Programming Assignment #3

Programming assignments are to be done individually. You may discuss the problem and general concepts with other students, but there should be no sharing of code. You may not submit code other than that which you write yourself or is provided with the assignment. This restriction specifically prohibits downloading code from the Internet. If any code you submit is in violation of this policy, you will receive no credit for the entire assignment.

This project has several goals:

- Ensure that you understand dynamic programming
- Able to argue for optimal substructure of the problem and define recurrence for optimal solution
- Able to implement and test the algorithm for the optimal solution proposed

Part 1: Test Cases [10 points]
Add 3 to 5 *interesting* test cases. We will judge if your test cases are interesting, and you will only receive credit for those deemed so. Your test cases should follow the same format in the given test cases in `input` folder, meaning each test is in its own file and that uses a ".in" extension. Your test cases must meet the following criteria:

- The entries but be for at least 4×4 or larger
- The values you use must be *valid*

Additionally, if you notice the structure of the test cases given, there are seperate input files, one for each test. Then, there is an oracle provided in "answers.txt" – you will need to make your own "answers.txt" to highlight to me that expected output for all the test cases you created.

Part 2: Design Document [30 points]
Complete the design document template. Make sure you answer every question. You will be graded for the *quality* of your written content, which includes the professional look-and-feel of the presentation and grammar. Take advantage of the campus' writing resources if you need.

Part 3: Constrained Shortest Path [25 points]
Thanos has just acquired the Power Stone on Xander and is ready to acquire the next Infinity Stone on his quest for ultimate power. He has pinpointed the exact location of the Reality Stone to be in the Collector's collection on Knowhere and his ready to head there. However, he must first decide what route to take. He wants to get there as fast as possible but is constrained by the amount of fuel in his ship.

More formally, consider an undirected graph G where every edge e has two weights: travel time $a_e \geq 0$ and travel cost (e.g. fuel) $b_e \geq 1$. You are given two nodes s and t in G (Xander and

Knowhere). The goal is to find the $s-t-path$ with the smallest travel time subject to a limit B on the total travel cost. The travel time of a path p is defined as the sum of travel times of edges along the path ($\sum_{e \in p} a_e$), and the travel cost of a path p is defined as the sum of travel costs of edges along the path ($\sum_{e \in p} b_e$). In other words, you need to find the $s-t-path$ p that minimizes $\sum_{e \in p} a_e$ such that $\sum_{e \in p} b_e \leq B$. Note you only need to output a number denoting the travel time needed to get from s to t. (Hint: Let OPT[v,b] be the minimum travel time from node v to t with cost at most b.)

Part 4: Allocating Resources Optimally [25 points]
Thanks to his efficient route planning, Thanos has successfully obtained 5 of the Infinity Stones and his headed to Wakanda where you and the Avengers are currently protecting the Mind Stone. At this point, Thanos is too powerful and will be able to obtain the last stone in a fixed amount of time. Your only chance of stopping him is to inflict as much damage as possible in this amount of time. As the Avengers, you have a list of attacks that you can use against Thanos (Hulk Smash, Iron Man Laser Beam, etc.) that do various amount of damage given a certain amount of time.

There are n attacks and Thanos will be able to take the last stone in T time.

Each attack can do a certain amount of damage $x \geq 1$. For any given attack, the damage can vary based on the amount of time spent using it. Assume you have a set of functions $\{f_0, f_1, ..., f_{n-1}\}$, where $f_i(t)$ is the damage inflicted on Thanos if you spend t amount of time using attack i. The functions f_i are nondecreasing; spending more time on an attack will not *lower* the damage done to Thanos. You can only spend integer amounts of time on each attack. Also, only one attack can be used at a time so that you do not hit another Avenger. Your goal is to divide the time across attacks such that the damage done to Thanos is maximized.

Given T time, n attacks and functions $\{f_0, f_1, ..., f_{n-1}\}$, determine the maximum damage that can be inflicted. You algorithm should be polynomial in T and n.

Part 5: Reflection Document [10 points]
Complete the reflection document template. Make sure you answer every question. You will be graded for the *quality* of your written content, which includes the professional look-and-feel of the presentation and grammar. Take advantage of the campus' writing resources if you need.

Program Details

You are given the following:

- `DamageCalculator` class: This class has already been implemented for you. It takes in a text file representing the attack functions and parses them for use in your program. Do not modify this class to make your solution work. We will be using our own version of `DamageCalculator` during grading. Treat it as a black box to access the damage functions $\{f_0, f_1, ..., f_{n-1}\}$.
- `Driver` class: You can run this to test your code. We will be using our own Driver class during testing, so ensure that your solution is not dependent on parts of your `Driver.java`

code. To test your code, use the parameters `-1` for part 1 and `-2` for part 2 followed by the appropriate file. Ex. `java Driver -1 4.in` would test part 1 of your solution with the input file `4.in`.

- `SpaceFlight` class: This class has already been implemented for you. It represents one edge between two planets and contains the destination planet, the travel time, and the travel cost. Use the getter methods to access these attributes of the flight.
- `PlanetPathScenario` class: This class has already been implemented for you. It takes in a text file representing the planets, fuel capacity, edges between planets, and start and end planets. Do not modify this class to make your solution work. We will be using our own version of this class during grading.
- `Program3` class: You will be implementing the solution here. We have provided the function headers that we will call when testing your program. You are allowed to create additional methods or classes to solve the problem. We will assume your `Program3.java` has a no-parameter constructor, and we will initialize `Program3` using the `initialize` method.

You may not change or augment the provided code in any way. Your solution should be provided in `Program3.java`. You may include additional class files as well.

What To Submit and When:

Interesting Test Cases on 11/13. You should submit `LastName_FirstName_Tests.zip` file containing your additional tests, each in their our file (the way the test cases given to you are done), and "answers.txt" that gives the expected outcome for each test case you created. Failure to follow these submission rules will result in a loss of points.

Design Document on 11/18. Submit a single pdf file titled `LastName_FirstName_Design.pdf`. You ***will*** loose points if you do not follow this submission rule.

Code on 11/25. You should submit a single file titled `LastName_FirstName_Lab3.tar.gz` or `LastName_FirstName_Lab3.zip` that contains the just one java file with your solutions (`Program3.java`). Note that the only file used to evaluate your code is your `Program3.java`. While you may add helper methods to the `Program3` class, DO NOT include additional java class files (any additional files will be ignored). Failure to follow these submission rules will result in a loss of points.

- **DO NOT USE PACKAGE STATEMENTS.** Your code will fail to compile in our grader if you do.
- It is **highly recommended** that you do comment your code and follow good programming style. It will be very unlikely that you receive partial credit if there are not descriptive comments and the TAs cannot understand what your code is/should be doing.

Reflection on 11/25. Submit a single pdf file titled `LastName_FirstName_Reflection.pdf`. You ***will*** loose points if you do not follow this submission rule.

All deliverables must be submitted via Blackboard BEFORE 11:59 pm on their respective due dates. Absolutely no late assignments will be accepted for any reason.