Data Structures and Algorithms in Java™

Sixth Edition

Michael T. Goodrich
Department of Computer Science
University of California, Irvine

Roberto Tamassia
Department of Computer Science
Brown University

Michael H. Goldwasser
Department of Mathematics and Computer Science
Saint Louis University

Study Guide: Hints to Exercises

WILEY

Chapter 10 Maps, Hash Tables, and Skip Lists

Hints

Reinforcement

R-10.1) The first insertion is $O(1)$, the second is $O(2)$, ...

R-10.2) Use the PositionalList.remove method to delete an entry from the map.

R-10.3) Take advantage of the existing findIndex method.

R-10.4) Think about which of the schemes use the array supporting the hash table exclusively and which of the schemes use additional storage external to the hash table.

R-10.5) There is a lot of symmetry and repetition in this string, so avoid a hash code that would not deal with this.

R-10.6) Try to mimic the figure in the book.

R-10.7) Try to mimic the figure in the book.

R-10.8) The failure occurs because no empty slot is found. For the drawing, try to mimic the figure in the book.

R-10.9) Try to mimic the figure in the book.

R-10.10) Think of the worst-case time for inserting every entry in the same cell in the hash table.

R-10.11) Mimic the way the figure is drawn.

R-10.12) Combine the hash values of the two components to make a hash value for the pair.

R-10.13) There is a subtle flaw—but can you find it?

R-10.14) The load factor can be controlled from within the abstract class, but there must be means for setting the parameter (either through the constructor, or a new method).

R-10.15) It is okay to insert a new entry on "top" of the deactivated entry object.

R-10.16) You will need to keep track of the number of probes in order to apply quadratic probing.

R-10.17) Think of where the entry with minimum key is stored.

R-10.18) Since the map will still contain n entries at the end, you can assume that each remove( ) operation takes the same asymptotic time.

R-10.19) Take advantage of the existing findIndex method.

R-10.20) The crucial methods are get and put.

R-10.21) In the new code, what happens when key equals table.get(mid)?

R-10.22) Assume that a skip list is used to implement the sorted map.

R-10.23) Mimic the style of the figures in the book.

R-10.24) You must link out the removed entry's tower from all the lists it belongs to.

R-10.25) Compare to the implementation of the addAll method given in Section 10.5.1.

R-10.26) Compare to the implementation of the addAll method given in Section 10.5.1.

R-10.27) Recall that most skip-list operations run in $O(\log k)$ time for a set with size k.

R-10.28) Recall that most hash-table operations run in $O(1)$ expected time.

R-10.29) Recall that most hash-table operations run in $O(1)$ expected time.

R-10.30) Recall that most hash-table operations run in $O(1)$ expected time.

R-10.31) Something from this chapter should be helpful!

Creativity

C-10.32) Consider a bootstrapping method for finding the primes up to $\sqrt{2M}$.

C-10.33) Your solution should make a single call to findIndex.

C-10.34) You may assume that such a method is supported by the auxiliary UnsortedTableMap

C-10.35) You must only call the findSlot utility once.

C-10.36) Model your solution after the existing support for other map methods.

C-10.37) You may assume that such a method is supported by the auxiliary UnsortedTableMap

C-10.38) The heart of the process can be performed by the findSlot utility.

C-10.39) When might the load factor fall below the threshold, and how can you detect this from within AbstractHashMap?

C-10.40) Start by defining an appropriate subclass of AbstractMap.MapEntry that includes the new next field that is desired.

C-10.41) You need to do some shifting of entries to close up the "gap" just made, but you should only do this for entries that need to move.

C-10.42) For part a, note that the symmetry will halve the range of possible values. For part b, note that such automatic collisions will not occur.

C-10.43) Perhaps borrow techniques from the Progression hierarchy of Chapter 2, or use Java's iterators.

C-10.44) Maintain a secondary PositionalList instance that represents the FIFO order, and store positions within that list with each entry of the hash table.

C-10.45) Each map entry instance should store its current index within the table.

C-10.46) Each map entry instance should store its current index within the table.

C-10.47) Each map entry instance should store its position within the bucket list.

C-10.48) Try out some examples.

C-10.49) Do a "double" binary search.

C-10.50) Dovetail two binary searches.

C-10.51) Do a lazy iteration through indices of the underlying array list.

C-10.52) Manage a primary iteration through all nonempty buckets, and then a secondary iteration through each element of a bucket.

C-10.53) Do a lazy iteration through the nonempty cells of the table.

C-10.54) Think about first sorting the pairs by cost.

C-10.55) In the insertion algorithm, first repeatedly flip the coin to determine the level where you should start inserting the new entry.

C-10.56) Consider augmenting each node v in a higher level with the number of missing entries in the gap from v to the next node over.

C-10.57) Consider augmenting each node v in a higher level with the number of missing entries in the gap from v to the next node over.

C-10.58) Think first about how you can determine the number of 1's in any row in $O(\log n)$ time.

C-10.59) Consider a two-pass solution, with use of an auxiliary structure.

C-10.60) Think of describing your algorithms in terms of boolean operations on the bit vectors.

C-10.61) Think of how you could transform D into L.

C-10.62) Consider the way subMap was implemented for SortedTableMap.

C-10.63) You need some way of grouping together entries with the same key.

C-10.64) You need some way of grouping together elements with the same key.

Projects

P-10.65) The biggest challenge is detecting the case of an infinite loop.

P-10.66) When searching for an existing key, make sure to consider both of the possible buckets.

P-10.67) Maintain a secondary PositionalList instance that represents the FIFO order, and store positions within that list with each entry of the hash table.

P-10.68) We've already implemented them for you; all that's left is the experiments.

P-10.69) In a Unix/Linux system, a good place to start is /usr/dict.

P-10.70) It is okay to generate these phone numbers more-or-less at random.

P-10.71) Sentinels can be used in place of the theoretical $-\infty$ and $+\infty$.

P-10.72) Try to make your screen images mimic the skip list figures in the book.

P-10.73) You need to find some way to let the intermediate nodes in the skip list keep track of the number of elements that have been skipped over.

P-10.74) For each word t that results from a minor change to s, you can test if t is in W in $O(1)$ time.