Other Open Addressing Techniques

2 min

There are more sophisticated ways to find the next address after a hash collision, although anything too calculation-intensive would negatively affect a hash table's performance. Linear probing systems, for instance, could jump by five steps instead of one step.

In a quadratic probing open addressing system, we add increasingly large numbers to the hash code. At the first collision we just add 1, but if the hash collides there too we add 4 ,and the third time we add 9. Having a probe sequence change over time like this avoids clustering.

Clustering is what happens when a single hash collision causes additional hash collisions. Imagine a hash collision triggers a linear probing sequence to assigns a value to the next hash bucket over. Any key that would hash to this "next bucket" will now collide with a key that, in a sense, doesn't belong to that bucket anyway.

As a result the new key needs to be assigned to the next, next bucket over. This propagates the problem because now there are two hash buckets taken up by key-value pairs that were assigned as a result of a hash collision, displacing further pairs of information we might want to save to the table.

Instructions

What other ways can we iterate through addresses in our underlying way? Maybe we could write a function that takes the key and returns a number of steps to jump to get to the next index. What would you call a function that transforms a string into a number in this way?