CMSC351-0101 - Fall '21 Homework 4 Due: Saturday, October 2, 2021

Problem 1. Imagine there is an algorithm, X, whose runtime follows the following recurrence equation:

$$T(n) = 3T(n/5) + 2n + 1, \qquad T(1) = 4$$

(a) Calculate $T(25)$ by hand. Show your work.

(b) Use the recursion tree method to solve the recurrence exactly, assuming n is a power of 5. For each subpart *briefly justify* and / or *show your work* when appropriate.

   (a) Draw the tree. You should show at least three levels at the top and at least two levels at the bottom (as done in class).

   (b) What is the height of the tree? (Note that a tree with one node has height 0, a tree with a root and some children has height 1, etc.)

   (c) How many leaves are there?

   (d) What is the total work done by the leaves?

   (e) What is the size of each subproblem at level i? (Note that the root is at level 0, its children are at level 1, etc.)

   (f) How much work does each subproblem at level i (above the leaves do)?

   (g) What is the total work for level i (above the leaves)?

   (h) Write a summation for the total work not including the leaves?

   (i) Simplify the summation.

   (j) What is the total work for the entire algorithm?

   (k) Verify the base case, $T(1)$.

Problem 2. For this question we will work with a slightly modified merge sort algorithm. In the class we saw how merge sort would divide the input array all the way to a single element array before we merge them. However, in modified version of merge sort we would split the arrays only up to the point where we have k subarrays, each of size, n/k. We would use insertion sort to sort each one of those k subarrays and then merge them using the merge subroutine. Consider, this level with k subarrays as the base case for this modified merge sort algorithm. Just like in the merge sort, if we merge two subarrays whose combined length is n, it takes $n - 1$ comparisons. Answer the following questions:

1. Draw a recursion tree, showing the top four and the bottom two levels.
2. What is the worst case number of comparisons to sort k subarrays each of size n/k using insertion sort (with sentinel)?
3. Solve the recursion tree for the worst case number of comparisons using merge sort. What is the base case?
4. What is the exact number of comparisons for the modified merge sort algorithm?
5. Is it better than the regular merge sort algorithm?
6. What happens when $k = n$?
7. What happens when $k = 1$?

Problem 3. We want to find the maximum and the minimum elements in an unsorted array of size, n using two different optimal strategies that yield the same runtime. You may assume n to be a power of 2.

(a) Write pseudo-code for an optimal iterative algorithm. Analyze the runtime exactly.

(b) Write pseudo-code for an optimal divide and conquer approach. Analyze the runtime exactly.