EE 360C: Algorithms
University of Texas at Austin — Programming Assignment #3
Dr. Pedro Santacruz — Due: May 8, 2020, 11:59pm

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: Deforestation: An Inevitable Fate

In quarantine, you've found yourself playing many games. As a fanatic of Civilization and Catan, you were excited when a new game was released.

Here, you are the leader of a community of Apes. For many years, your tribe has lived in peace in the tropical rain forests of South East Asia. However, recently, another species known as humans have begun claiming land in the forest. Moreover, you noticed that these humans are doing something horrid – they are cutting down the trees, and your tribe will no longer have access to the tree's fruit!

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

The forest is divided up into N sections. Initially, you control all N sections. The forest 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 fruit from each section i, each day.

The humans will eventually conquer the entire forest; thus, your objective is to maximize the amount of fruit 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 forest either from east or west – conquering all the sections before the wall. At the end of the day, you count the total amount of fruit 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 forest is under human control.

You should be particularly careful when choosing the location of the wall. It is known that the humans are super smart and always decide to attack from the direction that minimizes the number of fruits apes can collect. **That is, you may assume that the humans minimize the amount**

of fruits apes can receive (after she/he has chosen the location for the wall) in the CURRENT AND ALL FUTURE ROUNDS.

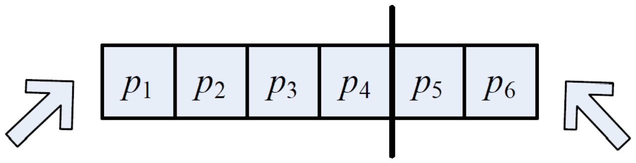

Figure 1: If the humans attack from the west, sections 5 and 6 will remain under player's control and the total fruit for the current day will be $p_5 + p_6$. If humans attack from the east, sections 1,2,3 and 4 will remain under player's control and the total fruit for the current day will be $p_1 +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

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 the player is to build a wall between sections 4 and 5. The humans attack from east and conquers sections 5 and 6 and that allows player to collect 11 fruit (from sections $1, 2, 3$ and 4) by the end of the day. At the beginning of the next day the player builds a wall between the sections 2 and 3. The humans overtake from east and conquers sections 3 and 4 allowing the player to collect 5 fruit (from section 1 and 2) by the end of the day. At the beginning of the third day, the player builds a wall between 1 and 2. The humans overtake from west and conquers section 1 allowing the player to collect 2 fruit from section 2. There are no more neighbor sections for the player to build a wall thus the game is over with total amount of fruit equal to 18.

Instructions

Implement a dynamic programming algorithm to find the maximum number of fruit 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 `maxFruitCount` 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 fruit count 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[] sections`, it is your job to return the maximum fruit count in function `maxFruitCount`. 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 a zip file titled `eid_lastname_firstname.zip` that contains all of your java files AND a **separate** pdf report `eid_lastname_firstname.pdf`. Do not put your java files in a folder before you zip them (i.e. the files should be in the root of the ZIP archive), and **do not include your pdf in the zip file**. Your solution must be submitted via Canvas by 11:59pm on the due date.
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 May 8th, 2020 (note that this is a Friday). The late submission deadline for this assignment is by 11:59 PM on May 11th 2020. You will receive a **20 point deduction** for submitting during late period.