Problem Set 2

This problem set is due **at 10:00pm** on **Wednesday, February 19, 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)

Divide and Conquer Algorithms

- Do Exercise 4.2-3 in CLRS on page 82.
- Do Exercise 9.3-1 in CLRS on page 223.

Amortized Analysis

- Do Exercise 17.1-3 in CLRS on page 456.
- Do Exercise 17.2-3 in CLRS on page 459.
- Do Exercise 17.3-7 in CLRS on page 463.

Union-Find

- Do Exercise 21.3-3 in CLRS on page 572.
- Do Exercise 21.3-5 in CLRS on page 572.

Problem 2-1. WageRank [45 points] After another successful year of Googol, Parry Lage prepares to negotiate his salary with the board of Counting Numbers Inc. As the co-developer of the WageRank algorithm, Parry believes himself capable of modeling the behavior of the board members and predicting his salary even before the meeting. The procedure, as he understands it, is that each of the 100 board members has drawn an optimal salary uniformly at random from the range $[0, 1000000]$ in dollars, not necessarily independently. Parry can propose any salary he desires, and each board member has one vote, which will be cast in favor of the proposal if the proposed value is no greater than this board member's optimal salary and will be cast against the proposal otherwise. If at least 51 of the 100 board members cast votes in favor of Parry's proposed salary, then this salary will be approved. Otherwise, Parry risks being fired! Parry wonders how likely it is that he can secure a proposal of at least $\$200000$, an *extremely* modest salary given all of his hard work.

(a) [8 points] Use Markov's Inequality to provide a bound on the probability that Parry will fail in his proposal of a salary of $\$200000$. In addition, solve for the highest salary that Parry can propose while keeping his probability of failure at most $\frac{1}{2}$.

(b) [7 points] Now, one of Parry's most trusted advisors has come to Parry and shared an important piece of information. Specifically, she informs Parry that the board members have chosen the salaries pairwise independently. Use Chebyshev's Inequality to provide a tighter bound on the probability that Parry will fail in his proposal of a salary of $\$200000$.

Hint: Recall that two random variables X and Y are pairwise independent if, and only if, $\Pr(X = x, Y = y) = \Pr(X = x) \cdot \Pr(Y = y)$.

(c) [5 points] Now, Parry, having co-developed WageRank, insists that he can make an even stronger assumption about the board members—namely, that they all select the salaries mutually independently. With this new assumption, use the Chernoff Bound to provide a yet tighter bound on the probability that Parry will fail in his proposal of a salary of $\$200000$.

(d) [25 points] Just as Parry believes that he's begun to work out the right salary to propose, he is informed by one of his advisors that the format of the board meeting is not what he believed. The board size is not fixed at 100 but rather includes n board members, and Parry can only propose one salary to the board, so he needs to choose very wisely. Additionally, the power dynamics on the board have shifted, and some board members have more votes than others. Specifically, suppose the board members $x_1, x_2 ... x_n$ each have $w_1, w_2, ... w_n$ votes respectively. You may assume the total number of votes, W, is odd.

Parry, understanding the intricacies of WageRank, has held a private meeting with every board member to ask the board member what a reasonable CEO salary is. He knows that board member x_i thinks s_i is the ideal salary for Parry. A board member will use all their votes to vote against Parry's proposed salary if it is higher than their wishes, and the board member will use all their votes in favor of the proposed salary if the proposal is less than or equal to their wishes. You may assume the s_i are distinct.

Devise an algorithm to determine the highest salary Parry can propose. For full credit, your algorithm should run in $\mathcal{O}(n)$. Partial credit will be given to an $\mathcal{O}(n \log n)$ solution. You may cite algorithms shown in lecture without proof.

Problem 2-2. Zark's Dynamic Photos [45 points]

Zark Muckerburg, having just purchased the photo-sharing app Delaypound (DP), realizes that with so many users creating and deleting content, he'll need a more adaptable storage array. Zark wants to support two operations: appending image data at the end of the array and deleting the image data from the end of the array when a photo is removed.

In particular, he begins with an array of length one and inserts the first element into the one free space in this array. In general, if at any point Zark has a full array of length n and wants to append an element, he creates a new empty array of length $n + 1$, copies the n elements from the length-n array to the first n entries of this new array, and then inserts the new element into the last space. If Zark wants to delete the last element from an array of length n, he creates a new empty array of length $n-1$ and copies the first $n-1$ elements from the length-n array to the $n - 1$ entries of this new array.

Suppose that copying each element when resizing the array has a cost of 1, as does adding a single element, or deleting a single element *when the array is not being resized*.

(a) [10 points] Calculate the cost of performing n insertions followed by n deletions. Given this cost, find the amortized cost per operation.

(b) [10 points] Zark is unsatisfied by this naïve storage method, so he puts his top engineers on the problem, and they propose an alternative solution. They suggest that, whenever the array reaches capacity, rather than being copied to a new array that is one unit longer, it can be copied to one that is *twice* as long. In particular, to append an element to a full length-n array, Zark can create a new empty array of length $2n$, copy the n elements from the length-n array to the first n entries of this new array, and then insert the new element into the $(n + 1)$th space.

To handle deletions, the engineers suggest that an array can be shrunk if the number of elements reduces to one-half the length of the array. In particular, if the array has length n but is only populated by elements in the first $\frac{n}{2} + 1$ slots and Zark wants to delete an element, then he can create a new empty array of length $\frac{n}{2}$ and copy the first $\frac{n}{2}$ elements from the length-n array to the $\frac{n}{2}$ entries of this new array.

The engineers purport that their algorithm achieves the desired amortized $\mathcal{O}(1)$ cost per operation. Propose an adversarial sequence of operations in which the amortized cost per operation is $\Omega(n)$.

(c) [25 points] Zark sees some potential to the solution that his top engineers propose, but is still unsatisfied, so he recruits you to see if an amortized $\mathcal{O}(1)$ cost per operation can truly be achieved. You, being a hardworking algorithms student, realize that although the doubling scheme for appending elements is spot-on, the halving scheme is a bit ineffective. Instead, you propose that Zark should only halve the size of the array if the number of elements reduces to one-quarter of the length of the array. In particular, if the array has length n but is only populated by elements in the first $\frac{n}{4}+1$ slots and one element is to be deleted, then he can create a new empty array of length $\frac{n}{2}$ and copy the first $\frac{n}{4}$ elements from the length-n array to the first $\frac{n}{4}$ entries of this new array.

Use the potential method to demonstrate that this implementation provides an amortized cost of $\mathcal{O}(1)$ per operation, as desired.

Hint: use the potential function $\Phi(m,n) = \max\{2m-n, \frac{n}{2}-m\}$, where m is the current number of stored elements and n is the current capacity (length) of the array.

Problem 2-3. Feedback Form [10 points] Please fill out a feedback form about this problem set at

`https://forms.gle/2j2iGmkDKqDiXLM89`.

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