Mod 10 Prompt - Noboru Hayashi

- I think large companies like Google, Amazon are having challenges that are to manage employee or customer data and get some insights from it by accessing the database, while having optimized access performance.
- Companies might access databases by searching with some specific filter such as searching employees by IDs, orgs or job positions. Sometimes companies need to browse a big portion of continuous data in order like web accessing history.
- Social Security Numbers are unique for each employee, so once using a hash to map one to one relation between SSN and employee record, the optimized access speed is O(1) with an array structure allowing random access. However SSNs are having specific formats related to specific geographic info of the person, so for some companies, employees' SSNs are not uniformly distributed. That would likely cause some data collision while storing employee records by hashing SSNs. I think a better alternative would be using a unique and not null field as a primary key such as employee IDs.
- Hashing is not suitable for applications needing sorted data, however if the company's directory needs a sorted file only periodically, it's still possible to utilize hashing strategy if the application mainly randomly accesses it. To get a sorted file, reorganization of the hash mapping is needed at the same time, which has an performance impact on the application.
- I would say the impact is bigger, since we need to implement much more complex sorting algorithms that sorts by multiple fields.