B403 Project Instructions (Spring 2017)

Contact: Erfan Sadeqi Azer (esadeqia@indiana.edu)

1 Introduction

For this project, you need to solve two programming problems on an online server called Hacherrank. So, the first step is to open an account. If you do not have one yet, you can easily open one in https://www.hackerrank.com/signup.

You might try to solve some problems in the “Warmup” domain to get familiar with the Hacherrank server. Starting with Warmup problems is highly recommended, but they are not part of this project. The problems that you are going to solve are:

1. **Rust and Murderer**: [https://www.hackerrank.com/challenges/rust-murderer](https://www.hackerrank.com/challenges/rust-murderer)
   - 7 points in total.
   - Soft deadline: **Feb 28th**. Before this date we (contact: Erfan Sadeqi Azer) can help you with any unexpected issue you have with your code, and check if you have uploaded all the required items (report, codes,...).
   - Hard deadline: **March 10th**. No submission will be accepted after this date.

2. **TBA**: 8 points in total; deadline: Mid April

   There are around 30 programming languages available for this problem, including Python, C, and Java. The time limit for each language is different. Thus, choose the one that you are most familiar with.

2 Algorithm

After you read and understood the problem, you may first start with designing an algorithm, analyze time and space complexity, and then try to improve the time complexity. There is a “Discussions” tab in the problem statement page. Hackerrank users use it to discuss the problem as well as potential algorithms with each other and with the designer of the problem. You are very welcome to read those and ask or contribute to those discussions within Hackerrank’s polices. Also, it is possible to find related discussions in the web. You
are allowed to look for and use anything on the theory part of the problem. However, you are not allowed to search for, read or use whole or part of any code that is written by anyone other than you (in particular, those in the “Leaderboard”), otherwise you will be given “F” for the entire course. Note that we will read and compare your codes.

3 Implementation

After you understood and analyzed your optimized algorithm, you can go ahead and implement it with your favorite language. After you checked the sample inputs and possibly other custom inputs, it is possible to run your code on the server. You can do this by “Run Code” button at the bottom of the problem statement page. Then, you might submit your code and see if the online judge accepts it. If it is rejected, feel free to debug your code and submit again. When your code is accepted, take a snapshot of the page, and keep it for your report.

4 Report

You need to write a report for each of the two problems, explaining your algorithm in your own words (this part should be in English. So, if you are giving a pseudo-code, it should not use any language-specific notations) with time and space analyses. Then explain your implementation. Mention a few core functions, and if you needed to do any implementation tricks to make it work. Include the screenshot of the accepted run. Specify the time that this run was submitted and what is the name of your account. Submit your report in PDF along with all your code files through the Canvas system. The assignment name is “Project 1.”

5 Some more points

- This project’s implementation must be done individually. Each student should write their own code, character by character. Any form of copying from other students’ codes or from an online resource, whether related or unrelated to this problem will be considered as plagiarism, and will not be tolerated (will result an “F” for the entire course). However, Discussing the algorithm and getting theoretical ideas from classmates or any other resource is allowed and encouraged.

- It is hard to estimate how long it will take to successfully finish the project. So, it is very important to start as soon as possible to have enough time to ask questions and fix potential bugs. Partial credit will not be given to the codes that do not work for all test inputs.
• Keep in mind that even though your code has been tested with an automatic judge, it will be read by your AI, too. So try to write a readable code and add commentary. Make explicit references in your report with names of functions or variables.