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. **Do not make your code publicly available (eg github repo) as this enables others to cheat and you will be held responsible.**

Problem Description: Strategy Game

A new strategy video game has the player taking the role of the ruler of a weak state which is under attack by a substantially stronger enemy. Initially the player controls all N cities of his state. The state is long and thus the cities can be represented as N squares on a line as depicted in Figure 1. At the end of each year the player receives from each city i he controls, p_i gold coins for taxes. The opponent eventually is going to conquer the whole state; thus, the objective for the player is to maximize the amount of gold he collects before all the cities are overtaken.

At the beginning of every year the player decides to build a wall between two neighbor cities he controls. Subsequently, the enemy observes the position of the wall and chooses to attack either from east or west conquering all the cities before the wall. At the end of the year the player receives taxes from the cities still under his control and the wall is destroyed. The same process is repeated every year until the whole state is conquered.

The player should be particularly careful when choosing the location of the wall. It is known that the enemy always decides to attack from the direction that minimizes the number of gold coins the player can collect. That is, you may assume that the enemy minimizes the amount of coins the player can receive (after she/he has chosen the location for the wall) in the current and all future rounds.

Example: Consider $N = 5$ cities and $p_1 = 8, p_2 = 6, p_3 = 2, p_4 = 4, p_5 = 2$. The best option for the player is to build a wall between cities 2 and 3. The enemy attacks from west and conquers cities 1 and 2 and that allows player to collect 8 gold coins (from cities $3, 4$ and 5) at the end of the year. At the beginning of the next year the player builds a wall between the cities 3 and 4. The enemy attacks from east and conquers cities 4 and 5 allowing the player to collect 2 gold coins (from city 3) at the end of the year. There are no more neighbor cities for the player to build a wall thus the game is over with total amount of gold coins equal to 10.

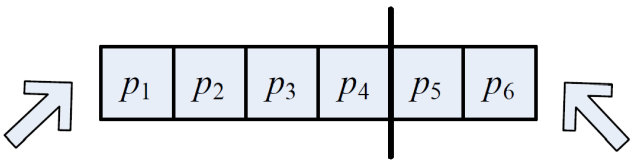

Figure 1: Strategy Game. If the enemy attacks from the west, cities 5 and 6 will remain under player's control and the profit for the current year will be $p_5 + p_6$. If the enemy attacks from the east, cities 1,2,3 and 4 will remain under player's control and the profit for the current year will be $p_1 + p_2 + p_3 + p_4$.

Instructions

Implement a dynamic programming algorithm to find the maximum number of coins to be obtained given an array of cities. The first part of this assignment will be to generate your report before you begin code implementation. This includes finding the recurrence formula and proving your algorithm's complexity. The second part of this assignment will be your actual code implementation. You will be given very little starter code in the form of `Driver3.java` and `Assignment3.java`, as well as some small test cases. Your solution should be built in the `maxCoinValue` function given to you in `Assignment3.java`. You may use `Driver3.java` as a means to test your code. Simply pass the name of a text file (just city coin values separated by spaces) as an argument to test.

Part 1: Report [30 points]
Write a short report that includes the information below. There are reasonable $O(n^3)$, $O(n^2 \log n)$, and $O(n^2)$ solutions. Faster run-time will give more points!
Note: You might find the $O(n^2)$ solution challenging to come up with.
Note: You can get full credit on the Code part of the assignment with a $O(n^2 \log n)$ solution.

(a) Explain what the dynamic programming subproblems are, and provide a recursive formula for OPT.

(b) Provide a succinct argument of correctness, and explain and justify the complexity of your algorithm.

Part 2: Code [70 points]
This part of your assignment will be graded both on correctness, and time-complexity.

- Given an `int[] cities`, it is your job to return the maximum coin value in function `maxCoinValue`. Each index i in the array has an integer value corresponding to p_i. Your value range is as follows, $1 \leq p_i \leq 30000$.

- For the first 70% percent of the test case $2 \leq N \leq 500$. (A properly implemented $O(n^3)$ solution should suffice for these test cases.)
- For the next 30% percent of the test case $2000 \leq N \leq 2500$. (Both $O(n^2)$ or $O(n^2 \log n)$ should pass these test cases.)
- Be sure to practice good programming style, as poor style may result in a deduction of points (e.g. do not name your variables foo1, foo2, int1, and int2).

Memory limit: 128MB

What To Submit

Failure to follow these instructions **will result in a deduction of points**. You should submit a single zip file titled `eid_lastname_firstname.zip` that contains:

- `Program3.java`
- Your report in pdf form, named: `eid_lastname_firstname.pdf`.

DO NOT USE PACKAGE STATEMENTS. Your code will fail to compile in our grader if you do and it won't be regraded if your program fails to compile due to package imports.
Your programming assignment is due by 11:59 PM on December 6th, 2019 (note that this is a Friday). The late submission deadline for this assignment is by 11:59 PM on December 9th 2019. You will receive a **20 point deduction** for submitting during late period.