Hash : How does it work internally?

Ask Question

This might sound as an very vague question upfront but it is not. I have gone through Hash Function description on wiki but it is not very helpful to understand.

I am looking simple answers for rather complex topics like Hashing. Here are my questions:

1. What do we mean by hashing? How does it work internally?
2. What algorithm does it follow ?
3. What is the difference between `HashMap` , `HashTable` and `HashList` ?
4. What do we mean by 'Constant Time Complexity' and why does different implementation of the hash gives constant time operation ?
5. Lastly, why in most interview questions `Hash` and `LinkedList` are asked, is there any specific logic for it from testing interviewee's knowledge?

I know my question list is big but I would really appreciate if I can get some clear answers to these questions as I really want to understand the topic.

java algorithm data-structures hash

edited Jun 12 '17 at 5:42
Sanket Makani
2,237 1 10 23

asked Dec 15 '10 at 18:25
Rachel
35.3k 97 235 342

3 Try Hash table on Wikipedia instead. A hash function is used as part of the process, but does not explain "how" a hash table works. – user166390 Dec 15 '10 at 19:05

There is no such thing as a `HashList` in Java or any other language I am aware of. Don't use code formatting for text that isn't code. – EJP Jun 12 '17 at 5:27

6 Answers

1. Here is a good explanation about hashing. For example you want to store the string "Rachel" you apply a hash function to that string to get a memory location. `myHashFunction(key: "Rachel" value: "Rachel") --> 10`. The function may return 10 for the input "Rachel" so assuming you have an array of size 100 you store "Rachel" at index 10. If you want to retrieve that element you just call `GetmyHashFunction("Rachel")` and it will return 10. Note that for this example the key is "Rachel" and the value is "Rachel" but you could use another value for that key for example birth date or an object. Your hash function may return the same memory location for two different inputs, in this case you will have a collision you if you are implementing your own hash table you have to take care of this maybe using a linked list or other techniques.
2. Here are some common hash functions used. A good hash function satisfies that: each key is equally likely to hash to any of the n memory slots independently of where any other key has hashed to. One of the methods is called the division method. We map a key k into one of n slots by taking the remainder of k divided by n. `h(k) = k mod n`. For example if your array size is `n = 100` and your key is an integer `k = 15` then `h(k) = 10`.
3. Hashtable is synchronised and Hashmap is not. Hashmap allows null values as key but Hashtable does not.
4. The purpose of a hash table is to have O(c) constant time complexity in adding and getting the elements. In a linked list of size N if you want to get the last element you have to traverse all the list until you get it so the complexity is O(N). With a hash table if you want to retrieve an element you just pass the key and the hash function will return you the desired element. If the hash function is well implemented it will be in constant time O(c) This means you dont have to traverse all the elements stored in the hash table. You will get the element "instantly".
5. Of couse a programer/developer computer scientist needs to know about data structures and complexity =)

edited Jun 24 '15 at 12:01
Sergey Maksimenko
479 5 17

answered Dec 15 '10 at 18:40
Enrique
6,899 6 37 52

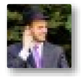
Both the links you have provided take me to wiki page which i have already visited and have mentioned in question that i have gone through them so can you update your first 2 points ? – Rachel Dec 15 '10 at 18:43

Thanks for updating the answer, now am able to get more out of it. – Rachel Dec 15 '10 at 19:19

There is no such thing as 'O(c) time complexity': you mean O(1). – EJP Jun 12 '17 at 5:26

1. Hashing means generating a (hopefully) unique number that represents a value.
2. Different types of values (`Integer` , `String` , etc) use different algorithms to compute a hashcode.
3. HashMap and HashTable are *maps*; they are a collection of unqiue keys, each of which is associated with a value.
   Java doesn't have a HashList class. A Hash*Set* is a set of unique values.
4. Getting an item from a hashtable is constant-time with regard to the size of the table.
   Computing a hash is not necessarily constant-time with regard to the value being hashed.
   For example, computing the hash of a string involves iterating the string, and isn't constant-time with regard to the size of the string.
5. These are things that people ought to know.

answered Dec 15 '10 at 18:31


SLaks
649k 131 1574
1707

@Slaks: So hashing would always generate a unique number ? – Rachel Dec 15 '10 at 18:33

2 No, it won't. It is not possible to generate a unique *32-bit* number for every possible string. That's why collisions exist. – SLaks Dec 15 '10 at 18:35

Ok. What would be an algorithm to compute `hashcode of long` – Rachel Dec 15 '10 at 18:37

@Rachel: Hash don't try to generate unique number. It tries to create a homogenous distribution for ouput, so that each output value would have roughly `1/nr_of_possible_hash_values` probability. – ruslik Dec 15 '10 at 18:57

3 @Rachel - what @ruslik means is this: a hash outputs a number, and that number is within a specific range of numbers. As an example, that number might be between 0 and 2^32-1. When you hash some data, a number in that range is returned, and ideally every single possible number is equally likely to be returned. The reason you want this for Hash Tables/Maps is that the table can be thought of as an array. When you hash a value, you use that number as the index in the array. You want to avoid using the same index twice. If the numbers a uniformly distributed then a collision is less likely. – Niki Yoshiuchi Dec 15 '10 at 19:43

1. Hashing is transforming a given entity (in java terms - an object) to some number (or sequence). The hash function is not reversable - i.e. you can't obtain the original object from the hash. Internally it is implemented (for `java.lang.Object` by getting some memory address by the JVM.
2. The JVM address thing is unimportant detail. Each class can override the `hashCode()` method with its own algorithm. Modren Java IDEs allow for generating good hashCode methods.
3. Hashtable and hashmap are the same thing. They key-value pairs, where keys are hashed. Hash lists and hashsets don't store values - only keys.
4. Constant-time means that no matter how many entries there are in the hashtable (or any other collection), the number of operations needed to find a given object by its key is constant. That is - 1, or close to 1
5. This is basic computer-science material, and it is supposed that everyone is familiar with it. I think google have specified that the hashtable is the most important data-structure in computer science.

answered Dec 15 '10 at 18:37

Bozho
463k 102 912 1037

Can you example algorithm implementation for generating hashcode function from long ? Also I was asked in interview what is the algorithm for generating hashcode, I was not sure of currently internal working and so wanted to understand how is it done internally. – Rachel Dec 15 '10 at 18:39

2 you can look at `java.lang.Long` 's documentation for that - the code is one-line: `return (int)(value ^ (value >>> 32));` – Bozho Dec 15 '10 at 18:40

I'll try to give simple explanations of hashing and of its purpose.

First, consider a simple list. Each operation (insert, find, delete) on such list would have O(n) complexity, meaning that you have to parse the whole list (or half of it, on average) to perform such an operation.

Hashing is a very simple and effective way of speeding it up: consider that we split the whole list in a set of small lists. Items in one such small list would have something in common, and this something can be deduced from the key. For example, by having a list of names, we could use first letter as the quality that will choose in which small list to look. In this way, by partitioning the data by the first letter of the key, we obtained a simple hash, that would be able to split the whole list in ~30 smaller lists, so that each operation would take O(n)/30 time.

However, we could note that the results are not that perfect. First, there are only 30 of them, and we can't change it. Second, some letters are used more often than others, so that the set with `Y` or `Z` will be much smaller that the set with `A`. For better results, it's better to find a way to partition the items in sets of roughly same size. How could we solve that? This is where you use hash functions. It's such a function that is able to create an arbitrary number of partitions with roughly the same number of items in each. In our example with names, we could use something like

```
int hash(const char* str){
    int rez = 0;
    for (int i = 0; i < strlen(str); i++)
        rez = rez * 37 + str[i];
    return rez % NUMBER_OF_PARTITIONS;
};
```

This would assure a quite even distribution and configurable number of sets (also called buckets).

edited May 22 '17 at 0:01
Arya McCarthy
4,129 3 13 34

answered Dec 15 '10 at 19:24
ruslik
11.9k 1 28 36

Consider the problem of searching an array for a given value. If the array is not sorted, the search might require examining each and all elements of the array. If the array is sorted, we can use the binary search, and therefore reduce the worse-case runtime complexity to O(log n). We could search even faster if we know in advance the index at which that value is located in the array. Suppose we do have that magic function that would tell us the index for a given value. With this magic function our search is reduced to just one probe, giving us a constant runtime O(1). Such a function is called a hash function . A hash function is a function which when given a key, generates an address in the table.

answered Apr 24 '16 at 5:16
Pramod Talwar
21 1

> What do we mean by Hashing, how does it work internally ?

Hashing is the transformation of a string shorter fixed-length value or key that represents the original string. It is not indexing. The heart of hashing is the hash table. It contains array of items. Hash tables contain an index from the data item's key and use this index to place the data into the array.

> What algorithm does it follow ?

In simple words most of the Hash algorithms work on the logic "index = f(key, arrayLength)"

> Lastly, why in most interview questions Hash and LinkedList are asked, is there any specific logic for it from testing interviewee's knowledge ?

Its about how good you are at logical reasoning. It is most important data-structure that every programmers know it.

answered Dec 15 '10 at 18:45
user517400