Analysis of Algorithms
CS 375, Fall 2022
Small Assignment 2
Due **BY THE BEGINNING OF CLASS** Wednesday, October 12

- This Small Assignment will be graded on effort: Make a demonstrable, strong effort to solve the problems and explain your answers thoroughly—including, if you were not able to solve an exercise, what progress you made, what was left unsolved, and what made it hard to solve—and you will receive full credit!

- We will go over this exercise in our next class meeting. Because of that, on-time submission of this assignment is essential! Any submission after the beginning of class (1:00pm on the deadline date) will get a grade no higher than a ✓−.

- For this Smaller Assignment, the standard file naming conventions apply: Please submit your typewritten answers in a PDF file named

  `CS375_SA2_<userid>.pdf`

  where `<userid>` is replaced by your Colby userid (your full userid, including class year)—for example, my file would be called `CS375_SA2_eaaron.pdf`—and submit it to your `SubmittedWork` folder in your Google drive space for this course. Please reach out to me right away with any questions or concerns about this!

- *A general note for CS375:* As always, 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. or TA's about using LaTeX 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!

Exercises

1. Give an *iterative* (not recursive) algorithm that returns a list of all permutations of elements of an input list.

   ```
   # Input: List L = [a_1, ..., a_n]

   # Output: List L' = [P_1, ..., P_n!] of all permutations of L.
   #         That is, each P_i is a permutation of L, i.e., P_i is a
   #         list containing exactly the elements of L, and no two
   #         P_i have the elements of L in the same ordering.
   ```

   As part of your process in designing this algorithm, be sure to think of a *loop invariant* for the algo's outer loop, as was demonstrated in the lecture notes of our Oct. 3 class. (You do not need to take the formal approach in our textbook of considering *initialization, maintenance,* and *termination*; just have a description of what is true

each time through the loop, as we did in class, and use it in the algo's design and explanation of correctness.) There may be multiple possible loop invariants that would work well.

For full credit, please be sure to do the following as parts of explaining your algorithm's correctness:

(a) Use your loop invariant as part of your explanation

(b) Explicitly, directly refer to the algo's specifications as part of establishing correctness

In addition, please give the most helpful asymptotic complexity bounds you can on both the time and space complexity of your algorithm, using O, Θ, or Ω notation. As always, explain how you arrived at your answers for these complexity bounds.

Please turn in your best effort for this exercise! As stated in the instructions at the top of this assignment sheet, even if for any reason you aren't able to complete the assignment exactly as given, please explain your thought process fully and describe the parts of the problem you hadn't fully solved, and turn in the best work you can by the deadline.

NOTE: As we discussed in class, there are $n!$ permutations of a list with n elements. It is taken as a definition that $0! = 1$, so we'd say the empty list has 1 permutation. A list with one element also has only 1 permutation. Please check your algorithm on boundary cases such as these!