**Chapter 10**

**R-10.1 Give a concrete implementation of the pop method in the context of the MutableMapping class, relying only on the five primary abstract methods of that class.**

**def pop(self, end):**

**if self = None:**

**return False**

**key = next(reversed(self)**

**return key, MutableMapping.pop(self, key)**

**R-10.4 What is the worst-case running time for inserting n key-value pairs into an initially empty map M that is implemented with the UnsortedTableMap class?**

On^2

**R-10.9 Draw the 11-entry hash table that results from using the hash function, h(i)=(3i+5) mod 11, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining**

**(in drawing)**

**C-10.42 Suppose that each row of an n×n array A consists of 1’s and 0’s such that, in any row of A, all the 1’s come before any 0’s in that row. Assuming A is already in memory, describe a method running in O(nlogn) time (not O(n2) time!) for counting the number of 1’s in A.**

Each row of the array A has Os and 1s. In any row, all of the 1s are before any 0s. Using a binary search algorithm, get the index of the last 1 in each row. The search will take Ologn time. For n rows, the time is Onlogn time.

**Chapter 11**

**R-11.2 Insert, into an empty binary search tree, entries with keys 30, 40, 24, 58, 48, 26, 11, 13 (in this order). Draw the tree after each insertion**

**(in drawing)**

**R-11.5 Dr. Amongus claims that the order in which a fixed set of entries is inserted into an AVL tree does not matter—the same AVL tree results every time. Give a small example that proves he is wrong.**

**(in drawing)**

**R-11.8 Draw the AVL tree resulting from the insertion of an entry with key 52 into the AVL tree of Figure 11.14b.**

**(in drawing)**

**C-11.29 Explain how to use an AVL tree or a red-black tree to sort n comparable elements in O(nlogn) time in the worst case.**

Insert n elements into the tree. Remove the minimum element from the tree with a search. Start with ordering the elements.

Insert: Ologn

Total inset: Onlogn

Search and remove: Ologn

Total time: Onlogn

