CIS 121 — Data Structures and Algorithms
Homework Assignment 3

Assigned: September 22, 2020 **Due:** September 29, 2020

Note: The homework is due **electronically on Gradescope** on September 29, 2020 by 11:59 pm ET. For late submissions, please refer to the Late Submission Policy on the course webpage. You may submit this assignment up to 2 days late.

A. **Gradescope**: You must select the appropriate pages on Gradescope. Gradescope makes this easy for you: before you submit, it asks you to associate pages with the homework questions. Forgetting to do so will incur a 5% penalty, which cannot be argued against after the fact.

B. **LaTeX**: You must use the LaTex template provided on the course website, or a 5% penalty will be incurred. Handwritten solutions or solutions not typeset in LaTex will not be accepted.

C. **Solutions**: Please write concise and clear solutions; you will get only a partial credit for correct solutions that are either unnecessarily long or not clear. Please refer to the Written Homework Guidelines for all the requirements. Piazza will also contain a complete sample solution in a pinned post.

D. **Algorithms**: Whenever you present an algorithm, your answer must include 3 separate sections. Please see Piazza for an example complete solution.

1. A precise description of your algorithm in English. No pseudocode, no code.
2. Proof of correctness of your algorithm
3. Analysis of the running time complexity of your algorithm

E. **Collaboration**: You are allowed to discuss **ideas** for solving homework problems in groups of up to 3 people but *you must write your solutions independently*. Also, you must write on your homework the names of the people with whom you discussed. For more on the collaboration policy, please see the course webpage.

F. **Outside Resources**: Finally, you are not allowed to use *any* material outside of the class notes and the textbook. Any violation of this policy may seriously affect your grade in the class. If you're unsure if something violates our policy, please ask.

1. [20 pts] CIS 121 TAs Take On Restaurant Week

Seeing all the fancy restaurants participating in Philly's Restaurant Week, a group of n CIS 121 TAs (you may assume that n is a power of 2) decide to go into Center City for a glorious 3-course meal.

The n TAs each have a favorite restaurant, and they will only be able to attend restaurant week if they can figure out a restaurant that is a favorite of a strict majority (i.e., is a favorite for strictly more than $n/2$ of the TAs).

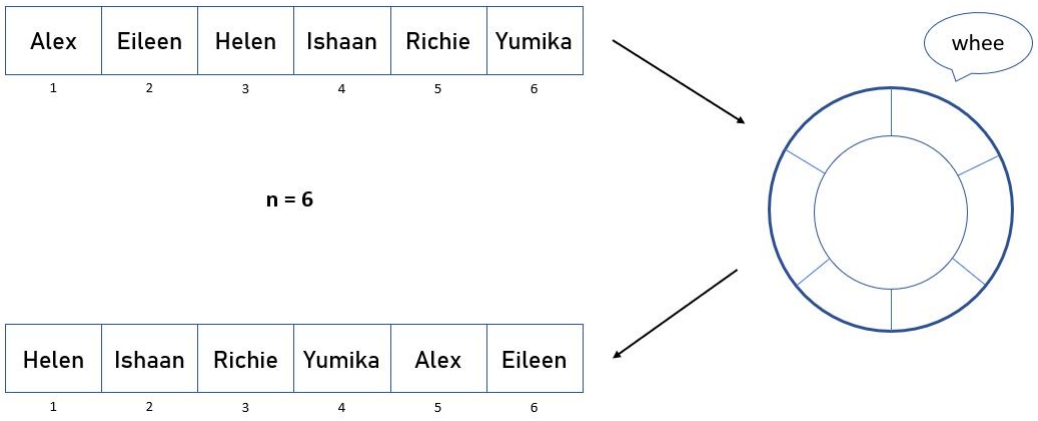
Sherry, who unfortunately cannot go, is tasked with organizing this dinner. The other TAs provide Sherry with one array $A[1..n]$ containing each of their respective favorite restaurants. For instance, if $A[i] =$ *Harper's Garden*, then the i^{th} TA prefers the restaurant *Harper's Garden*. Sherry wishes to figure out if there exists a favorite restaurant of strict majority. However, all Sherry is able to do is pick two TAs and determine (in constant time) whether or not they have the same favorite restaurant.

Help Sherry by designing a $O(n \lg n)$ divide-and-conquer algorithm that, given the array of restaurants $A[1..n]$, returns a favorite restaurant of strict majority if such a restaurant exists, and `Nil` otherwise. Make sure to include a proof of correctness and runtime analysis.

2. [20 pts] Trip to *Six DAG's*

Kostas takes n CIS 121 TAs to *Six DAG's*, a local amusement park, for a group field trip. Once they arrive at a ride, The TAs are initially alphabetically ordered single file in line, with the first alphabetical TA in spot 1 and the last alphabetical TA in spot n. For example, Alex would be in spot number 1, because his name is first alphabetically, and Yumika would be in spot number n, since her name is last alphabetically.

The ride is circular in shape, and the TAs sit down one by one around the circle. The ride spins the TAs around many times, and after it ends, the TA closest to the door exits the ride back in line, with each subsequent TA following behind in single file. Note that Alex may no longer be at the front of the line, but the TAs are still in relative alphabetical order. Consider this final line of TAs.
Example:



a. To help Kostas in re-locating TAs, develop a $\Theta(\lg n)$ time algorithm to find the position of a certain TA with name x. For instance if x is "Ishaan", then your algorithm must return 2. Suppose that all TAs have distinct names. You are given x, and you know that x is guaranteed to exist in this line.

b. Suppose instead that multiple TAs may have the same name - for instance, there may be two Andrew's. Note that although two different TAs may have the same name, you can return any TA with the input name x. Does your algorithm in (a) still meet the target runtime *in worst case*? Explain why or why not in a small paragraph - you do not need to rederive an algorithm.

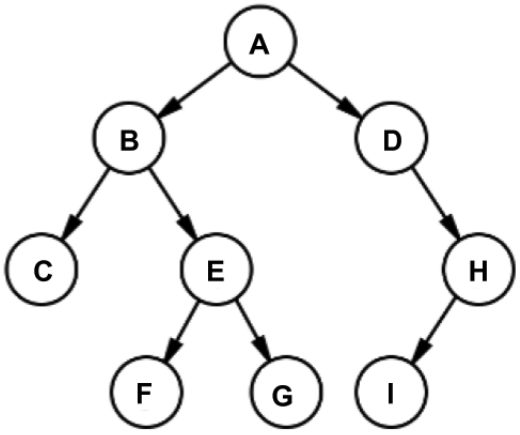
3. [20 pts] Tiffany's Ice Cream Adventure

Tiffany is on a hunt for the best ice cream in Philadelphia, so she decides to visit every ice cream shop in the city. Philadelphia's ice cream shop scene is set up as a binary tree of size n. Help Tiffany determine in which order she should visit the ice cream shops.

a. Tiffany initially plans her ice cream itinerary as such: at each subtree rooted at shop r, she visits r, then traverses the shops in r's left subtree, and then traverse the shops in r's right subtree. Print out the order that in which she visits the shops in $\Theta(n)$ time and $O(n)$ space **without using recursion**. No solution that uses recursion will be considered for points.

b. Another possible way she could visit the shops is the following: At each subtree rooted at shop r, she traverses the shops in r's left subtree, then visits r, and then traverse the shops in r's right subtree. Print out the order of this new itinerary in in $\Theta(n)$ time and $O(n)$ space **without using recursion**. No solution that uses recursion will be considered for points.

Example:

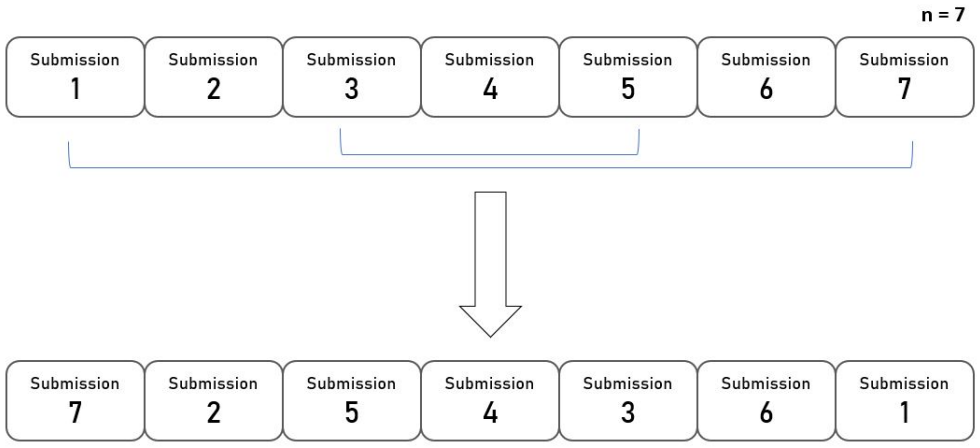


For a), the order would be A, B, C, E, F, G, D, H, I
For b), the order would be C, B, F, E, G, A, D, I, H

4. [20 pts] Dana's PennApps Puzzle

As PennApps XXI comes to a close, Dana is tasked with organizing submissions for grading. She finds that there have been a total of n submissions, with $n \geq 2$. The submissions are labeled $1, 2, ..., n$ and are stored in a bin B that acts as a queue. In order to make grading fairer, she decides to shuffle the submissions in a certain order. Specifically, she wants to reverse the order of only the odd-indexed submissions. For instance, she wishes to swap the first card with the last odd position, then swap the third card with the card in the second-to-last odd position, etc.

Example:



Dana asks her Sponsorship Committee for funds but realizes that she can only afford one stack S and one queue Q (separate from the original bin B), both of which are initially empty. Help Dana derive an algorithm that shuffles the order of the submissions in B in $O(n)$ time and $O(1)$ extra space outside of the provided stack and queue. Be sure to justify the runtime and space complexity of your algorithm. You may assume that each stack/queue operation takes $O(1)$ time. No proof of correctness is required.

5. [20 pts] Around the World in 121 Days

Kostas is sending the n CIS 121 TA's to n locations across the world to spread the gospel of data structures and algorithms. Each TA has a **distinct** climate preference for their location assignment and will not go to any location that is more tropical or more cool than that exact climate preference. Kostas is a nice guy and picked locations so that there is exactly one location for each TA, selected to their exact preferred climate. However, when choosing locations, Kostas forgot to write down which location belongs to which TA. He also doesn't have any way to quantify the climate of each location, and none of the TAs know how to verbalize their preferences and/or organize amongst themselves. **The only test that Kostas has is to ask some TA about a particular location, and the TA will tell him whether the location is too tropical, too cool, or just right for them.**

In this problem, we want to design an efficient algorithm that Kostas can use to reconstruct the original location assignments. You can assume that each test takes a constant amount of time and that there

are an equal number of TAs and locations as noted. For the purposes of probabilistic analysis, you may assume that a given TA is equally likely to prefer any of the n possible locations.

(a) Consider the following algorithm to figure out which location is for which TA. Let T denote the group of TAs and L be the set of locations. What is the *expected* running time of this algorithm? What is its *worst case* running time? Justify your answer.

```
LOCATIONS(T, L):
if |T| = |L| = 1 then:
    Assign the location to the TA
else:
    Pick a location l uniformly at random from L
    T' = T
    L' = L
    for each TA t in T do:
        Kostas asks TA t about location l
        if it is just right then:
                        Give location l to TA t
                        T' = T' − {t}
                        L' = L' − {l}
                        break out of the for loop
    LOCATIONS(T', L')
```

(b) Design an algorithm with a better bound than the one in part (a). Be sure to explain your algorithm well, analyze its expected running time, and analyze its worst case running time.

If you'd like (for this part only!) you may provide properly-formatted pseudocode to **support** your English explanation of your algorithm. That is, you still **must** provide an English explanation. You can easily format pseudocode using the `verbatim` environment.

(c) Suppose the following: each time Kostas asks a TA about a location, the TA says whether or not the location is just right without giving any information on whether it is too tropical or too cool. Does the algorithm in part (a) still work? What about the one in part (b)? Give a brief (1-3 sentence) justification for your answer.

[0 pts] Feedback: No feedback form for this assignment. Feel free to reach out to the Head TAs if you have any comments/concerns.