Level 5

Hashing

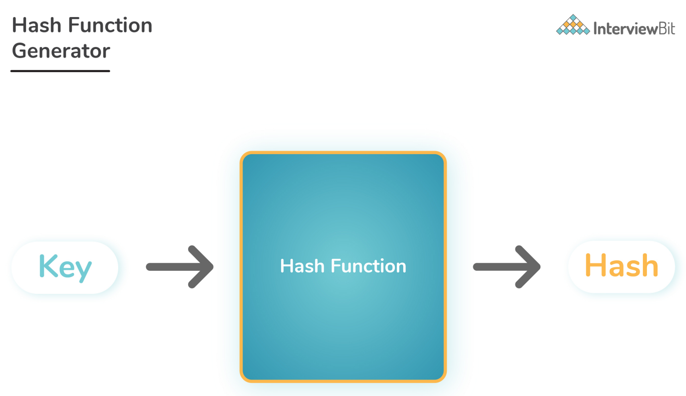
Go To Problems (/courses/programming/topics/hashing/#problems)

Introduction To Hashing

What is hashing? How does it work?

Hashing is the process of converting a given key into another smaller value for O(1) retrieval time.

- This is done by taking the help of some function or algorithm which is called as **hash function** to map data to some encrypted or simplified representative value which is termed as “hash code” or “hash”. This hash is then used as an index to narrow down search criteria to get data quickly.



- Hashing can be considered as a significant improvement over DAT to reduce the space complexity.
- In our example of employee system that we have seen in the Introduction part, we can simply pass the employee ID to our hash function, get the hash code and use it as key to query over records.
- Let us see one more example to understand how hashing works.
- Purely as an example to help us grasp the concept, let us suppose that we want to map a list of string keys to string values (for example, map a list of countries to their capital cities). So let's say we want to store the data in Table 1 in the map.

```
Key              |    Value
-----------------|-------------
Cuba             |    Havana
England          |    London
France           |    Paris
Spain            |    Madrid
Switzerland      |    Berne
```

And let us suppose that our hash function is to simply take the *length of the string*.

For simplicity, we will have two arrays: one for our keys and one for the values. So to put an item in the hash table, we compute its hash code (in this case, simply count the number of characters), then put the key and value in the arrays at the corresponding index.
For example, Cuba has a hash code (length) of 4. So we store Cuba in the 4th position in the keys array, and Havana in the 4th index of the values array etc. And we end up with the following:

```
Position             |   Keys array     |  Values array
(hash = key length)  |                  |
---------------------|------------------|---------------
   1                 |                  |
   2                 |                  |
   3                 |                  |
   4                 |    Cuba          |    Havana
   5                 |    Spain         |    Madrid
   6                 |    France        |    Paris
   7                 |    England       |    London
   8                 |                  |
   9                 |                  |
   10                |                  |
   11                |  Switzerland     |    Berne
```

Now, in this specific example things work quite well. Our array needs to be big enough to accommodate the longest string, but in this case that's only 11 slots. And we do waste a bit of space because, for example, there's no 1-letter keys in our data, nor keys between 8 and 10 letters. But in this case, the waste isn't so bad either. And taking the length of a string is nice and fast, so so is the process of finding the value associated with a given key (certainly faster than doing up to five string comparisons)1.

We can also easily see that this method would not work for storing arbitrary strings. If one of our string keys was a thousand characters in length but the rest were small, we would waste the majority of the space in the arrays. More seriously, this model can't deal with collisions: that is, what to do when there is more than one key with the same hash code (in this case, more than one string of a given length). For example, if our keys were random words of English, taking the string length would be fairly useless. Granted, the word "psuedoantidisestablishmentarianistically" would probably get its own place in the array. But on the other hand, we'd be left with thousands of, say, 6-letter words all competing for the same slot in the array.

In this topic, we explore hashing, a technique very widely used in interview questions. Hashing is designed to solve the problem of needing to efficiently find or store an item in a collection.
For example, if we have a list of 10,000 words of English and we want to check if a given word is in the list, it would be inefficient to successively compare the word with all 10,000 items until we find a match. Hashing is a technique to make things more efficient by effectively narrowing down the search at the outset.