

CS 280 – Discrete Structures

HW 3 – Grading Guide

General Notes

- Please be neat, especially with your names. **Write your name in uppercase letters only.**
- **Always explain** what you are doing and why you are doing it. Even if an explanation is not required, you may at least get partial credit if you provide an explanation.

Grading guide section 1.8

4. 3 points max, -1 for each error

14. 6 points max, 1 for each part. -0.5 for yes/no answer with no proof, -0.5 for correct answer but wrong proof

22. 6 points max, 1 for each part. 0.5 for Θ , 0.5 for Ω . No proof required

38. 5 points max. Any reasonable answer will get up to 3 points. A correct proof with only C's or k's gets 4 points. A correct proof with both C and k gets 5 points. -0.5 if all the C's are the same, -0.5 if all the k's are the same.

Comments:

Remember that yes/no/true/false questions will generally also require some proof for full credit.

Don't rephrase the question and use that as the proof. For example, if asked to prove x^3 is $O(x^4 + x^2)$, don't just say $O(x^3) = O(x^4 + x^2)$.

Try to use definitions/theorems in the book/from class. Saying something is obviously correct without substantiation may cause you to lose points.

Make sure you're doing the right problems.

Section 2.1

26) Total points: 5

Full points awarded if you got the general idea through, even if it wasn't very efficient or had typos. Some people lost 1 point if they made an updating error (*while* loops don't update the index themselves, *for* loops require you to do this anyway so people that used *for* loops didn't make this error). Others lost 1 point for unclear code or explanation. A few people used the *else* clause, which only caused problems for their programs; they normally lost just 1 point for this. Remember to use variable names that make sense.

Also, in algorithms that only require pseudocode like this one and like many more you'll see in this and subsequent CS theory courses, instead of going in to the details of how you are updating a filler array, it's easier to just say, "add so-and-so element to bottom of so-and-so list", or instead just print the element on the screen if storage is not required.

This just makes your life easier.

Section 2.2

4) Total points: 5

2 points for explaining the runtime of each procedure, total 4 points. 1 point for comparing the runtimes and saying which is better. The 1 point was also awarded to people that said which procedure was more efficient, even if their above runtimes were incorrect.

14a) Total points: 5

Induction wasn't required and almost no one used induction here. This question was graded on an individual basis, most people receiving full credit. A key point that you needed to mention is that all elements *must* be checked because any element could be the max element. People lost 1 point if they failed to mention this. Many people didn't really prove their answers, but got the general idea, which luckily was very close to the proof, and so they received full credit. The easiest way to prove this is to use a *catch-up* argument, i.e. to show that however good any optimal algorithm is, this one is at least as good as the optimal, hence making this also optimal. (You will see more of these proofs in CS 482, but harder). Proof by example received 0 points.

14b) Total points: 5

1 point for saying that it is optimal. 4 points for the proof, which was really a short explanation in this case. Many people gave the following *incorrect* reasoning and lost 1 or 2 points: "it is optimal because there is only 1 operation inside the loop and all elements must be checked". It is true that all elements must be checked, but even if there were 5 operations inside the loop, the runtime would be $O(5n) = O(n)$. So the first part of the above reason is incorrect and was penalized.

20) Total points: 5

Your answer to this question was checked with your algorithm from 2.1.26. 3 points for your runtime in Big-O notation, and 2 points for analysis/explanation of how you calculated the runtime. If you said how many operations were performed (such as $2*n$), but didn't convert this to Big-O notation ($O(n)$ in the above case), you lost 1 point. If your answer to this question was very far from $O(n)$, such as constant time or infinite time (or zero time), you probably got very few points, if any.

Section 2.3

2.3: Total score = 20

2.3.12: Sub-score = 5

3 marks for correctly identifying that the two factors that result in trailing zeroes are 2 and 5.

If the number of trailing zeroes (i.e. 24) is correctly calculated, 2 marks are awarded

Otherwise, if the answer is off by 1 (23, 25) because the student makes a careless mistake and forgets to include 100, for example, only 1 mark is awarded.

2.3.16: *Sub-score* = 5

(a) 1 marks awarded for showing that the factors of 6 (other than itself) add up to itself.
Another 1 mark awarded to showing the same thing for 28.

(b) 2 marks awarded for listing/showing the factors of $2^{p-1}(2^p-1)$.
Another 1 mark awarded for showing that these factors (other than $2^{p-1}(2^p-1)$) add up to itself.

2.3.40: *Sub-score* = 5

1 mark awarded for each part (a) – (d).

1 additional mark awarded if at least one question is answered correctly.

2.3.48: *Sub-score* = 5

3 marks awarded for establishing the equation needed to solve the problem.

Another 2 marks awarded if the correct answer is obtained ($Q = 8$)

Note: Some students misinterpreted the question and simply add up at the digits of the ISBN to check whether it is a multiple of 11. At most 3 marks are awarded for such misinterpretations of the question.