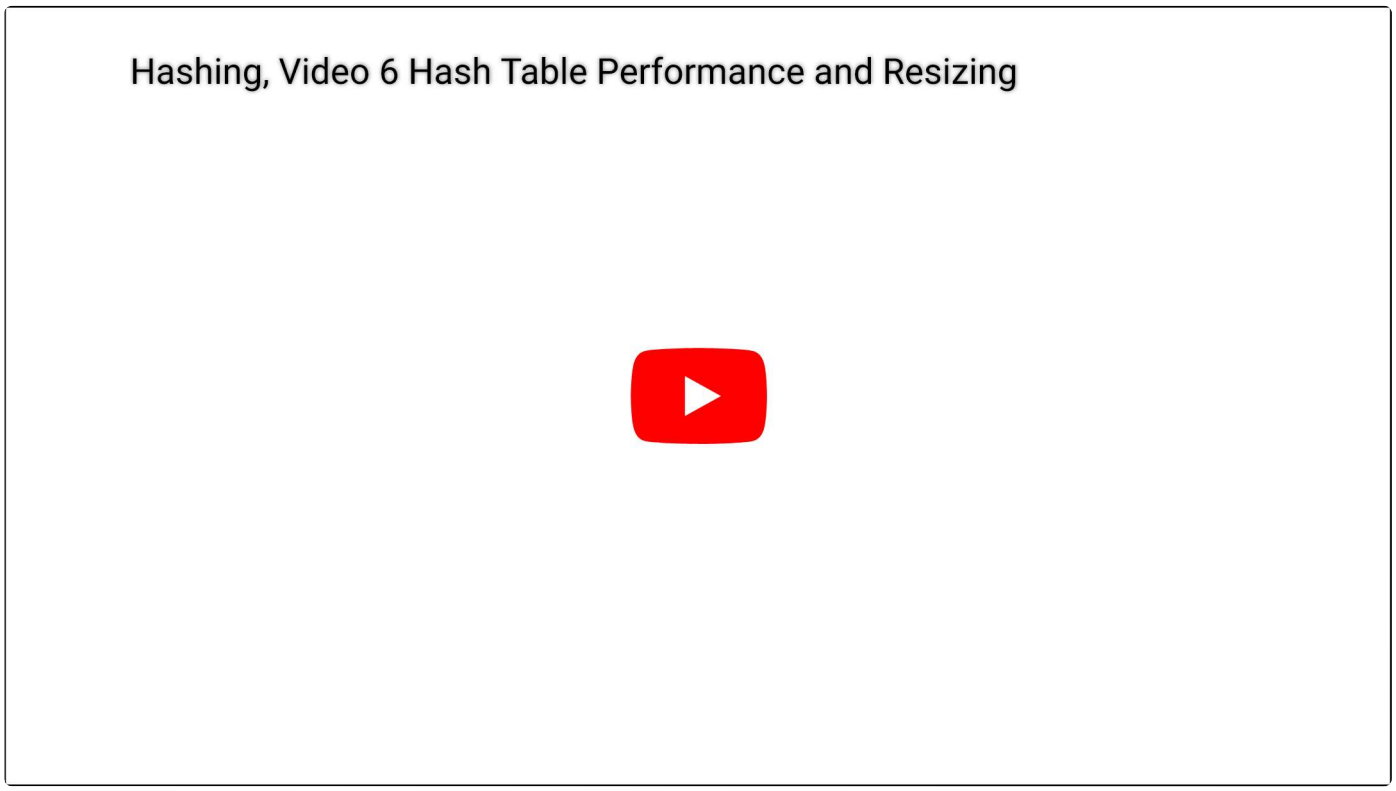
19.5 Resizing & Hash Table Performance

No matter how good our `hashCode()` method is, if the underlying array of our hash table is small and we add a lot of keys to it, then we will start getting more and more collisions. Because of this, a hash table should expand its underlying array once it starts to fill up (much like how an `ArrayList` expands once it fills up).



Professor Hug's Lecture on why we have resizing and how it works!

To keep track of how full our hash table is, we define the term **load factor**, which is the ratio of the number of elements inserted over the total physical length of the array.

$$\text{load factor} = \frac{\texttt{size()}}{\texttt{array.length}}$$

A very important equation

For our hash table, we will define the maximum load factor that we will allow. **If adding another key-value pair would cause the load factor to exceed the specified maximum load factor, then the hash table should resize.** This is usually done by doubling the underlying array length. Java's default maximum load factor is 0.75 which provides a good balance between a reasonably-sized array and reducing collisions.

Note that if we are trying to add a key-value pair and the key already exists in the hash map, the corresponding value should be updated but no resizing should occur.

As an example, let's consider what happens if our hash table has an array length of 10 and currently contains 7 elements. Each of these 7 elements are hashed modulo 10 because we want to get an index within the range of 0 through 9. The current load factor is 7/10, or 0.7, just under the threshold.

If we try to insert one more element, we would have a total of 8 elements in our hash table and a load factor of 0.8. Because this would cause the load factor to exceed the maximum load factor, we must resize the underlying array to length 20 before we insert the element. Remember that since our procedure for locating an entry in the hash table is to take the `hashCode() % array.length` and our array's length has changed from 10 to 20, all the elements in the hash table need to be relocated. Once all the elements have been relocated and our new element has been added, we will have a load factor of 8/20, or 0.4, which is below the maximum load factor.

Previous
19.4 Handling Collisions: Linear Probing and External Chaining

Next
19.6 Summary

Last updated 1 year ago