Deadline: December 4 at 8pm eastern

CS1802 Recitation 9

Fall 2020

December 1-4, 2020

Instructions: The problems in this recitation are based on the course material covered in the CS1800 lecture videos and are meant to prepare you for upcoming homework assignments. You earn full credit for a recitation by using your time well and demonstrating effort on the assignment. Submit your solution on Gradescope by uploading images of hand-written work, or uploading a PDF.

Logistics for Fall 2020: The recitation assignments are designed to be completed within the official 65-minute time. However, we know that schedules are harder to work with this semester, and so the deadline for recitations will officially be on Fridays at 8pm eastern. We recommend submitting your work in real-time at the end of your section, but it's OK if your preference is to submit later as long as you meet that last deadline.

- In-person: If you're able to join in-person, please come to the classroom where instructors will be there to help. Work on the assignment, ask us questions, and submit whatever you have when time is up.
- Synchronous, remote: If you're not able to join in-person but you can remotely join at the designated time, please join the recitation remotely. Work on the assignment, post any questions in the meeting chat, and submit whatever you have when the time is up.
- Asynchronous, remote: If you're both remote and not able to join in real-time, we suggest you register for the asynchronous online section. Dedicate 65 minutes to work on the assignment, and submit your solution by the Friday deadline.

Question 1.

List the following functions in increasing order of growth (slowest-growing to fastest-growing):

1. $f(n) = n^3 + 14 + n$
2. $g(n) = 12n + 15n + 25n + 34$
3. $h(n) = 4 \cdot lgn + 2^8$
4. $i(n) = 3^n + n^3$
5. $j(n) = 1 + 4 + 9 + 16 + 25 + 36 + 49 + 64$

Question 2.

Use mathematical induction to show that $n! > 3^n > 2^n > n^2 > n \log_2(n)$ for $n \geq 7$.

Question 3.

Suppose the run-time of an algorithm is given by the recurrence

- $T(n) = T(n/2) + T(n/2) + c$
- Base case: T(1)=1

c is a constant. Solve the recurrence relation to come up with a closed-form equivalent run-time. Express the upper-bound of the run-time in big-O notation.

Question 4.

You are given an unsorted array of n positive integers. The numbers in the array are in the range 1, 2, ..., $n + 1$ and each value appears in the array exactly once, except for one that doesn't show up at all. Your goal is to find the missing number.

Initial Algorithm: Create a second array with all $1, 2, ..., n + 1$ integers in it. Iterate over this new array; for every element in the new array, look for it in the first array by inspecting every number. If the new array's element isn't in the first array, then that's the missing one.

(a) What is the run-time of the above algorithm as described?

(b) Describe a more efficient way to solve the same problem.

(c) What is the run-time of your improved algorithm?

Question 5.

Show that $f(n) = 2^n$ is $O(3^n)$, but $f(n) = 3^n$ is not $O(2^n)$.