CS302 - Problem Set 3

Due: Monday, Oct 2. Must be uploaded to Canvas before the beginning of class.

Please be familiar with the sections of the syllabus on problem sets and honor code before starting this homework. You may also want to look at the grading rubrics.

1. *3-D Closest Points Continued* [**6 points**]
   In this problem, you will analyze the runtime of the algorithm for `CP_3D` from last week's problem set. Use recurrence relations and a proof similar to our proof of the master method to argue that the runtime of of the algorithm on n points is $O(n \log^2 n)$. (Recall $\log^2 n = (\log n)^2$.) You may assume that the asymptotic runtime of `CP_Almost2D` on a set of n points is $O(n \log n)$

2. *Master Method* [**6 points**]
   When solving divide and conquer algorithms, things can get messy when the input size n is not a power of b, where n/b is the size of the subproblems solved by the recursive calls. Our in-class proof of the master method assumed that n is a power of b. In this problem, you will show that this assumption can be dropped. Argue that the master method gives the same asymptotic scaling whether the problem size is $n = b^k$, or $n' = b^{k+1}$. This result shows that if our input is not originally a power of b, we can increase the size of the input (for example, by padding with zeros) to the next largest power of b without affecting the complexity of the algorithm.

3. *More Master Method*

   Suppose you have devised three different divide and conquer algorithms to solve the same problem:

   - Algorithm I: Divides the problem into three subproblems that are each a quarter of the size of the original problem. It uses linear time to combine the solutions to the subproblems.
   - Algorithm II: Divides the problem into 2 subproblems that are each 2/3 the size of the original problem. It uses constant time to combine the solutions to the subproblems.
   - Algorithm III: Divides the problem into 9 subproblems that are each a third the size of the original problem. It uses quadratic time to combine the solutions to the subproblems.

   (a) [**6 points**] For each algorithm, give the asymptotic complexity using big-O notation.

   (b) [**6 points**] State which of the algorithms you would use to solve an instance of this problem.

4. *QuickSort*

   Most of you should have covered QuickSort in 201. If you've forgotten, please review your notes. Try not to look at pseudocode - if you do, take some time in between looking at it and trying to recreate it yourself. You may find it helpful to work back and forth between creating the proofs and writing the pseudocode. Also, feel free to be a bit looser with your pseudocode. Use more English if that is easier. For example, writing “swap two elements of an array” is simpler than giving the mathematical description with a temporary value, etc.

(a) [**9 points**] Write pseudo-code for `Partition`.

(b) [**11 points**] Prove using a loop invariant that your algorithm for `Partition` is correct.

(c) [**0 points - do this if you need the practice**] Write pseudo-code for `QuickSort` using your `Partition` subroutine. Please make the pivot the first element of the part of the array in question.

(d) [**0 points - do this if you need the practice**] Prove your algorithm for `QuickSort` is correct.