Homework 5

CS 4104 (Spring 2014)

Assigned on Wednesday, April 2, 2014.
Submit PDF solutions by on Scholar by the beginning of class on Monday, April 9, 2014.

Instructions:

- You can pair up with another student to solve the homework. You are allowed to discuss possible algorithms and bounce ideas with your team-mate. **Do not discuss proofs of correctness or running time in detail with your team-mate.** Please form teams yourselves. Of course, you can ask me for help if you cannot find a team-mate. You may choose to work alone. *Each of you must write down your solution individually, and write down the name of the other member in your team. If you do not have a team-mate, please say so.* **If your solution is largely identical to that of your team-mate or another student, we will return it ungraded.**
- Apart from your team-mate, you are not allowed to consult any sources other than your textbook, the slides on the course web page, your own class notes, the TAs, and the instructor. In particular, do not use a search engine.
- Do not forget to typeset your solutions. *Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as n^2 and not as "n^2".* Students can use the LaTeX version of the homework problems to start entering their solutions.
- Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed pseudo-code without an explanation, we will not grade your solutions.*
- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.
- Do not describe your algorithms only for a specific example you may have worked out.
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*
- Describe an analysis of your algorithm and state and prove the running time. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.

Problem 1 (10 points) Solve the recurrence $T(n) = T(\lfloor\sqrt{n}\rfloor) + 1$. In words, the $T(n)$ is the running time of an algorithm that creates one sub-problem of the size equal to the square root of n, solves this sub-problem recursively, and spends one more unit of time. You can assume that $T(1) = T(2) = 1$ and that $n > 2$ in the recurrence relation. Remember to prove your solution by induction, even if you use the "unrolling" method to guess a solution.

Problem 2 (30 points) You are given three algorithms to solve the same problem of size n. Analyse each algorithm in $O()$ notation. Provide a clear proof of the analysis. Which algorithm would you choose and why? In other words, write down which algorithm is asymptotically the fastest and provide a proof why this algorithm is asymptotically the fastest of all three. If you can directly apply a formula we discussed in class, feel free to do so. For some sub-problems, you will have to come up with proofs from scratch, although your proofs will follow the general outlines we have used in class. Remember to prove your solution by induction, even if you use the "unrolling" method to guess a solution.

(i) Algorithm A solves the problem by dividing it into five sub-problems of half the size, recursively solving each sub-problem, and then combining the solutions in linear time.

(ii) Algorithm B solves the problem by dividing it into two sub-problems of size $n - 1$, recursively solving each subproblem, and then combining the solutions in constant time.

(iii) Algorithm C solves the problem by dividing it into three sub-problems of size $n/3$, recursively solving each sub-problem, and then combining the solutions in $O(n^2)$ time.

Problem 3 (25 points) Solve exercise 1 in Chapter 5 (page 246) of your textbook. Note that you cannot delete elements from the databases. The only operation you are allowed is to query the kth smallest value in one of the databases. You are likely to come up with an algorithm that somehow eliminates parts of each database and recurses on the remaining parts. Be sure to prove that whatever elements you may return from the recursive calls, these returned values are related to the the median element in the original set of $2n$ values, i.e., in the conquer step, how can you be sure that you have computed the median of the $2n$ values from the elements returned by the recursive calls?

Problem 4 (35 points) Solve exercise 3 in Chapter 5 (pages 246–247) of your textbook. Let us call the equivalence class with more than $n/2$ cards the *important* class. It is enough for your algorithm to return a card that belongs to this class, if it exists, or no card at all. Note that the problem specifies that the only operation you can perform on a pair of cards is to decide if they are equivalent. You cannot perform any other operation, e.g., compare them in order to sort them. Your proof of correctness must clearly address why your algorithm will find a set of important class, if it exists. There are many things that can go wrong. For instance, there may be an important class for all n cards but the recursive calls don't find any important class (for the subset of cards they process). Alternatively, the recursive calls may find a *different* important class. It is also possible that there is no important class for all n cards but your recursive calls find an important class. Your algorithm or your proof of correctness must consider all such bad eventualities and you must show that they cannot happen.