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

Programming Assignment 2

Deadline: 03rd March, 2023 (Friday), 11:59 pm

Instructions

- Use the same account at `https://www.cs3233.com`
  (but for a few rare cases who used your own name for PA1 — just re-create a fresh user account).
- Go to `https://www.cs3233.com/contests/` and then click "Register" for "CS3230 PA2 (17 Feb-03 Mar 2023)" session (again, please ignore CS3233 related contests, they are not for CS3230). There are 2 tasks (A and B, but this time A is split into two subtasks A1-weaker and A2-the real one). The system will allow submissions from 17 February, 2023 (Friday), at 09:00am onwards but this file will only be uploaded one week later as it contains reflection questions that are a bit too-spoilery. Those who have completed Programming 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 (likely can solve A1 only), or Java (in-between, in terms of runtime speed, and can solve all tasks, albeit with smaller margin of error).
- **NEW for PA2+PA3: For every submissions from Thu, 23 Feb 2023, 12nn onwards, please put "// TXY_A0123456C" (C++ or Java) or "# TXY_A0123456C" (Python) as the FIRST LINE of any submitted code to `https://www.cs3233.com`, of course after replacing XY with your tutorial group number and A0123456C with your own student number, to aid with the pairwise plagiarism checker later. For those who have cleared PA2, you do not need to resubmit your code just for this. But any new submissions from here onwards should follow this new rule.**
- 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 2.
- You can assume that the web server that hosts `https://www.cs3233.com` can perform up to 10^8 basic operations in 1s — when it is not heavily loaded by multiple submissions at the same time.
- 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 A1/A2 (could be one implementation or two sub-versions) and B.

- It is confirmed that you can name your pdf as any file name, but to make it easier for you, just stick to <Student ID>.pdf notation.
- 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 Matrix Multiplication [23 marks]:

6 marks Solve task A1 at `https://www.cs3233.com/contest/21/A1/` before deadline.
No further action needed if your solution gets Accepted
(you get auto 6 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 [3] partial marks if it actually solves all test cases, just that it is too slow.

10 marks Solve task A2 at `https://www.cs3233.com/contest/21/A2/` before deadline.
No further action needed if your solution gets Accepted (after plagiarism check).
Note that it is very unlikely that you can pass A2 but not A1.
There is no partial mark for this component, i.e., strict [0 vs 10 marks].
Hence there is no need to copy paste your latest non Accepted code for this part.

7 marks Answer the following reflection questions about A1+A2 (just a short answer each will be enough), each question worth 1 mark.

(a) Do you need to implement $\Theta(n^{2.8073})$ Strassen's algorithm to solve A1? Explain your answer!
(b) Do you need to implement $\Theta(n^{2.8073})$ Strassen's algorithm to solve A2? Explain your answer!
(c) Do you need to implement $\Theta(n^{2.37188})$ by Duan, Wu, Zhou to solve A2? Explain your answer!

From here onwards, we will discuss Freivald's algorithm that is mentioned very briefly during Lecture 06, with assumption that majority (99%?) of students will implement this algorithm as outlined in `https://en.wikipedia.org/wiki/Freivalds%27_algorithm` (self-study this algorithm).

Note that if you **really solve** A2 with a different algorithm (or alternative variations of Freivald's algorithm that is different from that Wikipedia article), you can skip answering part [defg] and please elaborate your alternative answer instead for the same 4 marks.

(a) Is Freivald's algorithm of type Monte Carlo or Las Vegas algorithm?
(b) Assuming that you have ensured that "inner matrix dimensions agree", then we have:
1). Test cases where $A \cdot B = C$ (you should output "AC")
2). Test cases where $A \cdot B \neq C$ (you should output "TLE")
Which test cases that Freivald's algorithm runs a bit faster vs its worst-case time complexity?
Which test cases that Freivald's algorithm is always correct vs may be wrong?
(c) The standard Freivald's algorithm has an error rate of $1/2^k$.
What is your chosen k so that you can solve A2?
(d) How many submission(s), if you setup Freivald's algorithm as answered earlier,
until you can solve A2 <u>for the first time</u>, in <u>theoretical</u> expectation?
To answer this question, please do not factor in potential implementation bugs...

Task 2 - Linear-time Select [17 marks]:

10 marks Solve task B at `https://www.cs3233.com/contest/21/B/` 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 [7] partial marks if it actually solves all test cases, just that it is too slow.

7 marks Answer the following reflection questions about B (just a short answer each will be enough), each question worth 1 mark.

(a) Are you able to use C++ std::sort/Java Collections.sort/Python list.sort() (all comparison-based sorts) that runs in $O(n \log n)$ per sort to solve B? Explain your answer!

(b) Can you explain the meaning of the new way unsorted sequence **U** is generated this time?

(c) Are you able to use your PA1-B2/radix sort (or PA1-B1/counting sort) code runs that is supposed to run in linear time to solve B? Explain your answer!

(d) Do you use worst-case linear time select as discussed at the very end of Lecture 06 or the expected linear time QuickSelect as discussed on Week 07 tutorial?

efg-1 If you use worst-case linear time select, elaborate a bit of your implementation details.
1. Does it actually pass the time limit of B?
2. Do you divide n elements into groups of 5 as discussed or other group sizes (e.g., group of 3 or group of 7, etc)
3. How do you find the median in a group of 5?

efg-2 If you use expected linear time QuickSelect, elaborate a bit of your implementation details.
1. Does it actually pass the time limit of B?
2. Is it Monte Carlo or Las Vegas algorithm?
3. So what should you do if you get Time Limit Exceeded?