EE360C: Algorithms
University of Texas at Austin — Programming Assignment 3
Dr. David Soloveichik — Due: December 7, 2018 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, and you may be subject to other disciplinary penalties.

This lab is due **Friday, December 7th at 11:59PM.** If you are unable to complete the lab by this time you may submit the lab late until Monday, December 10th at 11:59PM for a 20 point penalty.

The goals of this lab are:

- Familiarize you with programming in Java
- Show an application of Dynamic Programming
- Evaluate the runtime complexity of the algorithm

Problem Description

In this project, you will implement a dynamic programming solution and write a small report. We have provided Java code skeletons that you will fill in with your own solution. Please read through this document and the documentation in the starter code thoroughly before beginning.

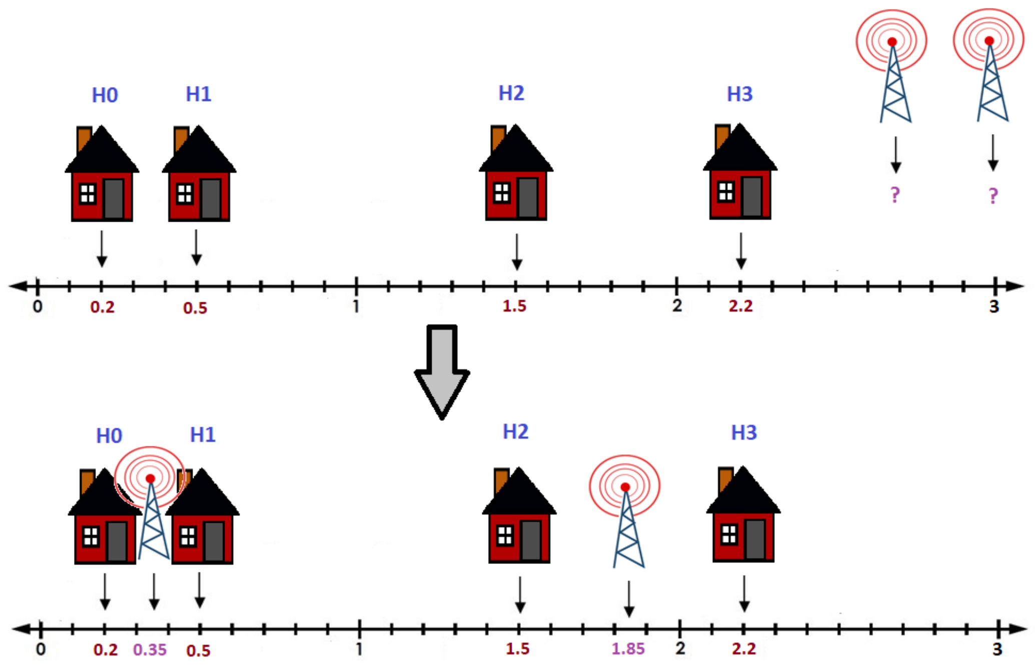
You are on the planning committee of a major telecommunication company and you are given the task of setting up k base station antennas in a town with n houses along a line. Being an electrical engineer, you know that the further an antenna is from the house it covers, the more power it wastes. You'd like to minimize the maximum distance that your antennas need to cover, ensuring that you do not want to leave any house uncovered.

More precisely, you are given a set X of n houses sorted by position along a line, i.e, House 1 is at x_1, House 2 is at x_2 and so on with $x_1 < \cdots < x_n \in \mathbb{R}$. You are also given an integer k, the number of antennas. You need to find the set of base station positions, C, of k points $c_1, ..., c_k \in \mathbb{R}$ that minimizes the antenna range. The antenna range is defined as the minimum distance r such that every x_i is at most r from some c_j. Note that all base stations are identical and cover the same distance. In other words, the antenna range is:

$$r = \max_{1 \le i \le n} \left(\min_{1 \le j \le k} |x_i - c_j| \right)$$

where $\min_{1 \le j \le k} |x_i - c_j|$ represents the distance from point x_i to the closest base station, and thus a lower bound on the antenna range required by that base station, and the max condition identifies the required antenna range—the largest distance between any house and its closest base station.

For example, consider a town called Townsville with 4 houses positioned at 0.2, 0.5, 1.5 and 2.2 with 2 base station antennas to be set up. The optimum antenna range for this given town plan is 0.35 which can be achieved by setting up the antennas at positions 0.35 and 1.85. This example can be better visualized with the illustration below.



Hint: The problem might make you scratch your head, but it isn't as difficult as it seems! Think of the possible subproblems that could be used. You could construct a $n \times k$ table indexed $r[t, j]$ that stores the optimal antenna range for the subproblems and another indexed $c[t, j]$ that stores a set of base stations for the subproblems. What can these sub-problems represent?

Part 1: Write a report [20 points]

Write a short report that includes the following information:

(a) Give the pseudocode for your implementation of a dynamic programming solution to find the above mentioned minimum antenna range r. Mention the bases cases and recurrence equation clearly and briefly explain what it represents.

(b) Justify the runtime of the algorithm in (a).

(c) Give the pseudocode for your implementation of a dynamic programming solution to find the set of base station positions, C.

(d) Justify the runtime of the algorithm in (c).

Part 2: Implementation to find Optimum Antenna Range [50 points]

Implement your dynamic programming based solution for finding the optimum antenna range which you devised in the report. You are provided several files to work with. Implement the function

that yields the optimum antenna range in `OptimalRange()` inside of `Program3.java`. You need to **only** update the data member `range` of object `town` which belongs to class `TownPlan`. **Do not** update the data member `position_base_stations` here.
Of the files we have provided, please only modify `Problem3.java`, so that your solution remains compatible with ours. However, feel free to add any additional Java files (of your own authorship) as you see fit.

Part 3: Implementation to find Set of Base Station Locations [30 points]
Implement your dynamic programming based solution for finding the base station positions which correspond to the optimum antenna range that you found. You need to implement the function that yields the base station positions in `OptimalPosBaseStations()` inside of `Program3.java`. You need to **only** update the data member `position_base_stations` of object `town` which belongs to class `TownPlan`. **Do not** update the data member `range` here. The reason for having these two functions update these data members separately is to give partial credit for the two sections.
If you are using two tables to construct the solution (as mentioned in the Hint), you are allowed to repeat the code from the previous part,i.e, from `OptimalRange()`.
Again, of the files we have provided, please only modify `Problem3.java`, so that your solution remains compatible with ours. However, feel free to add any additional Java files (of your own authorship) as you see fit.

Instructions

- Download and import the code into your favorite development environment. We will be grading in Java 1.8 on the ECE LRC machines. Therefore, we recommend you use Java 1.8 and NOT other versions of Java, as we can not guarantee that other versions of Java will be compatible with our grading scripts. **It is YOUR responsibility to ensure that your solution compiles with Java 1.8 on the ECE LRC machines.** If you have doubts, email a TA or post your question on Piazza.
- Make sure you don't add the files into a package (keep them in the default package). Some IDEs will make a new package for you automatically. If your IDE does this, make sure that you remove the package statements from your source files.
- If you do not know how to download Java or are having trouble choosing and running an IDE, email a TA, post your question on Piazza or visit the TAs during Office Hours.
- There are several `.java` files, but you only need to make modifications to `Program3.java`. **Do not modify the other files.** However, you may add additional source files in your solution if you so desire. There is a lot of starter code; carefully study the code provided for you, and ensure that you understand it before starting to code your solution. The set of provided files should compile and run successfully before you modify them.
- The main data structure for a town plan is defined and documented in `TownPlan.java`. A TownPlan object includes:
  - **n**: The number of houses
  - **k**: Number of base station antennas to setup

  - **position_houses**: An ArrayList of Float values containing the position of each house, in ascending order. The positions can be indexed in order from 0 to $n - 1$.
  - **position_base_stations**: An ArrayList of Float values to hold the positions of base stations for the optimal solution. This ArrayList (should) hold the position of each of the k base stations. This field will be empty in the `TownPlan` which is passed to your functions. The results of your algorithm should be stored in this field by calling `setPositionBaseStations(<your_solution>)`.
  - **range**: The Optimum antenna range that needs to be found. This field will be empty in the `TownPlan` which is passed to your functions. The results of your algorithm should be stored in this field by calling `setRange(<your_solution>)`.

- You must implement the methods `OptimalRange()` and `OptimalPosBaseStations()` in the file `Program3.java`. You may add methods to this file if you feel it necessary or useful. You may add additional source files if you so desire.

- `Driver.java` is the main driver program. Use command line arguments to choose between either displaying the optimal range computed through your solution of the set of positions of the base stations and to specify an input file. Use -r for range, -p for position of base stations and input file name for specified input (i.e. `java -classpath . Driver [-r] [-p] <filename>` on a linux machine). As a test, the 6-3.in input file should output the following for displaying both range and set of base station positions (corresponding command is `java -classpath . Driver 6-3.in`):

```
n = 6 k = 3
Optimum Antenna Range = 0.900000

n = 6 k = 3
Antenna 0 Position 2.500000
Antenna 1 Position 7.200000
Antenna 2 Position 11.000000
```

- You can find a few test case files under the TestCases Folder of the StarterCode zipped files given to you.

- **Input File Format:**
  The first number on the first line of file is the number of houses and the second number is the number of base station antennas to setup. The next line contains the position of the houses which **are sorted in ascending order**.
  For example, if the input file is as follows:

```
6 3
2.5 6.3 7.9 8.1 10.5 11.5
```

  Then there are 6 houses in the town and 3 base stations antenna are required to be set up.
  Think of the houses positions to be on a number line. Then,
  House 0 is at position 2.5
  House 1 is at position 6.3
  House 2 is at position 7.9

House 3 is at position 8.1
House 4 is at position 10.5
House 5 is at position 11.5

- To make things easier, we will **not** be testing for the cases where there are no houses or no base station antennas to setup.
- Make sure your program compiles on the LRC machines before you submit it.
- We will be checking programming style. A penalty of up to 10 points will be given for poor programming practices (e.g. do not name your variables foo1, foo2, int1, and int2).
- Before you submit, be sure to turn your report into a PDF and name your PDF file `eid_lastname_firstname.pdf`.

What To Submit

You should submit a single zip file titled `eid_lastname_firstname.zip` that contains all of your java files and pdf report `eid_lastname_firstname.pdf`. Do not put these files in a folder before you zip them (i.e. the files should be in the root of the ZIP archive). Your solution must be submitted via Canvas BY 11:59 pm on the due date.