Assignment 1: Asymptotics and Greedy Algorithms (10%)

Winter 2024 – CPSC 413

Due at 23:59, Feb. 9

Assignment policy:

- This is an **individual** assignment, so the work you hand in must be your own. Any external sources used must be properly cited (see below).
- Submit your answers to the written questions as a **single PDF** to the correct location on Gradescope. You will need to submit your solution to the coding question **separately** to the autograder on Gradescope.
- See D2L for the assignment late policy. You are responsible for tracking how many flex days you have used. Remember that once these flex days have been used, late assignments will not be accepted.
- All submissions must be clearly legible. Handwritten assignments will be accepted, but marks may be deducted if the TAs cannot read your writing. It is strongly recommended that you use LaTeX to typeset your work.
- Some tips to avoid plagiarism in your assignments:
  1. Discuss and share ideas with other students as much as you like, but make sure that when you write your assignment, it is your own work. A good rule of thumb is to wait 20 minutes after talking with somebody before writing it down. If you exchange written material with another student (including email, discord chat, text messages, etc.), take notes while discussing with a fellow student, or copy from another person's screen, then this work is **not** yours.
  2. Cite all sources for material you hand in that is not your original work. Include these citations directly into the submitted PDF at the end of the question you used the source for, or as a comment in your code. For example, if you reference a website when writing a proof, you should include a link to the website at the end of the question, along with a description as to how it was used. Use the complete URL so that the marker can check the source.
  3. Posting assignment questions to online forums (Chegg, Stack Overflow, etc.) is strictly prohibited. This is a copyright violation, and all cases will be immediately reported to the department.
  4. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. However, you may still get a low grade if you submit work that is not primarily your own. You will not be given credit for text or code that is clearly generated by another source.
  5. We will be looking for plagiarism in all submissions, possibly using automated software designed for the task.
  6. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help than it is to plagiarize.

Written questions:

1. (2 marks) Prove that given any constants $a > 1$ and $b > 0$, $n^b \in o(a^n)$. This shows that **any** exponential function grows asymptotically faster than **any** polynomial function.

2. (1 mark) Computer scientists tend to get lazy and omit the base when writing a logarithm. Your job in this question is to justify this from an asymptotic perspective. Prove that for any $a > 1$ and $b > 1$, $\log_a(n) \in \Theta(\log_b(n))$, showing that the base of the logarithm does not affect the asymptotic complexity.

3. (1 mark) Let $f(n) = 2^{2n+3}$ and $g(n) = 2^n$. Notice that $2n + 3 \in \Theta(n)$. Is is true that $f(n) \in \Theta(g(n))$? Prove your answer is correct.

4. (4 marks) Along the long, straight road from Froogletown to Squoogleville, there are n houses and n internet hubs. Each hub is capable of providing an internet connection to exactly one house, and doing so requires installing a cable between the hub and the house it connects. No two internet connections can share the same cable. Your job is to assign houses to hubs so that the total length of cable required is minimized. The input and output of the problem are as follows.

Input: An array A of n real numbers indicating the location of the houses along the road, and an array B of n real numbers indicating the location of the hubs along the road. Both arrays are sorted by increasing value. **Output:** The total length m of cable required to connect all the houses to hubs, such that m is minimum.

   Give an efficient algorithm for this problem. Prove your algorithm is correct and give the runtime.

Coding question:

5. (2 marks) You have just registered for a semester at Chaos University and are planning out your courses. However, the classes at this university do not follow an hourly timetable. Instead, classes are scheduled for arbitrary lengths, and begin and end at arbitrary times.

   Despite this, you are a keen student and want to take as many classes next semester as possible (and you don't care what they are). Because you work on weekends, you will need to fit all your classes between the time you can first get to campus at the start of the week and the time you have to leave for work at the end of the week. Finally, you must allow for **at least 10 units of time** between classes, so that you can run across campus to the next classroom.

   You've obtained the semester's weekly table of class start and end times, which are represented as integers between 0 and 99,999[1]. Write a Python 3 function that, given this file of start and end times, prints the maximum number of classes that you can take between time `start_time` (inclusive) and time `end_time` (exclusive).

   Your function should be contained in the Python 3 starter class provided on D2L. This class has a constructor with an argument named `classlist`, which is an array of tuples such as `[(11,25), (32,73)]`, where each tuple contains the pair of start (inclusive) and end times (exclusive) for a university class. All values, including interval start/end times and the function inputs `start_time` and `end_time`, will be integers between 0 and $99,999$.

   Upload your `scheduler.py` to the correct location on Gradescope. Make sure your code is clean and well-commented.

   Autograder notes:

   Your code will be graded by an autograder on Gradescope. For your code to work with the autograder, you will need to use the template class `scheduler.py` provided on D2L. You should write your code to fill in the `schedule()` function **without changing the arguments**, and you must return (not print) an integer indicating the maximum number of lectures you could take. You do not need to change the rest of the class. Do not change the filename, and do not upload any files other than your `scheduler.py`.

   You can test your code with the autograder as many times as you like before submitting. It is also recommended that you add print statements for debugging during your own initial development. There is a set of test cases included in `test.py`, provided on D2L, that you can use to test your scheduler code locally if you prefer.

   Note that marks will be allocated to your code structure and commenting, so pay attention to this too.

---

author: jcleahy@ucalgary.ca

[1] This is Chaos University, so nobody knows what units they use