PSO #2 Solutions Sketch (Week 3)

Week of 2021-09-06

1 Divide and Conquer Problems

1. 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)

   Solution: Instead of explicitly dividing recursively, we maintain two pointers x_1 and x_2, each of which point to the beginning of the two arrays we are intending to merge. So, given two arrays to merge, if the value of A at x_1 is less than or equal to the value at x_2, we increment x_1 and proceed. If the value at x_2 is less than the value at x_1, then we store the value at x_2 in a temporary variable y and move all the values starting from the index x_1 up until the index before x_2 to the right by 1. Then, we increment x_2, assign the value of y to the index x_1, and then increment x_1.

2. Find the square root of a positive number n.

   Solution: Initialize 0 and n as the first and last element in a "sorted" array of numbers consisting of the values contained in the real number line. Then, check the square of the number at the halfway point and if it is less than the number, assign the first element to the halfway point and if the it is greater than the number, assign the last element to the halfway point. Repeat until the desired accuracy is reached. The runtime should be $O(\log n)$.

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

   Solution: Use Binary Search to see if the middle element is a peak element. If it isn't, then recurse on the half of the array which contains the greater of the two neighbors of the middle element. The runtime should be $O(\log n)$.

4. (End of Slides. If you want to challenge students, give this to 'em.) Find the longest sequence of ones in a binary array A using a Divide and Conquer approach.

   Solution: Divide the array in half as in Merge Sort, but once arrays of size 1 are reached, maintain for each a subarray of 3 coordinate pairs, the first corresponding to the beginning and ending indices of the sequence of ones that is to the leftmost in the array, the second the beginning and ending indices of the longest sequence of ones in the array, and the third the beginning and ending indices of the rightmost sequence of ones in the array. For example, if we had an array like this: $B = [0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1]$, then the first coordinate pair for B would be $(1, 2)$, the second would be $(5, 7)$, and the third would be $(9, 9)$. When merging,

compare the second element of the third coordinate pair of the first array with the first element of the first coordinate pair of the second array. If they correspond to the last and first indices of their respective arrays, then compare the lengths of the second coordinate pairs of both arrays and the combined length of the rightmost sequence of ones in the first array and the leftmost sequence of ones in the second array to determine the longest sequence of 1s in the newly merged array. The first coordinate pair for the new merged array corresponds to the first coordinate pair for the left array, and the third coordinate pair for the new merged array corresponds to the third coordinate for the right array. If the third coordinate pair of the first array and the first coordinate pair of the second array do not meet, then simply compare the second coordinate pairs to determine the longest sequence of ones in the newly merged array.

2 Recurrences

1. Solve the following recurrences using the master theorem, or note if they cannot be solved

   (a) $T(n) = 3T(n/3) + \sqrt{n}$
   Applying master theorem case 1:
   $a = 3$, $b = 3$, $n^{\log_3 3} = n$. $f(n) = \sqrt{n} = O(n)$, so $T(n) = \Theta(n)$

   (b) $T(n) = T(3n/8) + 32$
   Applying master theorem case 2:
   $a = 1$, $b = 8/3$, $n^{\log_{8/3} 1} = n^0 = 1$. $f(n) = 32 = \Theta(1)$, so $T(n) = \Theta(\log n)$

   (c) $T(n) = 2T(n/8) + 4n$
   Applying master theorem case 3:
   $a = 2$, $b = 8$, $n^{\log_8 2} = n^{1/3}$. $f(n) = 4n = \Omega(n^{1/3})$, so $T(n) = \Theta(4n) = \Theta(n)$
   $af(n/b) \leq cf(n) \rightarrow 2f(n/8) \leq cf(n) \rightarrow n \leq c4n$ for some constant $c < 1$. This holds for $c \leq 1/4$, thus the regularity condition is satisfied

   (d) $T(n) = 2T(n/2) + n^2 \sin n$
   Master theorem case 3, but fails the regularity condition:
   $2f(n/2) \not\leq c \cdot f(n)$ when $n = \frac{3\pi}{2} + (2\pi)k$ for $k \geq 1$

   (e) $T(n) = 6T(n/3) + 2n^2 \log n$
   General master theorem case 3:
   $a = 6$, $b = 3$, $f(n) = 2n^2 \log n = \Omega(n^{e+\log_b a}) = \Omega(n^{e+\log_3 6}) = \Omega(n^{e+1.63})$a
   $T(n) = \Theta(f(n)) = \Theta(2n^2 \log n) = \Theta(n^2 \log n)$
   Regularity condition $af(n/b) \leq cf(n)$ for $1 > c \geq 2/3$

   (f) $T(n) = 8T(n/2) + n^3 \log n$
   General master theorem case 2:
   $a = 8, b = 2$,
   $f(n) = n^3 \log n = \Theta(n^3 \log n) = \Theta(n^{\log_2 8} \log n) = \Theta(n^{\log_b a} \log^k n)$ (where $k = 1$)
   $T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(n^{\log_2 8} \log^{1+1} n)) = \Theta(n^3 \log^2 n)$

2. Solve the following recurrences without using the master theorem

(a) $T(n) = T(n/4) + 6n$

$T(n) = O(n)$. Unravel recurrence for

$T(n) = (T(n/16) + 6n/4) + 6n = \ldots = T(n) + 6 \cdot \sum_{i=0}^{\log_4(n)-1} \frac{n}{4^i} = O(n)$

(b) $T(n) = 2T(n-1) + 3T(n-2), where T(0) = 1, T(1) = 1$ (Specify T(0) and T(1), also prove the guess using characteristic equation)

Solution: We guess $T(n) \leq c^n$, so we use the characteristic equation $c^n = 2 * c^{n-1} + 3 * c^{n-2}$ to find a suitable value for c, which is $c^2 = 2*c+3 \rightarrow (c-3)(c+1) = 0 \rightarrow c = 3, -1$. Thus, we guess $T(n) \leq 3^n$, or $T(n) = O(3^n)$.

Base Case: $n = 2, T(2) \leq 3^2$.
We show $T(2) = 2 * T(1) + 3 * T(0) = 5 \leq 3^2 = 9$, thus we have proven the base case.

I.H.: Assume for any $k \leq n$: $T(k) \leq 3^k$.

I.S.: $T(k+1) = 2 * 3^k + 3 * 3^{k-1} = 3^{k+1} \leq 3^{k+1}$. We have proven $T(k+1) \leq 3^{k+1}$, therefore $T(n) = O(3^n)$.