CS 161 **Problem Set 3**
Spring 2023 **Due:** Wednesday, April 26 at 1pm on Gradescope.

Style guide and expectations: Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we are looking for in homework solutions. We will grade according to these standards.

Make sure to look at the "**We are expecting**" blocks below each problem to see what we will be grading for in each problem!

Collaboration policy: You may do the HW in groups of size up to three. Please submit one HW for your whole group on Gradescope. (Note that there is an option to submit as a group). See the "Policies" section of the course website for more on the collaboration policy.

Exercises

We recommend you do the exercises on your own before collaborating with your group.

---

1. **(6 pt.)** In this exercise, we'll explore different types of randomized algorithms. We say that a randomized algorithm is a **Las Vegas Algorithm** if it is always correct, but the running time is a random variable. We say that a randomized algorithm is a **Monte Carlo Algorithm** if there is some probability that it is incorrect. For example, QuickSort (with a random pivot) is a Las Vegas algorithm, since it always produces a sorted array (but if we get very unlucky QuickSort may be slow).

   Suppose that there is a population of n flamingos. Half of the flamingos are pink, and half are white, but it's dark outside and you can't tell the difference until you catch one and look at it with a flashlight. Assume that catching a random flamingo and looking at it takes time $O(1)$.

   The algorithms given in Figure 1 all attempt to find a single pink flamingo. Fill in the chart below. You may use asymptotic notation for the running times; give the best big-Oh bound that you can. For the probability of returning a correct flamingo, do not use asymptotic notation; give the best bound that you can.

Algorithm	Monte Carlo or Las Vegas?	Expected running time	Worst-case running time	Probability of returning a pink flamingo
Algorithm 1				
Algorithm 2				
Algorithm 3				

   Note that the LaTeX code for the table is available in the source file.

   [**Hint:** *Remember (see Lecture 5 and the pre-lecture exercise) that the expectation of a geometric random variable with probability p is equal to $\frac{1}{p}$*]

   [**We are expecting:** *A filled-in table. No justification is required.*]

Algorithm 1: findPinkFlamingo1

Input: A flock of n flamingos
while *true* **do**
    Choose a random flamingo from the flock;
    if *That flamingo is pink* **then**
        return *that flamingo*
    else
        Release the flamingo back into the flock

Algorithm 2: findPinkFlamingo2

Input: A flock of n flamingos
for *100 iterations* **do**
    Choose a random flamingo from the flock;
    if *That flamingo is pink* **then**
        return *that flamingo*
    else
        Release the flamingo back into the flock
Choose a random flamingo from the flock;
return *That flamingo*

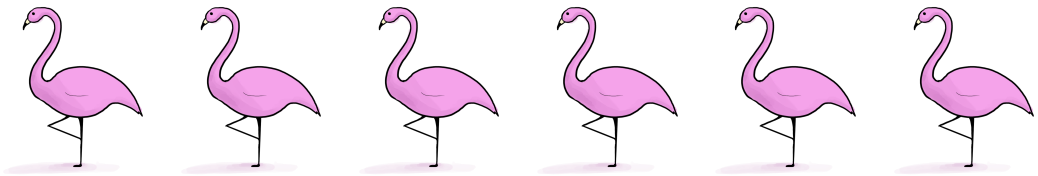
Algorithm 3: findPinkFlamingo3

Input: A flock of n flamingos
Put the flamingos in a line, in a random order ;
`/* Assume it takes time` $O(n)$ `to put the` n `flamingos in a random order; and assume that they stay put once ordered. */`
for $i = 0, \ldots, n-1$ **do**
    if *Flamingo i is pink* **then**
        return *Flamingo i*

Figure 1: Three algorithms for finding a pink flamingo

Problems

2. **[Flamingo Politics] (6 pt.)** Suppose that n flamingos are standing in a line.



Each flamingo has a political leaning: left, right, or center. You'd like to sort the flamingos so that all the left-leaning ones are on the left, the right-leaning ones are on the right, and the centrist flamingos are in the middle. You can only do two sorts of operations on the flamingos:

Operation	Result
`poll(j)`	Ask the flamingo in position j about its political leanings
`swap(i,j)`	Swap the flamingo in position j with the flamingo in position i

However, in order to do either operation, you need to pay the flamingos to cooperate: each operation costs one brine shrimp.[1] Also, you didn't bring a piece of paper or a pencil, so you can't write anything down and have to rely on your memory! Like many humans, your memory is limited, and you can only remember up to seven[2] integers between 0 and $n-1$ at a time (i.e. you can use at most seven integer-valued variables at a time in your algorithm).

Design an algorithm to sort the flamingos, which costs $O(n)$ brine shrimp, and uses no extra memory other than storing at most seven[3] integers between 0 and $n-1$.

[**Hint:** *Does this task look like anything we've seen in class?*]

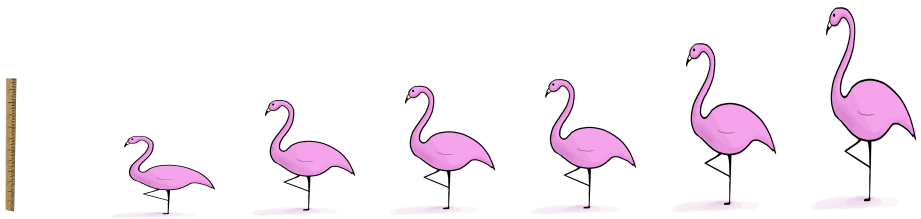
[**We are expecting:** *Pseudocode* ***AND*** *a short English description of your algorithm;* ***AND*** *a short explanation of why it uses only $O(n)$ brine shrimp and never uses more than seven numbers of memory.*]

[1] According to Wikipedia, flamingos eat brine shrimp. Yum!

[2] `https://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two`

[3] You don't need to use all seven storage spots, but you can if you want to. Can you do it with only two?

3. **[Flamingos of new heights] (6 pt.)** Suppose that n flamingos of distinct heights are standing in a line, ordered from shortest to tallest.



You have a measuring stick of a certain height, and you would like to identify a flamingo which is the same height as the stick, or else report that there is no such flamingo. The only operation you are allowed to perform is `compareToStick(f)`, where f is a flamingo (that is, you cannot directly access the heights of each flamingo). `compareToStick(f)` returns `taller` if f is taller than the stick, `shorter` if f is shorter than the stick, and `the same` if f is the same height as the stick. As in the previous problem, you can only store up to seven integers in $\{0, \ldots, n-1\}$ at a time. And, as in the previous problem, you have to pay flamingos one brine shrimp every time you perform `compareToStick`.

(a) **(2 pt.)** Give an algorithm in this model of computation which either finds a flamingo the same height as the stick, or else returns "No such flamingo," and uses $O(\log(n))$ brine shrimp.

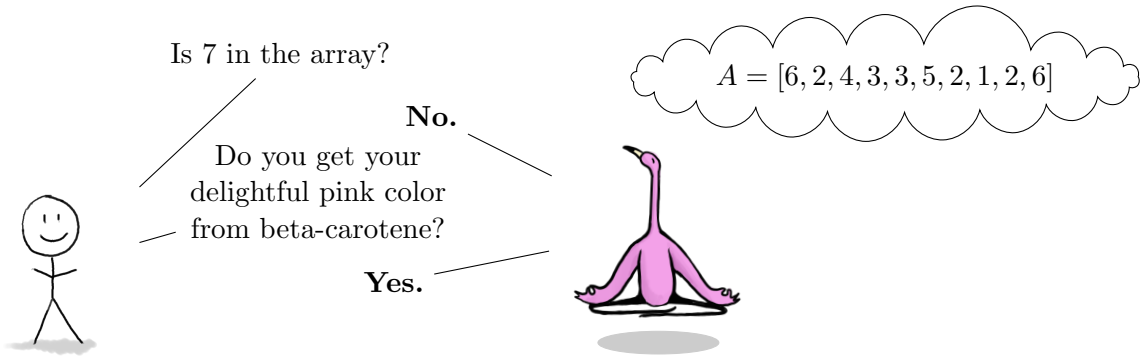
[**We are expecting:** *Pseudocode* ***AND*** *an English description. You do not need to justify the correctness or brine shrimp usage.*]

(b) **(4 pt.)** Prove that any algorithm in this model of computation must use $\Omega(\log(n))$ brine shrimp.

[**We are expecting:** *A short but convincing argument.*]

4. **[Wisdom of Flamingos] (5 pt.)** A wise flamingo has knowledge of an array A of length n, such that $A[i] \in \{1, \ldots, k\}$ for all i. (Note that the elements of A are not necessarily distinct). You don't have direct access to A, but you can ask the wise flamingo *any* yes/no questions about it. For example, you could ask "If I remove $A[5]$ and swap $A[7]$ with $A[8]$, would the array be sorted?" or "do molecular and morphological studies support a relationship between grebes and flamingos?"

   Unlike in the previous two problems, this time you did bring a paper and pencil, and your job is to write down all of the elements of A in sorted order.[4] As usual, the wise flamingo charges one brine shimp per question.



   (a) **(5 pt.)** Give a procedure that outputs a sorted version of A which uses $O(k \log(n))$ brine shrimp. You may assume that you know n and k, although this is not necessary.

   [**We are expecting:** *Pseudocode AND a short English description of your algorithm;* ***AND*** *a brief explanation of why it uses $O(k \log(n))$ brine shrimp.*]

   (b) **(1 BONUS pt.)** Prove that any procedure to solve this problem must use $\Omega(k \log(n/k))$ brine shrimp.

   [**We are expecting:** *Nothing; this part is not required.*]

[4] Note that you don't have any ability to change the array A itself, you can only ask the wise flamingo about it.

5. **[Please (Flamin-)Go Fill Out This Group-Work Survey] (1 pt.)**

   In collaboration with Professor Shima Salehi of the Stanford Ed School and her team, we have created a short google form on your experience with group-work in this class, which is linked below. SUNet IDs will be collected, but we will only use hashes of IDs for the study, so the responses will be effectively anonymous. The form is very short, and will take ≈ 1 minute to fill out: `https://forms.gle/PzTZpMBaoDjrzNdA7`.

   We really appreciate all your feedback and help with making CS161 the best possible learning experience it can be! If you would like to give more general feedback on the class as a whole, please check out the many possible options highlighted in the linked Ed post[5].

   Have you submitted your answers to the google form above?

   [**We are expecting:** *Yes*]

[5] `https://edstem.org/us/courses/38246/discussion/2970522`