Divide & Conquer Homework

In this assignment you will use the provided code template to implement solutions to each given problem. The solutions you submit must follow the guidelines provided in this document as well as any given in code.

Your solutions must implement a divide & conquer approach to each given problem. Both recursive and iterative implementations are accepted.

When using a known divide & conquer algorithm, only those covered in course material (see Ed) are allowed, and some are provided for you to use.

Restrictions

- The use of built-in `sort` capabilities are strictly prohibited.
- Template code is provided and must be used.
  - ***Note: The templates are refreshed each semester. Be sure to use the current version.***
- Your code must be compatible with **Python 3.10**.
- Do not add imports for any libraries beyond the Python standard library.
- Do not modify the function name or definition.

Testing

For each problem, one or more base test cases are provided. A properly implemented solution will pass any provided base test cases, but their trivial nature is not intended to ensure the overall correctness of your algorithm.

These are the same test cases that will be used to provide assurance that your submitted solutions run using the autograder. We do not release any other test cases.

In problem instances where more than one solution is optimal, *any* optimal solution will be considered correct, assuming that is the solution your algorithm produces.

The test cases can - and should! - be expanded while developing your solutions.

You can run your test cases by issuing the `python3 -m unittest` command from the directory containing your solution files.

Submission

Submit only - and all of - your solution files (i.e., **cs6515_[problem_name].py**) to the Gradescope assignment on or before the posted due date. Do not submit a zip file or any other files except those matching the naming convention defined.

You may submit your solutions as many times as desired and the most recent submission is what will be graded. Execution of your code may take up to 60 minutes. If not completed before the due date, your previous submission will be used.

Faster and correct solutions are worth more credit.

After your submission completes execution, you will see a score of `- / 20` listed until grading is completed.

A Friendly Competition

The CS6515 Head TAs are trying to decide who knows their students best.

After a fierce debate, Jamie declares that he can line up all the students in descending order of their current grades.

The other Head TAs claim they can accomplish the same feat, so a friendly competition is started.

As you know from the first missing natural number D&C problem, if a single number is out of place then all subsequent numbers are out of place, so that's not a good way to determine who is most correct.

Instead, we will count the number of pairs in the form of:

- if arr[a] $<$ arr[b] and a $<$ b

In other words, a count is added for every pair of students such that student 'a' has a lower grade than student 'b' and student 'a' is listed before student 'b'.

Develop an efficient divide & conquer algorithm to count the number of such pairs.

Notes:

- The output is an int representing the count of such pairs.
- If arr[a] $<$ arr[b] $<$ arr[c] and a $<$ b $<$ c then there are three such pairs.

Implement your algorithm for solving this problem in **cs6515_friendly_competition.py**.