CS3230: Design and Analysis of Algorithms
Semester 2, 2022-23, School of Computing, NUS

Programming Assignment 1

Deadline: 17rd February, 2023 (Friday), 11:59 pm

Instructions

- First, register a (free) account at `https://www.cs3233.com/register/`, the steps:
  1. **Username**: your username must be in this format 'TXY_A0123456C' (exactly 13 characters for everyone) where XY is your tutorial group number ('X' is digit '0' if you are in [T01..T09], do not omit this) and replace '123456C' with your student ID (your student ID is the one that starts with "A0".). Please be as careful as possible when inputting your username as there is no 'rename username' feature from the client-side yet :(. Do NOT use your real name.
  2. **Password** and **Password confirmation**: Just use random strong password generator *that you have never used elsewhere* (just to cover all bases in case there is a (successful?) hacking attempt to the server) and *save the password in your password manager* (so that you don't forget, as you there is no reset password feature and you obviously cannot re-create the same username (see the previous step...)).
  3. **Kattis account/Codeforces account/profile picture**: You can leave Kattis/Codeforces account fields to be a dummy value and upload dummy profile picture to bypass the form validation (all these three are not important for CS3230).
  4. **Registration code**: Finally, use this code "CS3233ISFUN" (in ideal world, I (Steven) could have bought another domain 'cs3230.com' and change the code, but to save cost, let's just share this online judge platform across both modules that I manage).
- Visit `https://www.cs3233.com/contest/20` (please ignore CS3233 related contests, they are not for CS3230). There are 2 tasks (A and B, but B is split into two subtasks B1-weaker and B2-the real one). The system will allow submissions from 3rd February, 2023 (Friday), at 09:00am onwards. Those who have completed Written Assignment 1 (not yet due until 11:59pm of the same day) can start a bit earlier.
- Solve the given tasks using any of these 3 programming languages (sorry we do not support any other programming language): C++ (preferred, can solve all tasks comfortably within the time limit if you use the correct algorithm), Python (can solve A and B1 only if you use the best algorithm, unfortunately the best algorithm for B2 in Python is slower than the second best algorithm in C++ so we have to leave this option out), or Java (in-between, in terms of runtime speed).
- Heavy one-strike-all-CA-marks-gone penalty is applicable to anyone who is proven beyond reasonable doubt submitting code that is 'identical' (with very high percentage of similarities) with another submission(s). Please deviate as far as possible from any AI-generated code as they will be definitely be used as one of the sources for comparison.
- What will be graded is not just your submissions at `https://www.cs3233.com`, but your manual reflection-based answers too (see below).
- After you are done submitting your code at `https://www.cs3233.com`, also submit the soft copy of your reflection-based answers at Canvas → Assignment → Programming Assignment 1.

- Write legibly. If we cannot read what you write, we cannot give points. In case you CANNOT write legibly, please type out your answers.
- When you submit your answer, please try to make it concise. However, it should contain all the relevant details. The page limit is still 6. But I believe that there is no need to write long answers this time, as you have written two (or three) source code for task A, B1/B2 (could be one implementation or two sub-versions).
- Before submitting your answers, please make sure you rename your submission file in the following way: <Student ID>.pdf. (Note, your student ID is that starts with "A0".) — but I feel Canvas actually auto-renames your submission file into its standard internal naming convention.
- If you need any clarification on any of the questions, we strongly encourage you to post in Class Discord server (instead of emailing us) so that everybody can see our responses. You may also approach Steven, Diptarka or your tutor.

Task 1 - Mastering Master Theorem [20 marks]:

12 marks Solve task A at `https://www.cs3233.com/contest/20/A/` before deadline.
No further action needed if your solution gets Accepted
(you get auto 12 marks if your submission passes plagiarism checker).

However, if your solution remains not Accepted until deadline, copy paste your latest not Accepted code into your report for your TA to scrutinize one last time and you will be given [1/4/7] partial marks depending on how far your not Accepted code is from correct.

4 marks a). If you get a few Wrong Answer(s)/other non Accepted solution(s) and eventually gets Accepted before deadline by yourself, explain the (Master theorem) cases that you miss earlier and how do you fix them in a short paragraph.

b). However, if you already get Accepted in your first try, answer the following sub-question instead: Give as many creative suggestions to Steven to improve exactly this task for S2 AY2023/2024, e.g., how to make the task covers more variations of Divide and Conquer recurrences, etc.

4 marks What are your learning points about Master theorem after going through this version of task A?
What are Master theorem pros/potential-automation?
(PS: do you double check your Written Assignment 1 answers using this code?)
What are Master theorem cons/limitations?

Task 2 - Linear-time Sort [20 marks]:

12 marks Solve task B1 at `https://www.cs3233.com/contest/20/B1/` before deadline.
No further action needed if your solution gets Accepted (after plagiarism check).

However, if your solution remains not Accepted until deadline, copy paste your latest code in your submission for your TA to scrutinize one last time and you will be given [1/4/7] partial marks depending on how far your not Accepted code is from correct.

4 marks This task B1 is generally about beating the lower bound of $\Omega(n \log n)$ of comparison-based sorting. We have tested (as best as we can) that no form of $O(TC \times n \log n)$ comparison-based sorting can successfully pass the time limit on our secret test cases. Can you give an explanation on why is that so (or surprise us by writing a code that runs in $O(TC \times n \log n)$ but gets Accepted by the judging server)?

What are the simplifying constraints that you can take advantage of to design a sorting algorithm that is not comparing integers?

Explain your solution in details, show/prove that it is correct, and analyze its time complexity.

2 marks Solve task B2 at `https://www.cs3233.com/contest/20/B2/` before deadline.
No further action needed if your solution gets Accepted (after plagiarism check).
Note that it is very unlikely that you can pass B2 but not B1.
There is no partial mark for this component, i.e., strict [0 vs 2 marks].
Hence there is no need to copy paste your latest non Accepted code for this part.
Feel free to skip this 2 marks if it does not worth your time/effort.

2 marks This part will be graded regardless of the status of your B2 submission after deadline.
First, explain what is the main difference between task B1 and B2!

a). If you manage to get Accepted at task B2,
Explain your solution in details, show/prove that it is correct, and analyze its time complexity.
If your B2 solution is identical with B1, you can just say so; Otherwise, explain the new solution.

b). If you does not manage to get Accepted at task B2, try to give a list of possibilities of the potential issues (that you have not tried, otherwise you might have fixed the issue yourself), e.g., "Sorry, I only know how to code in Python, so I have to skip B2...".