CS 381 PSO #2

Week 3

Divide and Conquer Steps

1. Divide
   a. divide the problem input into pieces
   b. E.g. MergeSort: Divide the array into two pieces (or 3, or 4, depending on your version)
2. Conquer
   a. perform the problem task on the pieces such that they are solved
   b. E.g. MergeSort: Each half, once you get down to array sizes of 1, will be considered sorted
3. Merge
   a. put the pieces back together such that the total combined is solved
   b. E.g. MergeSort: Go through each sorted subarray combining them in linear time such that they remain sorted, thus allowing the subarray in the next step up in the recursive chain to be conquered

Divide and Conquer

Consider an unsorted array A of integers.

Construct a Merge Sort algorithm to sort A using at most O(1) extra space. (Runtime does not have to stay the same as normal Mergesort)

Divide and Conquer

Find the square root of a positive number n.

Divide and Conquer

Find a peak element of an array A (local maximum is also a suitable name).

General Master Theorem Reminder

Case 1: $f(n) = O(n^c), c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$

Case 2: $f(n) = \Theta(n^{log_b a} \log^k n), k \geq 0$, then $T(n) = \Theta(n^{log_b a} \log^{k+1} n)$

Case 3: $f(n) = \Omega(n^c), c > \log_b a$, then $T(n) = \Theta(f(n))$

Regularity Condition: $af(n/b) \leq kf(n), k < 1$, and for some n large enough.

Master Theorem Questions

$$T(n) = 3T(n/3) + \sqrt(n)$$

$$T(n) = 2T(n/8) + 4n$$

$$T(n) = 6T(n/3) + 2n^2 \log(n)$$

$$T(n) = 8T(n/2) + n^3 \log(n)$$

Non-Master Theorem Recurrences

1. $T(n) = T(n/4) + 6n$
2. $T(n) = 2T(n-1) + 3T(n-2)$, where $T(0) = 1, T(1) = 1$

Divide and Conquer (Bonus Problem)

Find the longest sequence of ones in a binary array A using a Divide and Conquer approach.