TASK:

you need to apply some concepts about metaheuristics, specifically local search and evolutionary algorithms to solve optimization problems

DOCUMENTATION :

Submitting the code and the tests is not enough. In fact, in addition to well commented code and test cases, to successfully complete Assignment 3 you must also commit a file called SOLUTION.MD.

Inside SOLUTION.MD you must describe in words and possibly using diagrams your solution.

Do not copy, paste, and comment your code inside SOLUTION.MD; this is not want your are asked here! Instead, elaborate about your solution and the main choices you made to complete it.

**Problem Description**

You need to implement the following two search algorithms:

* Steepest-Ascent Hill Climbing with Random Restart
* Genetic Algorithm

and use them to solve the "Review Assignment" **combinatorial** optimization problem.

**The Review Assignment Problem:**

At University , first semester INF Bachelor students***S*** must write an essay on a given topic ***t***, that they choose among a predefined set ***T*** of topics.

To pass the class, the students must also review ***r*** essays written by they classmates.

To avoid cheating, students cannot review their own essays. Additionally, to avoid large clusters of students working on similar essays, no more than ***c*** students can be assigned to the same topic ***t***.

Each student must express a preference on each of the available topics using a value in the **range *[-p, 0, +p]* (*-p* and *p* included)**. Specifically**, *p*** is an integer value, negative values indicate the un-willingness to review the essays on a topic, whereas positive values indicate the willingness to review an essay on a topic.

To solve the Review Assignment problem, you need to search for assignments of students to essays that maximize the overall preference score, where the preference score is computed summing the preference value of each assignment.

**Write Testable Code**

To improve the testability of your code you should implement the search algorithms in a **generic way**, i.e., they should not be problem specific and provide as inputs, and encapsulate the problem specific details in well defined functions.

The repository contains already sample functions and public tests that illustrate how you could write testable code.

So your job to complete the task is to: implement the overall search algorithms and the missing functions that encode the details of the Review Assignment Problem.

**Files:**

**Public tests**

The public folder contains all public tests that are visible to everyone. **This folder should be considered untouchable** so that any additional changes can be made without harm to your project.

**Personal tests**

All tests that you write for yourself should be placed in this folder. The contents of this folder will only be changed by you.

**Private tests**

All private tests that will be used for grading will be placed under this folder. **This folder also should be considered untouchable**

**Usage of Libraries and Existing Classes/Functions**

System libraries, plotting libraries, numpy, pandas, and data science related libraries can be used. However, the usage of libraries that implement search algorithms and metaheuristics is allowed ONLY for testing purposes. That is, you can write tests that invoke those libraries to check your solution, but you do so at your expense (you need to understand how to use them). For testing, you can also use the usual testing libraries: pytest and pytest-\*.

NOTE: If you plan to use additional libraries, remember to list them inside the requirements.txt file!

**Short list of existing libraries for search and evolutionary algorithms**

NOTE: This list was taken from

* [genetic algorithms](https://pypi.org/project/geneticalgorithm/)
* [DEAP](https://github.com/DEAP/deap)
* [JMetalPy](https://jmetal.github.io/jMetalPy/index.html)
* [PyGAD](https://pygad.readthedocs.io/en/latest/)
* AND MANY MORE

For linear combinatorial problems, you can also (at your own risk) use libraries that use integer linear programming,

**Passing Requirements**

You need to commit the following files:

* SOLUTION.MD
* The python scripts implementing your solution
* Personal tests (named after the usual test\_\*.py naming convention).
  + Unit tests: testing specific functionalities/units of code
  + System tests: testing that the whole algorithms can find a reasonably good solution to known functions. You can code your own test objective function (e.g., max ones/sum of binary) or refer to existing test functions [1],[2].
  + [1] https://www.sfu.ca/~ssurjano/optimization.html
  + [2] https://en.wikipedia.org/wiki/Test\_functions\_for\_optimization

NOTE: We do not enforce coverage anymore as precondition, but your solution will be tested using private tests. So please, make sure you test your solution and commit all the tests.