Problem Set 3

This problem set is due **at 10:00pm** on **Wednesday, February 26, 2020**.

Please make note of the following instructions:

- This assignment, like later assignments, consists of *exercises* and *problems*. **Hand in solutions to the problems only.** However, we strongly advise that you work out the exercises for yourself, since they will help you learn the course material. You are responsible for the material they cover.
- Remember that the problem set must be submitted on Gradescope. If you haven't done so already, please signup for 6.046 Spring 2020 on Gradescope, with the entry code MNEBKP, to submit this assignment.
- We require that the solution to the problems is submitted as a PDF file, **typeset on LaTeX**, using the template available in the course materials. Each submitted solution should start with your name, the course number, the problem number, your recitation section, the date, and the names of any students with whom you collaborated.
- You will often be called upon to "give an algorithm" to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of your essay should provide the following:
  1. A description of the algorithm in English and, if helpful, pseudocode.
  2. A proof (or indication) of the correctness of the algorithm.
  3. An analysis of the asymptotic running time behavior of the algorithm.
  4. Optionally, you may find it useful to include a worked example or diagram to show more precisely how your algorithm works.

EXERCISES (NOT TO BE TURNED IN)

Union-Find

- Do Exercise 21.3-4 in CLRS on page 572.

Problem 3-1. Cam's Books [60 points]

Cam Petitive and Opal T. Mull have to do readings for a class, but they dread going to the library!

In typical MIT style, the library owns m books which students refer to by number, from 1 to m. Their professor gives the name of a book, b, each time she assigns a reading. If a student has the book checked out, they complete the reading swiftly and happily (cost 0). Otherwise, they have to walk across campus to check the book out from the library (cost 1). The library has lots of copies of each book, so students don't have to worry about reserving books or book shortages. The problem is, each student is only allowed to have k books checked out at any time, so students who have k books checked out must return one of their previously checked-out books when checking out a new one.

Opal has somehow managed to figure out not only the n readings, but also the optimal sequence of books to return, to minimize trips to the library. Cam hasn't done the same, but he thinks he can compete with Opal without knowing the readings or doing such complicated calculations.

Note that the professor seems to enjoy repetition as a teaching tool, and doesn't hesitate to assign old readings. At the beginning of the semester, both Cam and Opal have zero books checked out.

(a) [15 points] The library doesn't set explicit due dates, so Cam thinks he can exploit this by holding on to his books for as long as possible. He works out the following strategy for each reading b :

- If b is already checked out, he does not go to the library, and completes the reading swiftly and happily.
- Otherwise, he checks out b from the library as follows:
  - If Cam hasn't reached his limit of k books checked out yet, he doesn't return any books.
  - Otherwise, he returns the book with the *most recent* check out date.

Prove that Cam's strategy is not competitive with Opal's if $k \geq 2$. That is, prove that there are input sequences of length n such that the ratio of Cam's to Opal's library trips tends to ∞ as n tends to ∞.

(b) [25 points] The semester has progressed (both Opal and Cam have the same k books checked out), and now Cam thinks he has a better idea. Cam keeps his k library books stacked up on his desk, to remember the last time each was used for a reading. Whenever a reading is assigned, if Cam has the book already, he slides it out, does the reading, and places it on the top of the stack. If Cam has to go to the library, he slides out the bottom book from the stack, returns that one, and puts his newly checked-out book on the top of the stack when he gets back.

Use the potential method to prove that Cam's strategy is k-competitive with Opal's. That is, for any sequence of reading assignments of length n, the ratio of Cam's to Opal's library trips is no more than k.

Hint: At any time let S_{CAM} be the set of books that Cam has checked out, and let S_{OPT} be the set of books that Opal has checked out. Use the potential function

$$\Phi = \sum_{b \in S} w(b)$$

where $S = S_{CAM} \setminus S_{OPT}$ is the set of books that Cam has which Opal doesn't, and $w(b)$ is how high up book b is on Cam's shelf, with the bottom book's height $w(b_{bottom}) = 1$ and the top book's height $w(b_{top}) = k$.

(c) [5 points] Which aspect of the proof for part (b) does not work if you were to try to prove Cam's original strategy is k-competitive with the same potential function?

(d) [15 points] Cam makes a friend in the class, Amy Ortez. Amy is intrigued to see that her book-return strategy is very similar to Cam's. Amy keeps her books on a shelf instead of in a stack. If she needs to go to the library, she slides out the rightmost book from the shelf, returns it, and inserts her newly checked out book to the left of her other books upon returning. Whenever a book is assigned which she already has checked out, Amy slides it out, does the reading, and then places it back where it was located on the shelf.

Use the potential method to prove that Amy's strategy is also k-competitive with Opal's.

Problem 3-2. Catering Challenge [30 points]

App Bonatite is catering for pavilion festivities at MIT. They know the total amount of food they will bring, and have a list of all the students who are attending, but they aren't sure how much food to portion for each student. So they reach out to some contacts at the dining halls, who report back relative amounts of what students eat. For instance, it might be observed that student A eats 1.5 as much as student B. No one keeps record of specific quantities of food eaten; only *ratios* between how much different students eat are observed. App Bonatite wants to know when they will have enough information to set up all the food for the pavilion festivities, or at least be able to quickly report how much information they currently have.

More formally: suppose we have some student s. Let $f(s)$ denote the amount of food s should get (assume some consistent unit of food throughout this problem). Devise a data structure which can handle the following operations.

- *Update*(s_1, s_2, x): Given two students s_1 and s_2 such that $f(s_1)/f(s_2) = x$, update the data structure with this knowledge.
- *GetRatio*(s_1, s_2): Given two students s_1 and s_2, return their relative food ratio $f(s_1)/f(s_2)$ or "insufficient information" if this information is not yet known.
- *SetupFood*(s, f): Given a student s, and the amount of food f they eat, report whether or not there is sufficient information to set up all the food. (In particular, if App Bonatite knows the ratios between all students, then they can set up all the food).

Design and describe a data structure that can handle the *Update*, *GetRatio* and *SetupFood* operations in amortized $O(\alpha(n))$ time.

Problem 3-3. Feedback Form [10 points] Please fill out a feedback form about this problem set at

`https://forms.gle/2BN8VGMChPYFBpwX7.`

It should not take more than a few minutes and will greatly help us improve teaching and material for future semesters!