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: Atlantis Discovered

You've gone to sleep one night after a long EE 360C study session, and you find yourself dreaming about algorithms.

In your dream, you have become Aquaman, king of the lost city of Atlantis. But you sense a disturbance in the waters. The humans have finally found Atlantis! You notice that they have already begun claiming the land as theirs, and they've been destroying all the farms, which means your people will no longer have access to food!

You have never encountered an enemy as fierce and powerful as them – they will stop at nothing to gain control of the land. You have come to realize that no matter what, your people will lose all of the land in the coming days. However, for whatever time remains, you would like to maximize the livelihood of your community.

The city is divided up into N sections. Initially, you control all N sections. The city is long and thus the sections can be represented as N squares on a line as depicted in Figure 1. In total, you collect p_i food from each section i, each day.

The humans will eventually conquer the entire city; thus, your objective is to maximize the amount of food you collect before all the sections are overtaken.

At the beginning of each day, you can build a stick wall between two neighboring sections you control. Subsequently, the humans observe the position of the wall and choose to overtake the remaining city either from east or west – conquering all the sections before the wall. At the end of the day, you count the total amount of food you collected from the side of the wall **that remained under your control**, and the wall is destroyed. The same process is repeated every day until the entire city is under human control.

You should be particularly careful when choosing the location of the wall. It's known that the humans can swim over the city, find the more bountiful half, and overtake it from the direction that minimizes the amount of food your people collect in the **long run**. That is, the humans will attack in a way that *minimizes the amount of food your people collect (after you've chosen a location for*

the wall) in the CURRENT AND ALL FUTURE DAYS.

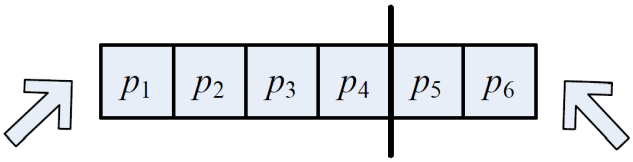

Figure 1: If the humans attack from the west, sections 5 and 6 will remain under your control and the total food for the current day will be $p_5 + p_6$. If humans attack from the east, sections 1,2,3 and 4 will remain under your control and the total food for the current day will be $p_1 + p_2 + p_3 + p_4$.

Example: Consider $N = 6$ sections and $p_1 = 3, p_2 = 2, p_3 = 4, p_4 = 2, p_5 = 6, p_6 = 8$. The best option for you is to build a wall between sections 4 and 5. The humans attack from east and conquer sections 5 and 6 and that allows you to collect 11 food (from sections $1, 2, 3$ and 4) by the end of the day. At the beginning of the next day you build a wall between the sections 2 and 3. The humans overtake from east and conquer sections 3 and 4 allowing you to collect 5 food (from section 1 and 2) by the end of the day. At the beginning of the third day, you build a wall between 1 and 2. The humans overtake from west and conquer section 1 allowing you to collect 2 food from section 2. There are no more neighbor sections for you to build a wall thus Atlantis has fallen with total amount of food collected equal to 18.

Instructions

Implement a dynamic programming algorithm to find the maximum amount of food to be obtained given an array of sections. 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 `maxFoodCount` 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 section food count separated by spaces) as an argument to test.

Part 1: Report [20 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 [80 points]
This part of your assignment will be graded both on correctness, and time-complexity.

- Given an `int[] sections`, it is your job to return the maximum food count in function `maxFoodCount`. 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
You should submit to Gradescope `Program3.java`. Please do not submit `Driver.java`. Failure to follow these instructions will result in a penalty of up to 10 points.

Your PDF report should be legibly scanned and submitted to Gradescope. Both your code and PDF report must be submitted by 11:59 PM on Wednesday, May 4th, 2022. If you are unable to complete the assignment by this time, you may submit the assignment late until Friday, May 6th, 2020 at 11:59 PM for a 20 point penalty.