

Analysis of Algorithms
CS 375, Spring 2021

Concept-Heavy Intensive Problem Set (*CHIPS*)

Due in part by Friday, May 7, and
in part by Wednesday, May 12, and
in part by Friday, May 14.

(Please see the notes about deadlines and descriptions of exercises below!)

- There are three parts to this Concept-Heavy Intensive Problem Set. Two of the parts are to be done fully independently; the third part is to solve exercises as a team and to meet with me to get feedback, discussing your solutions.

Please note that at least one meeting with me is required as part of this assignment. *After meeting with me, your team will be able to submit revisions for your exercises; only the final revised version will be considered for your grade*

- Your team is welcome to go through this meeting-revision cycle more than once, so please submit work early and make appointments for meetings with me early! No work may be accepted, however, after Colby's deadline of 9pm on May 14.
- You may choose your own team for this assignment (preferred team size: 3–4 people). Please let me know if you would like my help in finding teammates!
- Detailed instructions for each of the three parts are given in the Overview sections next, which include submission instructions and deadlines for each. CS375 standards and conventions for filenames, submission, etc., continue to apply.
- Please recall the guidelines for algorithms given on previous HW assignment sheets. They continue to apply.
- In general, there may be multiple correct ways of presenting an algorithm, although excessively inefficient or inelegant solutions may not receive full credit. If you have questions about whether your proposed solution is excessively inefficient or inelegant, please ask your Prof.!
- *A general note for CS375:* Please present answers cleanly and **explain them clearly and thoroughly**, giving all details needed to make your answers easy to understand; typed-up (rather than handwritten) answers are especially appreciated. (Feel free to talk with your Prof. about using L^AT_EX to typeset your answers!) Graders may not award full credit to incomplete or illegible solutions. Clear communication *is* the point, on every assignment.

In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

1 Overview

Before getting to the exercises on your CHIPS (presented in Section 2), here is an overview of what to do and submit for each of the three parts of your CHIPS.

1.1 Problem-Solving Strategies

- This first part of your CHIPS is to be done **fully independently**, without talking with anyone about it (except for asking me for clarifications on something).
- This part is intended to be completed and submitted before talking with your teammates about the exercises below. Therefore, its deadline is early, and it is a hard deadline.
- This part will be graded on a binary basis—full credit if you’ve made a substantial effort; 0 if you give answers that do not reflect a substantial effort—and will be worth 10% of your CHIPS grade.

For the first part of your CHIPS, before actually talking with your teammates about any of the problems / exercises below, please read over the exercises and for each one, write down **your own, independently generated** problem-solving strategies for that exercise—i.e., write down *how* you would go about solving it. If you have multiple ideas and aren’t sure which to use, write them all down! If you don’t have any ideas for how to solve a problem, that’s okay too—please say more than just that you don’t know how to do an exercise, though; instead, go into what’s confusing, or what parts you don’t understand, or what specific lecture notes or textbook sections you might use but aren’t sure how to apply.

This does not need to take very long—you don’t need to do the exercises, or even know how to do the exercises, to complete it! Just fully explain your thought processes and current states of understanding at the beginning of your CHIPS. (As you’ll see later, this will be a foundation for the reflection in Section 1.3, as described below.)

I expect that this will require at least a short paragraph for each exercise—writing just a sentence or so wouldn’t go into full detail of your process for an exercise—although you are welcome to write more than that, if you’d like!

Reminder: These are your independent thoughts and ideas, and this part of your CHIPS is to be done **without talking with anyone about it**. I know that this is an unusual requirement, and it will be relaxed soon! But because this is graded on a binary scale, with no need for you to do an exercise or even know how to—this is just a recording of your starting ideas—there is no need to discuss it with anyone to get full credit, so please do this part fully independently (except, as always, asking me clarifying questions as needed).

What to submit For this part of your CHIPS, please submit a PDF file called

`cs375s21_CHIPS_strategies_<username>.pdf`

to your SubmittedWork folder (filling in your username, as usual).

Deadline This part of your CHIPS must be submitted by 11:59pm Friday, May 7. (All times given in this document are local to Maine.) To ensure your team can move forward with the exercises, that is a hard deadline; submissions after that deadline will have a lateness penalty of 25%. (As always, if there are extenuating circumstances, please let me know!)

1.2 Exercises; Meetings; Revisions

Section 2 below contains the exercises for your CHIPS. Please do the exercises *as a team*—you are expected to talk as a team about the exercises and how you might solve them, and for my meeting with your team, every teammate should be prepared to explain their team’s solutions.

Your team will submit a single PDF document containing all of your work for all of the exercises (as you’ve done with all the problem sets this semester)—typewritten solutions are greatly appreciated!

Your team also **must meet with me at least once** (see below for more information) as part of completing this assignment.

Meetings, feedback, and revisions! Your team may submit your exercises, schedule a meeting with me to receive feedback, and then submit a revision of your work; after a “designated submitter” member of the team submits their work to their **SubmittedWork** folder, please email me to let me know it’s submitted and to set up a meeting. (I intended to meet with teams, rather than individuals, to preserve the overall team structure of the assignment.) Moreover, your team can go through this cycle *as many times as you’d like*, submitting as many revisions as you’d like, before the final deadline on May 14. I will be happy to give your team feedback to help you improve your work on each successive revision—I look forward to talking with you!

What to submit For this part of your CHIPS, your team should submit a single PDF file called

`cs375s21_CHIPS_<usernames>_revision<n>.pdf`

to a “designated submitter’s” **SubmittedWork** folder. That is, only one member of the team needs to submit a file. Please make sure that the `<usernames>` portion of your filename contains the underscore-separated userids (no numbers needed) of every teammate; the `<n>` after the word “revision” in the filename should specify the revision number being submitted. So, for example, my revision 1 with my teammate Stephanie Taylor would be in file `cs375s21_CHIPS_eaaron.srtaylor_revision1.pdf`.

Deadline For this part of your CHIPS, your first version must be submitted by 11:59pm on May 12; every version must be submitted at least 24 hours before meeting with me to discuss it. To satisfy the assignment, your team must meet with me at least by 9:00pm Friday, May 14 (the end of the allocated timeslot for CS375 on Colby’s Final Exam schedule). This is a hard deadline; submissions after that deadline may incur severe lateness penalties. (As always, if there are extenuating circumstances, please let me know!)

1.3 Reflection

- This last part of your CHIPS is to be done **fully independently**, without talking with anyone about it (except for asking me for clarifications).
- As with the first part, this part will be graded on a binary basis—full credit if you’ve made a substantial effort; 0 if you give answers that do not reflect a substantial effort. This part will be worth 5% of your CHIPS grade.

As part of the learning experience of these exercises, please reflect on the work you did with your teammates on your CHIPS. Here are some guiding questions that you might consider as part of your reflection:

- What did you learn specifically from working with your teammates on these exercises? What specific parts of your group discussions of each exercise did you find most illuminating or helpful?
- How were the approaches used by your team similar to what you initially expected? How were they different?
- Are there things regarding these exercises that you would still like to understand more, even after finishing your CHIPS?

This is not an exhaustive list—please feel free to reflect on and include other aspects of your process in your write-up for this part of your CHIPS, too!

I expect that this will require at least one or two short paragraphs for the CHIPS overall—less than that wouldn’t be fully reflective of your working together as a team on the exercises—although you are welcome to write more than that, if you’d like!

Once again, as with the first part of your CHIPS, these are intended to be your independent thoughts and ideas, and this part of your CHIPS is to be done **without talking with anyone about it**. Because this is graded on a binary scale, with no need for particular technical knowledge—these are your reflections, not problem solving—there is no need to discuss it with anyone to get full credit, so please do this part fully independently (except, as always, asking me clarifying questions as needed).

What to submit For this part of your CHIPS, please submit a PDF file called

`cs375s21_CHIPS_reflection_<username>.pdf`

to your SubmittedWork folder.

Deadline This part of your CHIPS must be submitted by 9:00pm Friday, May 14 (the end of the allocated timeslot for CS375 on Colby’s Final Exam schedule). This is a hard deadline; submissions after that deadline may incur severe lateness penalties. (As always, if there are extenuating circumstances, please let me know!)

2 Exercises

This section contains the exercises to be solved as a team. Instructions for submitting them are given above. As always, feel free to be in touch with questions!

1. Express the running time (as a function of n) of the following pseudocode algorithm using Θ -notation.

```
r = 0
i = n * n
while i > 1 do
    for j = 1 to i do
        r = r + 1
    i = i/2
```

In the above, $i/2$ is integer division. To simplify your analysis, you may assume that n is a power of 2.

Be sure to give a detailed answer, and explain all details behind your answer: state what *barometer operation* you counted to arrive at your answer and explain what made that a good choice for a barometer operation; show all work that you did to get *an exact count* of those operations and arrive at a summation that expresses the running time; and explain how you got from that summation to the Θ complexity bound, giving all details—for *this exercise only on your CHIPS*, please include all of the low-level details (including *witness* values for c_1, c_2, n_0 used in your reasoning), rather than doing a high-level complexity analysis.

2. An algorithm to compute 2^n would be defined by the following specifications:

```
// Input: n, a non-negative integer (i.e., 0 or greater)
// Output: 2^n (i.e., 2 to the n'th power)
```

- (a) Write a recursive algorithm for computing 2^n for any non-negative integer n **that is based on the formula** $2^n = 2^{n-1} + 2^{n-1}$. (This may not be the most natural way to think of an algorithm to compute 2^n , but please be sure to use it for this exercise!)
- (b) Set up a recurrence for the *number of additions made by the algorithm* on input n and solve it **using the recursion tree method**.

Please show your work by including a table of the form illustrated on the problem set for the Recurrences course module; as you did then, your table for this exercise should include one row for each level in the tree and a representative k 'th level to illustrate the relevant pattern, similar to the k 'th step in the unwinding method. Then, explain how you use information from your table to arrive at a final answer. (You are also welcome to include a picture of the recursion tree as part of your answer, if you'd like.)

Based on your solution to this recurrence, what is the Θ complexity class of this algorithm?

- (c) Is this a good algorithm for computing 2^n ? Please explain your answer! (A couple of sentences could well be a sufficient explanation for this exercise.)
3. Consider the following pseudocode for the sorting algorithm **Crazy-Sort**:

```

Crazy-Sort(A)
  if ( $|A| > 1$ ) then
    Crazy-Sort(1st third of array A)
    Crazy-Sort(2nd third of array A)
    Crazy-Sort(3rd third of array A)
    Crazy-Merge(3 sorted thirds of array A)

```

- (a) Assume that **Crazy-Merge** takes time $\lg n$ to merge the 3 thirds of an array A of size n . Under this assumption, write down a recurrence for the running time of **Crazy-Sort**. (You do not need to solve this recurrence, just write it down.) As always, explain your answer.
- (b) Now, instead, assume that **Crazy-Merge** takes time n to merge the 3 thirds of an array A of size n . Under this assumption, write down a recurrence for the running time of **Crazy-Sort**, and solve the recurrence using a solving method of your choice. Please explain your answer and show your work: If you use unwinding, show how you arrive at the pattern you find; if you use the recursion tree method, show how you sum up work on the levels of the tree; if you use the Master Theorem, state the case you are using and why; etc.
4. This semester in CS375, our problem set for the Asymptotic Complexity course module included the following question as a “slight digression,” with no answer to be submitted at the time:

Do you wonder which is *asymptotically* more efficient, the iterative Insertion Sort, or the recursive Insertion Sort? Or are they in the same asymptotic complexity class? Please **do not submit an answer to this question**, but it’s an interesting and important question, and we’ll learn how to answer it when we cover methods for analyzing the complexity of recursive algorithms—coming soon to CS375!

As part of your CHIPS, please answer that question! Which is asymptotically more efficient, the iterative Insertion Sort or your recursive Insertion sort? Or are they in the same asymptotic complexity class? Please explain your answer thoroughly; for complexity analysis, however, high-level analysis of asymptotic complexity bounds are sufficient for this exercise.