CS168 Spring Assignment 1

SUNet ID(s): johngold jsalloum
Name(s): John Gold Justin Salloum
Collaborators: None

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Part B

(a) The first strategy has the merit of being the simplest, but it performs very poorly compared to the other strategies. Surprisingly, the second strategy performs better than the 3rd strategy. However, the 4th is the best and stands out as the ”sweet spot”. For all 30 trials, the maximum number of elements in a bucket was 3.

Part C

(a) The first strategy, where we choose a bucket at random, is most similar to hashing. The hash function is similar to a ”random” choice of a bucket. Our simulation could also be though of as trying to hash 100,000 elements into 100,000 buckets, and the first strategy would have buckets with up to 10 elements chained together (in a linked list). This is inefficient compared to the other strategies where no bucket would have more than 3 or 4 elements chained together. This could be implemented by using a second hash function after we found a collision, similar to generating a second random number/bucket. To further adapt this to our 4th ”sweet spot” strategy, we have one hash function that returns a number between (0, N/2), and another that returns a number betwen (N/2, N).

(b) Going forward with the proposed two hash function method would require some trade-offs. This tradeoff really depends on how long it takes to calculate the hash of a value, which is now done twice instead of once. The process of inserting and lookup, especially in the worst case should take less operations. Instead of worst case lookup through a 10-chain, it would instead involve looking through 2 3-chains.

Figure 1: Strategy 1

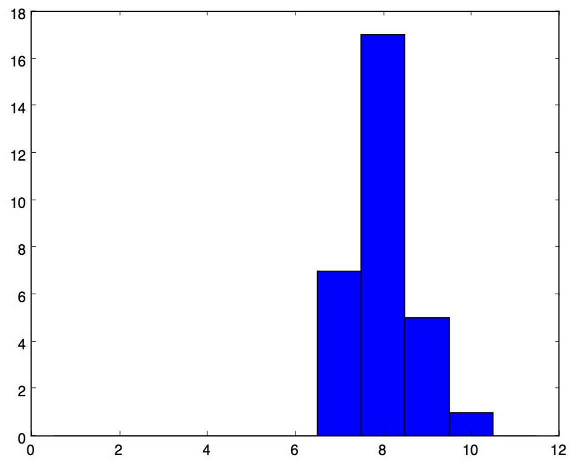


Figure 2: Strategy 2

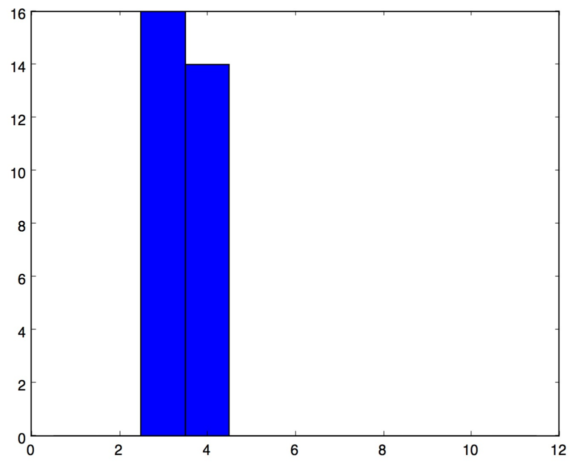


Figure 3: Strategy 3

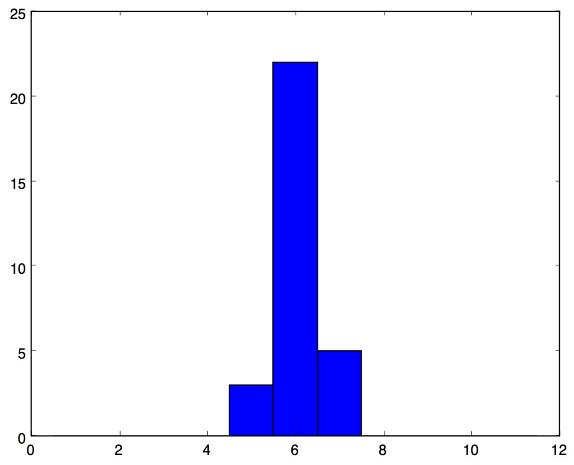
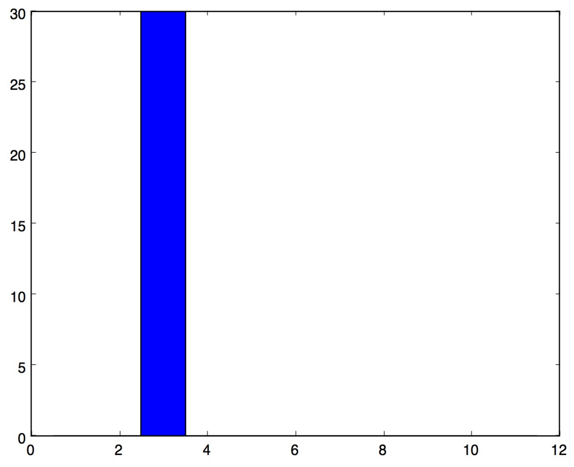


Figure 4: Strategy 4



0
5
10
15
20
25
30
0
2
4
6
8
10
12