CS4102 - ALGORITHMS - SPRING 2019

DUE WEDNESDAY, FEBRUARY 13, 2019 AT 11P VIA COLLAB

H2

Collaboration Policy: You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own independently written solution. List the names of all of your collaborators. Do not seek published solutions for any assignments. If you use any published resources when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff.

Collaborators: List in comments at the top of your file

Sources: List in comments at the top of your file

Figure 1: My Front Yard

PROBLEM 1 *Robot Leaf Mulcher*

Figure 1 shows my new front yard. It was absolutely beautiful last spring when we bought the house, and that continued into the summer, but then fall happened. The beautiful, mature trees started dropping leaves! They dropped at such a rate that with a rake and leaf blower, we could not keep up. After an entire afternoon of raking, the yard was again covered in leaves within 24 hours.

With the multiple passive-aggressive fliers left on our door by lawn companies, we've decided to create a high-tech solution to our problem. Robot vacuums have been available for years, and thankfully robot lawnmowers have recently started appearing in stores. To solve our leaf problem and cash in on this trend, I would like to create a robot leaf mulcher. However, as good product designers, we must first decide the best width of robot mulcher based on yard dimensions and the layout of trees and bushes. I would prefer to create the widest mulcher possible for my yard, allowing us to maximize battery life and minimize the amount of time the mulcher must traverse the lawn.

Since the trees and bushes in my yard appear to have been planted haphazardly many years ago, we must take their unmovable positions into account. A mockup of the lawn can be seen in Figure 2. Before we build our first prototype robot mulcher, we need to know the maximum width such that it will fit between any two trees in the yard, i.e. we need to determine distance between the closest pair of tree or bush trunks (for simplicity, disregard the leaves and branches around the base of the bushes).

Implement an algorithm which takes as input the locations of our trees and bushes as Cartesian coordinates, and returns the maximum width of the robot mulcher we can use in the yard (i.e. the closest pair of trees). Your program should have the following properties:

- Your algorithm must be written in Python (2 or 3) or Java (10).

- You must download the appropriate wrapper code from Collab based on the language you choose: `main.py` and `closest_pair.py` for Python, `Main.java` and `ClosestPair.java` for Java.
- Implement the body of the `closest_pair_distance()` or `closestPairDistance()` function, which receives as input the body of the input file as a list of lines. You must return the distance between the closest trees or bushes. An example input file is shown in Figure 3, which has answer ≈ 4.12311. Note: the coordinates may be integers or floating point numbers.
- Two additional test files are provided, `test1.txt` and `test2.txt`, both with answer ≈ 1.4142135623. You should produce additional tests, including edge cases.
- You *may* modify the `Main.java` or `main.py` files to test your algorithm, but they **will not** be used during grading.
- You must submit your `ClosestPair.java` or `closestpair.py` files on Collab. Do not zip them. Do **not** submit `Main.java`, `main.py`, or any test files.
- A few other notes:
  - Your code will be run as:
    `python main.py` or `python3 main.py` for Python,
    or `javac *.java && java Main` for Java.
  - You may upload multiple Java files if you need addional classes, but do not assign packages to the files

In the case that I discover my robot mulcher becomes the envy of the neighborhood and super popular, and this whole "professor thing" doesn't work out for me, I may mass produce them and become a robot mulcher salesman. Therefore, I would like this algorithm to be very efficient, so that I can quickly determine the appropriate size of mulcher for any size lawn, even tree farms with thousands of trees! If I put in the locations of thousands of trees and bushes, the algorithm should still run in a reasonable amount of time. For this reason, an $\Omega(n^2)$ algorithm is too slow; to be efficient enough, your algorithm **must** run in $O(n \log n)$ time.

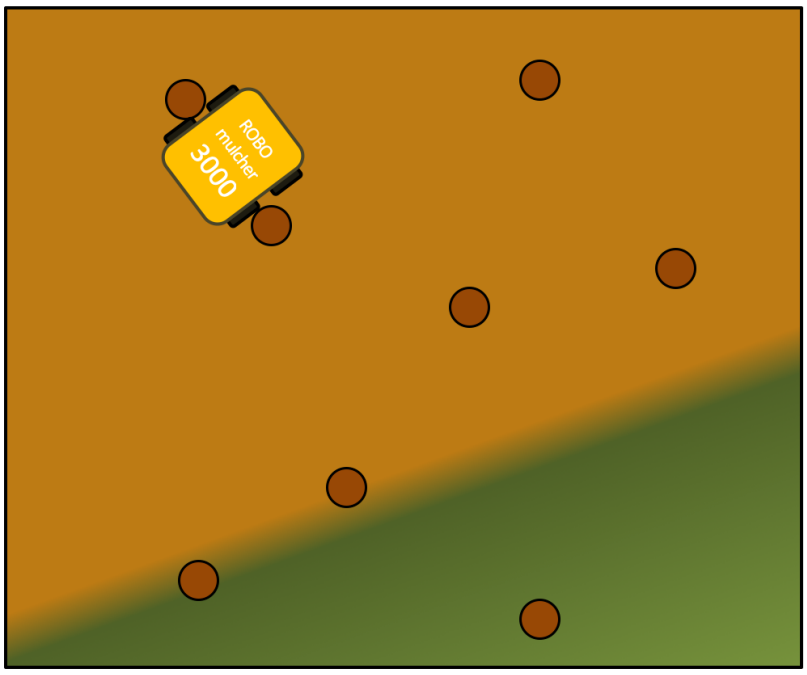

Figure 2: A 2-D grid of locations of trees and bushes. I would like to create the widest robot mulcher that will fit between any two trees or bushes, as pictured.

```
5
4 13
12 10
8 9
1 1
42 108
```

Figure 3: Example input, for which your algorithm should return ≈ 4.12311