Programming Assignment 1 DRAFT

COMP526: Efficient Algorithms
Fall 2024

DUE Wenesday, 13 November by 11:59pm

Background

Sorting is a fundamental algorithmic task. The basic sorting problem is as follows: given an array a (or Python `list`) of numbers, order the elements of a from smallest to largest. For example, we might have the following input and output:

In this assignment, the goal is to sort arrays using only prefix reversal operations. Here, a ***prefix reversal*** at index i reverses the order of elements up to index i in the array. For example, starting from the array $a = [1,2,3,4,5]$, the prefix reversal reverse$(a,2)$ returns the array $[3,2,1,4,5]$.[1]

Given an arbitrary array a, a ***sorting prefix reversal sequence*** is a sequence of prefix reversals that, when applied to a, results in a sorted array. For example, starting with the array $a = [4,1,3,2]$, we can apply prefix reversals to sort a as follows:

$$\begin{aligned}[4,1,3,2] &\overset{3}{\longmapsto} [2,3,1,4]\\ &\overset{1}{\longmapsto} [3,2,1,4]\\ &\overset{2}{\longmapsto} [1,2,3,4].\end{aligned}$$

Since prefix reversals applied were $3,1,2$, the prefix reversal sequence $[3,1,2]$ is a sorting prefix reversal sequence. This sequence has ***length*** 3.

The ***cost*** of a sorting prefix reversal sequence is measured by the length of the sequence. Thus, the goal of the prefix reversal sorting problem is to find a sorting prefix reversal sequence of minimal length.

Assignment Description

For this assignment, you will write a program that performs the prefix reversal sorting task. That is, you will implement functions that compute sorting prefix reversal sequences from a given array. While simple algorithms exist that will compute sorting

[1] Note that array indices start at 0, so that $a[2]$ is the *third* element in the array.

prefix reversal sorting sequences from any array, your goal is to sort the arrays with the smallest possible cost (i.e., length of sorting prefix reversal sequences).

Specifically, you will write 4 functions, each of which computes a sorting prefix reversal sequence for arrays generated according to one of the four families:

- *Uniformly random permutations*: The array a contains each value from 1 to n exactly once in a random order.
- *Random tritonic permutations*: The array a contains each value from 1 to n exactly once, and there are indices a, b with $0 < a < b < n-1$ such that a is
  - *increasing* between indices 0 and a,
  - *decreasing* between indices a and b, and
  - *increasing* between indices b and $n-1$,
- *Random binary sequences*: the array contains only values 0 and 1, and each value is chosen with equal probability, independent of other values, and
- *Random ternary sequences*: the array contains only values 0, 1, and 2, and each value is chosen with equal probability, independent of other values.

It is possible to write a single function that sorts arrays for all of these families, however you should think about how to exploit the given structure of the given array to find a lower cost prefix reversal sorting sequence.

Program Specification

Your program must be written in Python. Two program files are provided:

- `pr_sorting_tester.py` contains code to generate sequences from the families above and test the output of your sorting procedures. Your final program will be tested using a (nearly[2]) unaltered version of this file, and **any changes you make to this file will *not* be applied in testing your submission**.

  This program imports four functions from `pr_sorter.py` which you must implement yourself (see below). For each of the four functions below, the tester generates `num_tests` (set to 5) arrays of length `test_array_size` (set to 50). Thus, the total number of arrays tested is 20. The program verifies that each of the four functions correctly produces sorting prefix reversal sequences for the 5 test arrays, and sums the total number of reversals performed to sort all arrays. This final value is output when running the program.

- `pr_sorter.py` will contain your implementations of the four functions tested. Each of the following functions takes as input a `list` of `int`s, which represent arrays of integer values to be sorting, and must output another `list[int]`, the sequence of prefix reversals that sort the input.

[2]When I test your program, I will use a different random seed so that the actual sequences tested will be different from the ones generated by the provided testing program.

- `pr_general_sort` This function will be tested on uniformly random permutations. It should be capable of generating a sorting prefix reversal sequence for any given input.
- `pr_tritonic_sort` will be tested only on tritonic input sequences. Thus, you may assume that any sequence passed to this function is tritonic.
- `pr_binary_sort` will be tested only on binary inputs.
- `pr_ternary_sort` will be tested only on ternary inputs.

Your primary task is to complete the function definitions above to perform the prefix reversal sorting task. You may define any additional functions you would like. However, you must not alter the function declarations in the provided file, and your program must run with the original, unaltered version of `pr_sorting_tester.py` provided.

Attribution and Collaboration

It is expected that you take personal responsibility for all code you submit. You are encouraged to seek outside resources (e.g., scholarly research articles) to deepen your understanding of the problem and for general reference for the Python language.

- **You must cite all external resources used in the comments of your submitted program.** A simple list of article title, author (if applicable), and link to the resource suffices—you need not use any particular bibliographic format. If you implement an algorithm described in the literature, you must additionally indicate which portion of your code contains the implementation.
- **You will not receive credit for code you did not write**. If you use a small code snippet found in some external resource (i.e., anything you didn't write yourself), you *must* cite the original source of the code and indicate that the included code is not yours where it appears in the program.
- **Generative AI should not be used to write any substantial part of your program.** If you use generative AI to help write small code snippets, you must cite the source of the code as you would code taken from any other resource.
- **You should not share your code or view other students' code directly. Evidence of sharing code will be treated as a breach of academic integrity and will be reported to the university.** You may discuss high level, conceptual approaches to solving the prefix reversal sorting problem with your peers in class. You should not discuss implementation details with others.

Resources

Your submission must be completed in the Python programming language. If you do not have (a recent version of) Python already installed on your machine, you can learn how to install Python here:

- `https://wiki.python.org/moin/BeginnersGuide/Download`

If you are new to Python (or want a refresh on the basics) you may find the following website helpful:

- `https://www.pythoncheatsheet.org`

Scoring

Your final mark for the programming assignment will be determined by the correctness and efficiency of your solution. The final mark will be on a scale from 0 to 100.

- Your program must correctly sort all arrays generated by the sorting program. If your program fails on *any* array, your mark of the assignment is 0.
- Your program must be reasonably efficient in its computation. I will test your program on my personal computer (2023 MacBook Pro) with a timeout of 2 minutes. If your program does not complete within 2 minutes, you will also receive a mark of 0.
- There is a simple algorithm that sorts all arrays of length n using fewer than $2n$ prefix reversals. Running this algorithm on all of the challenge inputs from the tester program results in a total of roughly 1350–1450 total prefix reversals (depending on which random seed is used to generate the sequences). In order to receive a passing mark (≥ 50 points) your program should significantly improve upon this baseline by using fewer than $1,200$ prefix reversals.
- If all of the conditions above are met, marks will be awarded based on the number of prefix reversals used on the arrays generated by the testing program: fewer total prefix reversals will result in higher marks.