing the medians of each list and chopping parts off the lists accordingly. This is possible because we know that finding the median of any sorted list A_n takes constant time $\Theta(1)$ (simply get the middle element in A_n since we know the length and it is sorted).

   Knowing this, we can compare the medians of two lists, A_1 and A_2, and cut down the lists based on which list contains the smaller and larger medians. Say we determine that, knowing list A_1 and A_2 have medians m_1 and m_2 respectively, $m_1 < m_2$. We can then cut off a portion from the front of A_1 and a portion from the back of A_2. By doing this, we continuously reduce the sizes of A_1 and A_2 until we get single values for each list. For this algorithm, we can cut of $\lfloor \frac{l_1+l_2}{4} \rfloor$ elements from the front and back (where l_1 and l_2 is the length of A_1 and A_2 respectively), reducing the combine sizes in half each time.

   Using all of this, we can use the following algorithm to find the median of both lists:

   a. Calculate medians m_1 and m_2 for the lists A_1 and A_2
   b. If $m_1 < m_2$, cut off $\lfloor \frac{l_1+l_2}{4} \rfloor$ elements off the front of A_1 and $\lfloor \frac{l_1+l_2}{4} \rfloor$ elements off the back of A_2.

      If $m_1 > m_2$, cut off $\lfloor \frac{l_1+l_2}{4} \rfloor$ elements off the front of A_2 and $\lfloor \frac{l_1+l_2}{4} \rfloor$ elements off the back of A_1.

      If $m_1 = m_2$, return either m_1 or m_2 because this is median of both lists combine.
   c. If A_1 and A_2 are both length of 1, return the median of the two numbers.

      Otherwise, repeat step (a) with total number of elements in A_1 and A_2 reduced in half.

   Looking at this algorithm, we can see that each iteration takes constant time $\Theta(1)$ to calculate the two medians (by simply indexing), compare their values, and cut off portions of the lists. Beyond this, each iteration removes portions of the lists, reducing the number of elements in half each time without having to recursively branch the work (like in merge sort). This indicates the algorithm is logarithmic in behavior, with constant time at each level, or overall behavior of $\Theta(\log n)$.

   This algorithm also indicates a recurrence of $T(n) = T(\frac{n}{2}) + \Theta(1)$. Using the master method, we can solve this recurrence for $T(n) = \Theta(\log n)$ which confirms what we see above.

   Note: Since this algorithm is $\Theta(\log n)$, it can also be said that it is $O(\log n)$.

   An example of this algorithm is shown below, with $A_1 = \begin{bmatrix} 1 & 3 & 5 & 7 & 11 & 12 \end{bmatrix}$ and $A_2 = \begin{bmatrix} 2 & 3 & 4 & 6 & 10 & 13 \end{bmatrix}$.

1. $A_1 = \begin{bmatrix} 1 & 3 & 5 & 7 & 11 & 12 \end{bmatrix}$, $m_1 = 6$
   $A_2 = \begin{bmatrix} 2 & 3 & 4 & 6 & 10 & 13 \end{bmatrix}$, $m_2 = 5$
   Cut off $\lfloor \frac{l_1+l_2}{4} \rfloor = \lfloor \frac{6+6}{4} \rfloor = 3$ elements off front of A_2 and back of A_1
2. $A_1 = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix}$, $m_1 = 3$
   $A_2 = \begin{bmatrix} 6 & 10 & 13 \end{bmatrix}$, $m_2 = 10$
   Cut off $\lfloor \frac{l_1+l_2}{4} \rfloor = \lfloor \frac{3+3}{4} \rfloor = 1$ elements off front of A_1 and back of A_2
3. $A_1 = \begin{bmatrix} 3 & 5 \end{bmatrix}$, $m_1 = 4$
   $A_2 = \begin{bmatrix} 6 & 10 \end{bmatrix}$, $m_2 = 8$
   Cut off $\lfloor \frac{l_1+l_2}{4} \rfloor = \lfloor \frac{2+2}{4} \rfloor = 1$ elements off front of A_1 and back of A_2
4. $A_1 = \begin{bmatrix} 5 \end{bmatrix}$, $m_1 = 5$
   $A_2 = \begin{bmatrix} 6 \end{bmatrix}$, $m_2 = 6$
   Lengths of both A_1 and A_2 are 1, so take median of 5 and 6 $\rightarrow \frac{5+6}{2} = 5.5$