CSE332 Week 4 Section Worksheet

1. For each of the following types of hash-tables, insert the following numbers into an initially empty table of size 10 (use h(k) = k%tableSize as the primary hash function):

23, 27, 94, 33, 14, 3

1. Separate Chaining (with each bucket being an unsorted linked list)
2. Double Hashing, with h2(k) = 7- (k%7) as the secondary hash function

When done, give the load factor for the resulting table.

The load factor is the same for both: .6 (6 elements divided by a table size of 10).

1. In 1.b., what’s wrong with h2(k) = k%7 as a secondary hash function?

When k is a multiple of 7 it evaluates to 0 - this is very problematic for a secondary hash function. That means the 2nd, 3rd, 4th, etc. indices checked would all be the same as the initial index.

1. Imagine we have a hash table with a poorly chosen primary hash function (or we just get very unlucky with our insertions) and all the keys are mapped to the same index by our primary hash function. How would this affect a table using separate chaining? How would this affect a table using double hashing (assume that the secondary hash function distributes the keys fairly evenly)?

With separate chaining we’d have all elements in a single unsorted linked list – operations would take linear time, in the worst case.

If the table used double hashing, we’d have a collision at first, but the location after that would depend on the secondary hash function. If the secondary hash function were pretty good, we would get a fairly good distribution of indices after that initial index.

1. Insert the following into an initially empty binary min heap. Draw the resulting heap and write its array representation:

12, 14, 3, 8, 30, 9, 2, 7