COP3503 Computer Science 2
Dr. Andrew Steinberg
Spring 2023

Programming Assignment 3
The Cookie Problem
Due: 3/26/2022 at 11:59pm

Objective: Students will apply concepts of greedy algorithm design.

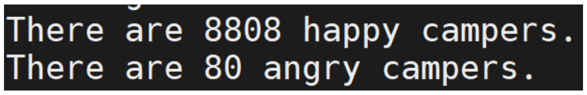
Assignment Description:

You currently work at a summer camp and are a counselor for n campers. Every afternoon is cookie time where the office drops off m cookies for your camp group (you can assume that $m > n$). The summer camp advises only 1 cookie can be given to prevent a sugar rush. The campers cannot share the same cookie. The m cookies that were dropped off are not all the same in size. These cookies can be measured by a size factor represented as $s_j, 0 \leq j \leq m$. The campers in your camp group have a level of greed. This can be measured by a greedy factor as g_i, $0 \leq i \leq n$. It is your job to maximize the number of happy campers in your group with the proper cookie. <u>Your goal is to come up with a greedy algorithm that finds the optimal solution along with an efficient implementation that runs in $O(mlgm)$ time.</u>

For this assignment, you must follow these **requirements**.

1. You create your solution defined in a class called `Cookies`.
2. You have been provided with 10 text files. Each text file contains a numerical value representing the greedy or size factor. Files named `cookies` represent the size factor and files named `campers` preset the greedy factor.
    a. The number after the file names is associated with the test case.
3. Your class should have the following attributes:
    a. An integer array that stores the greedy factor of each camper.
    b. An integer array that stores the size factor of the cookie.
    c. An integer representing the number of happy campers that will get the cookie that satisfies their greedy factor.
    d. An integer representing the number of angry campers that will get the cookie that doesn't satisfy their greedy factor.
4. The `Cookies` Class has an overloaded constructor that has four parameters.
    a. The first parameter represents the number of cookies you will hand out.
    b. The second parameter represents the number of campers.
    c. The third parameter represents the name of the txt file containing the greedy factor levels of each camper.
    d. The fourth parameter represents the name of the txt file containing the size factor of each cookie.
5. The `Cookies` Class has a public non static method called `read`. `read` will scan the respective text files and store the values in their respective array.

6. The `Cookies` Class has a public non static method called `passCookies`. This method will solve the problem of the assignment. **Here is where you will implement your greedy solution in maximizing the number of campers getting the desired cookie.** This method **must run** in the worst case $O(mlgm)$ where m represents the number of cookies. This is already called in the provided driver file for each case we test. Your solution must also use the greedy techniques that have been presented in lecture to solve the problem. **Any other techniques will result in no partial credit for the passCookies method criteria in the rubric. The graders will double check this!**
7. The `Cookies` Class has a public non static method called `display`. This method will display the results `passCookies` computed. Simply, you will display the result of the attributes representing the happy and angry campers. Each result will be displayed on a separate line. Here is an example.



```
There are 8808 happy campers.
There are 80 angry campers.
```

A driver file (`CookiesDriver.java`) has been provided for you to show you how the methods are called along with 5 test cases to see if you get the same scenario result.

What to submit: Submit a file called `Cookies.java` to webcourses. You are not required to submit the driver file as that will be provided for the graders to test your code. Please make sure the driver file provided works for your code. Any name changes may cause your program not to work when graded, which will result in a lower score on the assignment and would not be changed. You do not need to submit the text files of moves.

Assumptions that can be made: Here is a list of assumptions that can be made in the assignment.

1. The number of cookies to hand out will always be greater than the number of campers.
2. It is possible that some campers can have the same greedy factor.
3. It is possible that some of the cookies can have the same size factor.
4. Each camper can only receive one cookie.
5. Cookies **<u>cannot</u>** be broken into fractions (like in the fractional knapsack problem). You can only give a whole cookie.

Comment Header: Please make sure to provide the appropriate comment header (like in the Eustis assignment) as the first thing on the top of your file. Otherwise, if your comment header is done incorrectly including not being placed literally at the top of the file will result point deductions in accordance with the rubric.

Theoretical Constraint: In this assignment your main method that will compute the solution must run theoretically in $O(mlgm)$. If your solution does not follow this theoretical constraint, **then a score of 0 is applied as the grade for the assignment with no other partial credit from any components of the rubric.**

Time Constraints: For this assignment, your code solution must entirely run within 1 second on Eustis. If your solution takes longer than 1 second, then a score of 0 will be applied to the assignment grade with no partial credit. The test script already has timeout exception set so you can test your code's time constraints. As long as you follow the theoretical constraints, your code should be able to run fast!

Text Files: All text files are placed in the directory as your driver file and solution file. DO NOT CREATE SUBFOLDERS! If you create subfolders, then this will cause our batch grader to not run your code properly. This will result in points being deducted that cannot be disputed. You can assume that all proper moves are associated and that there are no invalid characters in the text files.

Provided Text Files: The input text files have been provided for you in a zip folder. You will see a total of 10 files. 5 files represent the number of campers and 5 files represent the number of cookies. The numbers at the end represent their associated test case.

Important Note for running Eustis and Packages: Many of you are probably using IDEs like Netbeans and Eclipse to build your Java Solutions. Please note that some of these IDEs will automatically place your Java file in some sort of package. Please make sure your Java file is not defined in some package as this can result package private errors. Any such error that occurs during the grading will not be fixed and points will be deducted as such in accordance with the respective categories in the rubric. Also, DO NOT create a main method in your solution file!! This will result in your code not running properly with the driver file which will result in points being deducted from the respective categories.