Math 126 Problem 5: Hashing Around the Clock

Spring 2027

Submission Instructions

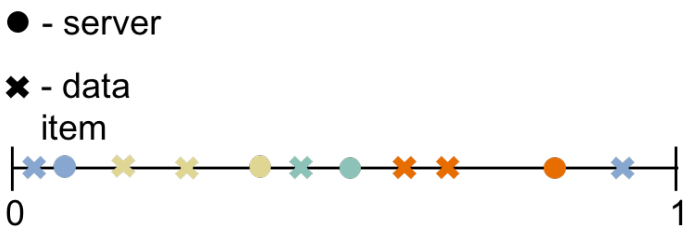
Please upload your work to Gradescope by **11:59pm on Monday, February 8, 2027 (Pacific time).**

- You are encouraged to discuss ideas and work with your classmates. However, you *must acknowledge* your collaborators at the top of each solution on which you collaborated with others and you *must write* your solutions independently.
- Your solutions to theory questions must be written legibly, or typeset in LaTeX or markdown.
- I recommend that you write your solutions in a Python notebook using Google Colab.
- You should submit your solutions as a *single PDF* via the assignment on Gradescope.

Grading: The point of the problem set is for *you* to learn. To this end, I hope to disincentivize the use of LLMs by *not* grading your work for correctness. Instead, you will grade your own work by comparing it to my solutions. This self-grade is due the Friday *after* the problem set is due, also on Gradescope.
Original notebook: Google Colab.

Hashing Around the Clock

In modern systems, hashing is often used to distribute data items or computational tasks to a collection of servers. What happens when a server is added or removed from a system? Most hash functions, including those discussed in class, are tailored to the number of servers, n, and would change completely if n changes. This would require rehashing and moving all of our m data items, an expensive operation.



Here we consider an approach to avoid this problem. Assume we have access to a completely random hash function that maps any value x to a real value $h(x) \in [0, 1]$. Use the hash function to map *both* data items and servers randomly to $[0, 1]$. Each data item is stored on the first server to its right on the number line (with wrap around – i.e. a job hashed below 1 but above all servers is assigned to the first server after 0). When a new server is added to the system, we hash it to $[0, 1]$ and move data items accordingly.

Part A

Suppose we have n servers initially. When a new server is added to the system, what is the expected number of data items that need to be relocated?

Part B

Show that, with probability $> 9/10$, no server "owns" more than an

$$\frac{\log_e(10n)}{n}$$

fraction of the interval $[0, 1]$. This can be proven *without* a concentration bound.

Hint 1:

$$\left(1 - \frac{1}{z}\right)^z \leq \frac{1}{e}$$