CS41 Homework 11

This homework is due 11PM Tuesday December 6.

Submit this homework using github as a LaTeX file. For this homework, you will work with a partner. It’s ok to discuss approaches with others at a high level, but most of your discussions should just be with your partner. The only exception to this rule is work you’ve done with a lab partner while in lab. In this case, note who you’ve worked with and what parts were solved during lab. If there are questions about academic integrity, please visit the section on Academic Integrity on the course website.

Note: Do not be put off by the multi-part nature of this assignment. Several parts to these problems have simple, short solutions. Formal proofs are not necessary for this assignment, but you should provide convincing arguments.

0. Before final submission, make sure to fill out the README file.

1. Warm-up analysis. Prove the following asymptotic statements. I’ve provided a solution for the first one:

   (a) Show that \( n! = 2^{\Omega(n \log n)} \).

   Solution. Consider the following crude inequalities:
   \[
   n! = \prod_{i=1}^{n} i > \prod_{i=n/2}^{n} i > \prod_{i=n/2}^{n/2} n/2 > (n/2)^{n/2} = 2^{(n/2) \log(n/2)},
   \]
   Also, note that \( n/2 \log(n/2) = n/2 \log(n) - n/2 = \Omega(n \log n) \). Therefore, \( n! = 2^{\Omega(n \log n)} \).

   (b) Show that \( (k!)^{n/k} = 2^{\Omega(n \log k)} \).

   (c) Let \( S_{n,k} \) be the set of all possible arrays of length \( n \), where the first \( k \) items consist of the numbers 1, \ldots, \( k \) in any order; the second \( k \) items consist of the numbers \( k+1, \ldots, 2k \) in any order, and so on. For example, if \( A = [3, 1, 2, 6, 5, 4] \) and \( B = [6, 1, 3, 2, 4, 5, 1] \), then \( A \in S_{6,3} \) but \( B \) is not in \( S_{6,3} \).

   Show that \( |S_{n,k}| = (k!)^{n/k} \).

2. Close-to-sorted Revisited. Recall from Homework 2 that a array of numbers is “\( k \)-close-to-sorted” if each number in the array is less than \( k \) positions from its actual place in the sorted order. (So a 1-close-to-sorted array is actually sorted.)

   (a) In Homework 2, you gave an \( O(n \log k) \)-time algorithm for sorting a array of numbers that is \( k \)-close-to-sorted. Show that this is optimal for any comparison-based algorithm by proving a matching \( \Omega(n \log k) \) lower bound.

   Hint: use an encoding argument and statements from problem 1. You may assume these statements are true and correct even if you haven’t completely solved them.

   (b) Now, consider the problem of \( k \)-close-to-sorting an array. Here, you’re given an arbitrary \( n \)-element array, and you must return \( k \)-close-to-sorted array that contains the same elements. Give an \( O(n \log n) \)-time algorithm that \( k \)-close-to-sorts a array.
(c) Show that if \( k = O(\log n) \), then any comparison-based algorithm that \( k \)-close-to-sorts a array must make \( \Omega(n \log n) \) comparisons.

**Hint:** do *not* use an encoding argument.

3. **Lower Bounds for Search.** Recall that binary search allows you to find an item in an \( n \)-element sorted array and return its index by inspecting only \( O(\log n) \) elements of the array and comparing them to the item you’re searching for.

Show that this is tight by giving a matching \( \Omega(\log n) \) lower bound for any comparison-based algorithm that searches a sorted array.