Solution Set 4

Solutions to problem set 4.

Solution 1

a- Kim is unfortunately using a bad hash function. Her function computes the sum of integers from **1** to **n** and hashes **n** to the slot numbered **sum**. To calculate the hash of a given *k*, the loop would run for k steps. The function thus has a complexity of O(k). Note, how a hash table using this hash function would fail to provide O(1) insert and retrieval operations.

b- The hash function is computing the sum of numbers from 1 to k. We already know that the summation of consecutive numbers from 1 to k can be represented by the following formula

> $$sum = \frac{k(k+1)}{2}$$

The hash function can compute the above formula and return the hashed value of the input key. The above computation would take constant time when worked out.

c- If Kim wants to avoid collisions she will want to make sure that the biggest key can hash to a slot without collision. For example, if the biggest key is equal to 10, then it's hash would be:

> $$\frac{10(10+1)/2}{2} = 55$$

If Kim uses an array as a hash table, then it must be big enough to accommodate the biggest key. The array size should be at least $k(k+1)/2$ elements long in length which is equivalent of saying the space elements long in length which is equivalent of saying the space is

$$O(k^2)$$

where k is value of the biggest key expected.