CSE100 Algorithm Design and Analysis Discussion Ch 11 Spring 2019
Last update: 2/26/2019

You're expected to work on the problems before coming to the lab. Discussion session is not meant to be a lecture. TAs will guide the discussion and correct your solutions if needed. We may not release solutions for all problems. If you're better prepared for discussion, you will learn more. TAs will record names of the students who actively engage in discussion and report them to the instructor. The instructor will factor in participation in final grade.

1. (Basic) What is the main advantage of hash tables over direct-address tables?
2. (Basic) What is the assumption of simple uniform hashing?
3. (Basic) Why the hash function $h(k) = k \bmod 2^L$ for some integer L is not desirable?
4. (basic) Consider a hash table with $m = 11$ slots and using the hash function $h(k) = k$ mod 11. Say we insert elements $k = 41, 18, 3, 8, 19, 5, 1, 7$ in this order. Show the final table in the following two cases. In both cases the same hash function $h(k)$ is used.
   (a) When chaining is used to resolve collisions. Please insert the element at the beginning of the linked list.
   (b) When open addressing and linear probing are used to resolve collisions.

   Sol. Chaining:
   Pos 0:
   Pos 1: 1 $\rightarrow$
   Pos 2:
   Pos 3: 3$\rightarrow$
   Pos 4:
   Pos 5: 5$\rightarrow$
   Pos 6:
   Pos 7: 7 $\rightarrow$ 18$\rightarrow$
   Pos 8: 19$\rightarrow$ 8 $\rightarrow$ 41$\rightarrow$
   Pos 9:
   Pos 10:

   Open Addressing:
   Pos 0: 7
   Pos 1: 1
   Pos 2:
   Pos 3: 3
   Pos 4:
   Pos 5: 5
   Pos 6:
   Pos 7: 18
   Pos 8: 41
   Pos 9: 8
   Pos 10:19

5. (advanced) You are given two sequences, $\langle a_1, a_2, ..., a_n \rangle$ and $\langle b_1, b_2, ..., b_n \rangle$, where each sequence consists of distinct integers. Describe a linear time algorithm (in the average case) that tests if a sequence is a permutation of the other. Assume that the simple uniform hashing assumption holds. Explain the running time of your algorithm.

   Sol. We create a hash table of size $\Theta(n)$ and use chaining to resolve collisions. Recall that under the simple uniform hashing assumption, a search, either successful or unsuccessful, take $O(1)$ time on average. We first insert $a_1, a_2, ..., a_n$ into the hash table. This is done in $O(n)$ time in the average case. Then, we search each b_i in the hash table, which is done in $O(1)$ time. If we can find every b_i in the hash table, it, together with the fact that each of the two sequences consists of distinct integers, one is a permutation of the other. The running time is clearly $O(n)$.