COP3503 Computer Science 2
Dr. Andrew Steinberg
Spring 2025

Programming Assignment 3B
Greedy Children
Due: 3/30/2025 at 11:59pm

Objective: Students will apply concepts of greedy algorithm design.

Assignment Description: It is Halloween, and you are handing out pieces of candy to the greedy children in your neighborhood. Every so often your doorbell rings and outside there are n children. You always grab m pieces of candy ($m > n$) to hand out. Something interesting about this Halloween Season that was recommended by a group of concerned parents was to inform neighbors how greedy their child was and making sure they get the candy they deserve. This would allow the person at the door to properly hand out candy to the trick or treaters. The idea was to prevent children from being upset on the pieces of candy each received (along with egging your house). Some children are picky than others. This can be measured by a "greedy" factor g_i, $0 \leq i \leq n$. The greedy factor is displayed on the children's costume. Each piece of candy has a sweetness factor represented as s_j, $0 \leq j \leq m$. As the owner of the house in the neighborhood, it is your job to maximize the number of greedy children that will get a proper piece of candy where $g_i \leq s_j$ without throwing some sort of meltdown. Your goal is to come up with a **greedy algorithm** that finds the greedy local optimal solution along with an efficient implementation that runs in $O(mlgm)$ time.

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

1. You create your solution defined in a class called `GreedyChildren`.
2. You have been provided with 10 text files in a zip folder called "`InputTextFiles.zip`". Each text file contains a numerical value representing the greedy or sweetness factor. Files named `candy` represent the sweetness factor and files named `children` preset the greedy factor.
    a. The number after the names of the files are associated with a particular test case. Example, `candy1.txt` and `children1.txt` represents test case 1.
3. Your class should have the following attributes:
    a. An integer array that stores the Greedy factor of each child.
    b. An integer array that stores the sweetness factor of the piece candy
    c. An integer representing the number of happy children that will get the candy that satisfies their greedy factor.
    d. An integer representing the number of angry children that will get the candy that doesn't satisfy their greedy factor.

4. The `GreedyChildren` Class has an overloaded constructor that has four parameters.
    a. The first parameter represents the number of candy pieces you will hand out.
    b. The second parameter represents the number of children
    c. The third parameter represents the name of the txt file containing the greedy factor levels of each child.
    d. The fourth parameter represents the name of the txt file containing the sweetness factor of each piece of candy.
5. The `GreedyChildren` 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 `GreedyChildren` Class has a public non static method called `greedyCandy`. This method will solve the problem of the assignment. Here is where you will implement your greedy solution in maximizing the number of children getting a proper piece of candy. This method **must run** in the worst case $O(mlgm)$ where m represents the number of candy pieces. 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.
7. The `GreedyChildren` Class has a public non static method called `display`. This method will display the results `greedyCandy` computed. Simply, you will display the result of the attributes representing the happy and angry children. Each result will be displayed on a separate line. Here is an example.

```
There are 739 happy children.
There are 11 angry children.
```

A driver file (`GreedyChildrenDriver.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 `GreedyChildren.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.

COP3503 Computer Science 2
Dr. Andrew Steinberg
Spring 2025

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

1. The number of candy pieces to hand out will always be greater than the number of children that are trick or treating.
2. It is possible that some children can have the same greedy factor.
3. It is possible that some of the candy pieces can have the same sweetness factor.
4. Each greedy child can only receive **one** piece of candy.
5. Candy pieces **cannot** be broken into fractions (like in the fractional knapsack problem). You can only give a whole piece of candy.

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 $O(mlgm)$. Points will be deducted if your algorithm is not linearithmic time. Be very careful when using built in functions such as sorting algorithms!

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 that means it is stuck in an infinite loop and points will NOT be awarded in categories where testing is required. This includes test case points.

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.

What happens if my code doesn't compile or is stuck in an infinite loop? How will my code be graded? If you are code falls under not compiling or is stuck in an infinite loop, you will not be eligible to receive any points on the test cases, class implementations, constructor implementation, theoretical runtime, and greedyCandy implementation categories. That means you would receive 0 points in those respective categories. The only points you would be eligible for are code style, comments, and code submission/header. You will also receive low marks if your code does not compile.

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.