22 Hashed Collections

22.1 Introduction

As we have seen, hashing provides very fast (effectively $O(1)$) algorithms for inserting, deleting, and searching collections of key-value pairs. It is therefore an excellent way to implement maps. But it also provides a very efficient way to implement sets.

22.2 Hash Tablets

A hash table is designed to store key-value pairs by the hash value of the key. But suppose we wanted to store keys alone, with no accompanying value. We can then simply store keys in the hash table by the hash value of the key. We call such a "degenerate" form of hash table a *hash tablet*. Note that several of the hash table examples in the previous chapter about hashing are really hash tablets. Figure 1 shows the features of a HashTablet class.

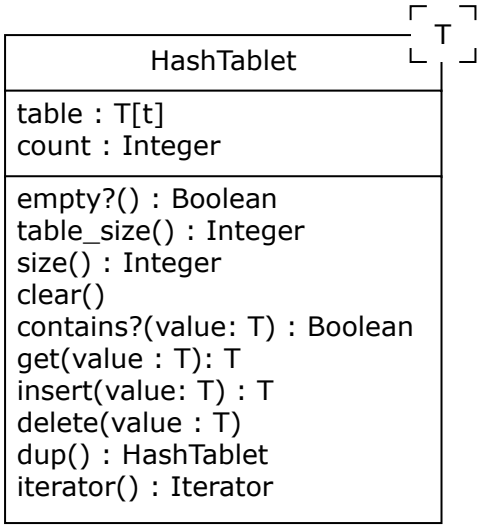

Figure 1: A HashTablet Class

The HashTablet holds a table array into which it hashes values of type T (this table can actually have some other type—for example, if chaining is used, it might be a reference to a node or a node type). The HashTablet allows values to be added, removed, and searched for. The insert() operation replaces the value if it is already in the table, and returns the new element. The delete() operation does nothing if the value is not present in the HashTablet. The get() operation returns the value in the table that matches the value argument. This operation seems pointless, but if T is a key-value pair and values are found by comparing keys, the argument to get() may have a dummy value, while the pair returned from the table may contain a legitimate value. This is how values can be retrieved by their keys.

A HashTablet also provides an operation to make a shallow copy of the entire instance (useful for implementing sets), and it provides an iterator. The HashTablet class provides an implementation data structure, like the BinarySearchTree class, so it is not part of the Container hierarchy.

22.3 HashSets

Hash tablets are a convenient implementation data structure because they can be used to implement hashed collections of various kinds, not just maps. For example, suppose we wish to use hashing to implement sets. If we use a hash table to do this, then the key is the set element we wish to store, but what is the value in the key-value pair? There is none—we must simply ignore this field in the key-value pairs stored in the hash table, wasting the space. On the other hand, if we use a hash tablet then there is no value field, so no space is wasted. Hash tablets are thus a good way to implement hash sets.

A HashSet implements the Set interface and holds a HashTablet as a private attribute. The basic Collection operations are easily realized using HashTablet operations; for example, a HashSet is empty if its HashTablet's size is 0; clearing a HashSet is done by clearing its HashTablet; iterating over the HashSet is done by iterating over the HashTablet, and so forth.

HashSet union and complement operations may be implemented by copying the host HashSet (and its HashTablet) and iterating over the argument Set, adding (for union) or removing (for complement) all its values; intersection is done by iterating over one Set and adding to the result HashSet only those values that the other contains.

22.4 HashMaps

Hash tablets can be used to implement other collections as well. Consider maps. We can define a Pair class with key and value attributes, and comparison and hash operations that use only the key attribute. We can then implement a map using a hash tablet that stores instances of this Pair class. The hash tablet operations will store, delete, and retrieve instances via the key because the Pair class's operations have been defined to work this way, and the retrieved instances can be queried to obtain the value paired with a key, so all map operations can be implemented. Most of the work will be done by the hash tablet, so it is easy to code the hash map.

A HashMap implements the Map interface and holds a HashTablet as a private attribute. The HashTablet stores instances of a Pair class with a key attribute of type K and a value attribute of type T, and it has hash and comparison operations that work on the key attribute.

22.5 Implementing Hashed Collections in Ruby

The Ruby `Object` class includes a `hash()` operation that produces a `Fixnum` value for every object. This value can be modified using the modulo function for any hash table. Also, the `hash()` function can be overridden to provide appropriate hash values for a `Pair` class, for example.

Ruby has a built-in HashMap class called `Hash`. This class already implements the entire Map interface except the contains?() and iterator() operations. The former is a synonym for `has_value()`, which is in `Hash`, and the latter is easily written. Thus the easiest way to implement a HashMap class in Ruby is to sub-class `Hash`. A HashSet could be implemented in Ruby using a HashTablet as discussed above, or using the built-in `Hash` class. The former approach will use less space, but the latter will probably be faster because the built-in type is implemented very efficiently in the Ruby interpreter.

22.6 Summary and Conclusion

Hashing is an efficient way to implement sets as well as maps. A degenerate form of hash table that we call a hash tablet is a useful implementation data structure that makes implementing hashed collections quite easy. Ruby provides a fast built-in `Hash` type that already implements hash maps and can easily be used to implement hash sets.

22.7 Review Questions

1. What is the difference between a hash table and a hash tablet?
2. What sorts of collision resolution techniques can be used in a hash tablet?
3. Does a HashSet inherit from HashTablet?
4. Why is a Pair class needed when a HashTablet is used to implement a HashMap?
5. What does the hash_function() of a Pair class do?

22.8 Exercises

1. Begin writing a `HashTablet` class in Ruby by writing the attributes, invariant, and `initialize()` operation for the class. This will require that you decide on a collision resolution strategy.
2. Continue writing a `HashTablet` class in Ruby by implementing the `size()`, `tableSize()`, `clear()`, `copy()`, and `iterator()` operations.
3. Finish implementing the `HashTablet` class in Ruby by implementing the remaining operations. Note that values stored in the table must have a hash function that `HashTablet` can use.
4. Assume that a `HashTablet` class is available and begin implementing a `HashSet` class in Ruby by writing the attributes, invariant, and `initialize()` operation for this class.
5. Continue implementing a `HashSet` class in Ruby by writing the `contains?()`, `[]=`, and `delete()` operations of this class.
6. Using a `HashTablet` to implement a `HashMap` requires that a `Pair` class storing a key and a value be created. Write such a `Pair` class in Ruby and include a hashing function and comparison operators for the class.
7. Assume that a `HashTablet` class is available and begin implementing a `HashMap` class in Ruby by writing the attributes, invariant, and `initialize()` operation for this class. You will need to use the `Pair` class from the last exercise.
8. Continue implementing a `HashMap` class by writing the `contains?()`, `[]=`, and `[]` operations of this class.
9. When discussing ArrayList in Chapter 10 we suggested that implementing an ArrayList in Ruby could be done quite simply by sub-classing the built-in Ruby `Array` class. Implement the HashMap class by sub-classing the built-in `Hash` class.

22.9 Review Question Answers

1. A hash table stores key-value pairs by hashing the key. A hash tablet stores keys only using hashing. A hash tablet is thus a degenerate or simplified version of a hash table.
2. Any sort of collision resolution techniques can be used in a hash tablet.
3. A HashSet does not inherit from HashTablet, but it contains a HashTablet attribute and the HashTablet ends up doing most of the work when the HashSet is implemented. The HashSet thus *delegates* work to the HashTablet.

4. A Pair class is needed when a HashTablet is used to implement a HashMap because a HashTablet only stores one value, not a key-value pair. The HashTablet can be made to work as if it stored pairs by defining a Pair class and storing instances of the Pair class in the HashTablet. Thus a degenerate hash table can be made to work like a full-fledged hash table quite simply.
5. The hash_function() of a Pair class computes a hash value from the key attribute of the Pair class. This allows the HashTablet to store and retrieve the Pair based on the key.