CSE 331: Introduction to Algorithm Analysis and Design
Fall 2012
Homework Policies

It is **your responsibility** to make sure you read and understand the contents of this document. If you have any questions, please contact the instructor and/or the TA. Or better yet, post a comment on the blog.

We begin with our policies for the homeworks. Not following these rules will result in some sort of penalty. (Read all of these policies by adding "Unless mentioned otherwise, " to the beginning of the first sentence.)

1. Collaboration is generally allowed on the homeworks. For your convenience, the collaboration policy is being reproduced below from the syllabus:

   - You are allowed to collaborate provided you have thought about **each problem for at least 30 minutes on your own**. This will help you in the exams.
   - You can collaborate on any homework in a **group of size at most 3**, including yourself. Note that you cannot collaborate with different groups for different problems. You must write the name of *everyone* in your group on your submission.
   - You can **only discuss the problems with your group till you come up with the proof ideas**: the detailed formal proof is something you should work on alone.
   - Your submitted homework must be **written in your own words**. Everything, including the proof idea, has to be written up individually. In particular, at no point of time should you have in your possession the written homework of someone else.

   Deviating from the rules above will be considered to be cheating.

2. We encourage you to typeset your homework using some text editor– this is not mandatory though. However, if you are writing up your homework, please make sure that your handwriting is legible. We reserve the right to deduct up to 20% of the total points for submitted work that is hard to read.

3. When a problem asks you to give an algorithm, we expect you to analyze the correctness and running time of the algorithm. Also by default, an efficient algorithm means one that runs in polynomial time.

4. Remember that when solving a problem, you can *only* use the information given in the problem statement. For example, you may not use information that is maybe common knowledge

but not specified in the problem statement. If for some reason, you want to assume something, please state it explicitly at the beginning of your answer. However, note that it will be up to the grader's discretion whether your assumption is reasonable. (See next point.)

5. Be very precise in what you write in your answers. Given a statement that could mean either (i) you did not get the main idea or (ii) you got the main idea but were not precise enough in stating it; the grader will assume that *you did not get it*.

6. When a problem asks for a proof, we ask you to divide up your proof into two parts: the first (typically smaller) part will spell out the main idea(s) in your proof and the second (typically longer) part will contain all the details of the proof. For example, the first paragraph in the proof of (1.3) in the textbook is the proof idea, while the second paragraph has the entire proof. By default, the "Proof idea" part will be worth at least 50% of the points for the proof. We reserve the right to give you a 0 if you only present the proof details without the proof idea (even if your detailed proof is correct). On the flip side, if you make a mistake in the details of your proof, you are guaranteed a certain fraction of the grade if you had the right proof idea.

   Also a proof written in English is much easier to understand than one that is heavy with mathematical notation. For this reason, we ask you to write your proofs in as much English as possible.

7. An algorithm that is stated in English is easier to understand than one written in pseudocode. However, pseudocode is sometimes better, especially for a precise definition. We ask that you provide an English description/summary of your algorithm and then provide the pseudocode if necessary. We reserve the right to deduct points if you just present the pseudocode of the algorithm but do not provide any English description.

8. The only results that you are allowed to quote without proof are those that appear in the book (or were covered in class). Solved exercises in the book are considered as such results but unsolved exercises are *not*. All other results that you use in your homework must be proved. (Also results that you have seen in your earlier courses such as discrete math and data structure are permissible. However, for such results you are *required* to give a reference.) Using a result other than those above, will result in deduction of points. To be on the safe side, it is always good to give a reference for any result that you might be using.

9. Typically we will not take in re-grading requests, especially if it is of the form of not agreeing with the partial credit given. For each homework, the TA will provide the grading rubric that was used, so please consult that first before going to the TA with a grading question. However, if there is a genuine reason for re-grading (e.g. your correct algorithm was marked as incorrect), please contact the TA within *a week* of when the graded material is handed out in class. In particular, if you do not pick up your graded material on time, you lose the opportunity to get back to us within the stipulated time period.

10. If you are stuck thinking about a problem on the homework, we *strongly* encourage you to go and talk to the instructor and/or the TA and/or post a comment on the blog. However, please be aware that you might not get a response if you contact us after 5pm on the day before the homework is due.

11. No homework will be accepted after **1:15pm** the lecture it is due. This time deadline *will be strictly enforced*.

Here are some suggestions for you. Following these are not mandatory but we strongly encourage you to do so. (Interspersed are some general comments related to homeworks.)

1. **Start early!** The homeworks will be challenging, so start thinking about the homework problems early. Just reading a problem and letting it "marinate" in your head helps a lot. In particular, please do *not* start working on the homework the night before it is due.

2. Working with algorithms is an art and the only way you can get better (and hence, do well in the exams) is to practice. Homework problem are the minimal practice you should do. We encourage you to work on other exercises in the book. If you do work on a problem and have a question, come and talk to us.

3. Typically, the homework will have three problems: the first one being the easiest, the second being medium hard and the final one being the hardest problem. However, note that this is just our guideline and what is easy for us might be hard for you and vice-versa. Also the homeworks in general will get harder as the semester goes on (as the material will become more involved). Hence, the easy problem on HW 10 might be harder than the hard problem in HW 1.

4. We will assume that you will collaborate in your homeworks (when permitted). Thus, when we think of a question as being hard, it is assumed that it will need a fair bit of work *even as a group* to solve that problem. Typically, the easy problem should be solvable if you understand the lecture. Note that there will be no "plug and chug" kind of questions in the homework (e.g. run the algorithm on this specific input), so you might have to spend some time outside of the lecture to really understand the material covered in class.

5. In a homework or exam if you do not follow why you lost some points, go and talk to the grader and try to figure out where you went wrong. We do not really like it when someone comes and is only interested in getting more points but we like it when someone comes by and wants to understand what went wrong and improve.

6. Typically, in the homeworks (and the exams), the easy, medium hard and hard problem(s) will be worth 40%, 45% and 15% of your grade respectively.

7. Another plug for working on the material outside the lectures: the exams will not test what was taught in the lectures verbatim. One of the goals of this course is to get you to think abstractly and algorithmically about problems. This cannot be force-fed but only comes about when you do the algorithmic thinking on your own. Understanding what I say in a lecture is a start but it is by no means a sure-fire way of really understanding the material (One way to find out if you understand the material– teach it to someone else!) Working on problems is one way to build your algorithmic thinking.

8. Many of the problem statements will be pretty verbose and will not have crisp mathematical description. As we will see in the lectures, this is unavoidable in real life and we want you to have practice in looking at a problem statement in English and then converting it

into a precise mathematical problem. In real-life there are red herrings: information in the "problem statement" that have nothing to the real mathematical problem. So in some homework problems, we will slip in some red herrings!