Discussion Prompt Questions
Module 12
Brian Loughran
Johns Hopkins Data Structures

1. What are some of the largest employers with which you are familiar? What do you imagine are some of the greatest challenges facing them in terms of information management?
2. What are some of the many ways that company may need to access data?
3. What are the pros and cons of using a social security number as a primary key? What are some good alternatives?
4. How big an impact, on your design considerations in general, is it to need a sorted file only periodically (for a directory, for example)?
5. How big an impact, on your design considerations in general, is it to sort by different fields of the data periodically (a directory is likely ordered by last name, first name, for example)?

ANSWERS:

1. Some of the largest employers I can think of include Amazon, Microsoft, Google, Apple, Walmart, Alibaba, etc. These employers can have hundreds of thousands, or even millions of employees, so being able to manage all the data that those employees are responsible for can be quite a challenge. I actually work for a large company, Pratt & Whitney, where information management is a tremendous struggle. With thousands of people all working on the cutting edge of aerospace technology, being able to agree on a company-wide methodology to complete a task, or find a model which correlates well to certification reports on engines can be a daunting task. Especially working in an evolving field, where the requirements, methodologies, and industry standards are always changing, it can be tough for a company to manage the information, and as an employee to be able to track and understand all that data.
2. For a company with data such as inventory, different ways they may need to access the data would be by name (if you are doing a search criteria), by price (if you want to find an item at a certain price range), or inventory (if you are doing some restocking or supply chain management). A bank, perhaps, may need to access customer data by customer name, phone number, routing number, social security number, or address.
3. Using a social security number as a primary key gives you a numerical key (great for sorting) which can represent a lot of data for that individual. If a bank, for instance, were to use a social security number, under that key you could store a lot of information, such as name, date of birth, address, occupation, balance, and many more. It has the added benefit that only one person has that social security number (in contrast to sorting by name, where you can have any number of John Smiths). Some disadvantages to using the social security number are in the realm of security. If the data is hacked, then the hacker will have instant, easy access to the victim's social security number. Also, anyone looking up the data for a person would either need the social security number for a primary key search, or would have potentially easy access to that data. That would also not apply to anyone outside of the U.S. So while it is a convenient to use the social security number, it is not very secure.
4. If the sort only has to happen periodically, then we can choose a sort algorithm that does not have pristine sorting capability. If we are sorting every month, there is potential to just sort on

an off day overnight. As long as the dataset is not too big, any reasonable algorithm should be able to sort by the morning. You may also be inclined to keep multiple lists. For example, if a directory has name, phone number and address, you may be inclined to have sorted lists for each, that way you can do efficient searching on any field. For many fields and inserting on the fly, it may not be practical to keep track of these lists, but it would likely be more practical if you only have to sort periodically.

5. Sorting by different fields may multiply the time it takes to sort by the number of fields. Thus, if your sort algorithm takes O(nlgn) to sort one field, it will take O(mnlgn) to sort m fields. Sorting periodically has the disadvantage of not being able to efficiently sort the entry directly after it has been inputted. So sorting periodically would not be a good implementation if you need to access data immediately. But sorting periodically could be useful if you want to put in a large amount of entries, and do not want to wait each entry for the data to be sorted. So depending on the application and its need, periodic sorting can be helpful in some cases