Pros and cons of hash-based structures

Selecting transcript lines in this section will navigate to timestamp in the video

- So when should you use a hash-based data structure? Data and hash maps is stored in such a way that searching is much faster than many other data structures. However, hash maps do take up more space. Insertion and deletion is also quick. There's a little bit of processing involved for retrieval and insertion, but it's the same amount of processing every time, no matter how big the hash table is. This means that lookup, insertion and deletion all take O of one or constant time, because its runtime is consistent across any input. The only reason this might differ is if you have collisions and must handle them with separate chaining, which create linked lists with additional values. If that's the case, then these operations may take up more time, and up to O of n if all the values are stored in one bucket. Now, if you have a custom object as a key, you might need to provide your own hash and equality functionality, as we talked about before. But in a situation where you have a lot of key-value pairs or volatile data, these beat arrays hands down. Depending on how you want to handle your data, one of the other structures may work out better. There's no real sorting with hash-based data structures.