

# VE 281 Homework 2 (Due 10/31 11:59PM)

Name: \_\_\_\_\_

ID: \_\_\_\_\_

## • Exercise 1. Open Addressing

Consider inserting the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table of length  $m = 11$  using open addressing with the auxiliary hash function  $h(k) = k$ . Illustrate the result of inserting these keys using linear probing, using quadratic probing with  $c_1 = 1$  and  $c_2 = 3$ , and using double hashing with  $h_1(k) = k$  and  $h_2(k) = 1 + (k \bmod (m - 1))$ .

### Solution:

We use  $T_t$  to represent each time stamp  $t$  starting with  $i = 0$ , and if encountering a collision, then we iterate  $i$  from  $i = 1$  to  $i = m - 1 = 10$  until there is no collision.

Linear probing:

$h(k, i) = (k + i) \bmod 11$		$T_0$	$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$T_6$	$T_7$	$T_8$
0	mod 11		22							
1	mod 11									
2	mod 11									
3	mod 11									
4	mod 11									
5	mod 11									
6	mod 11									
7	mod 11									
8	mod 11									
9	mod 11									
10	mod 11	10	10							

Quadratic probing:

$h(k, i) = (k + i + 3i^2) \bmod 11$		$T_0$	$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$T_6$	$T_7$	$T_8$
0	mod 11		22							
1	mod 11									
2	mod 11									
3	mod 11									
4	mod 11									
5	mod 11									
6	mod 11									
7	mod 11									
8	mod 11									
9	mod 11									
10	mod 11	10	10							

Double hashing:

$h(k, i) = (k + i(1 + k \bmod 10)) \bmod 11$		$T_0$	$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$T_6$	$T_7$	$T_8$
0	mod 11		22							
1	mod 11									
2	mod 11									
3	mod 11									
4	mod 11									
5	mod 11									
6	mod 11									
7	mod 11									
8	mod 11									
9	mod 11									
10	mod 11	10	10							

• **Exercise 2. Slot-Size Bound for Chaining**

Suppose that we have a hash table with  $n$  slots, with collisions resolved by chaining, and suppose that  $n$  keys are inserted into the table. Each key is equally likely to be hashed to each slot. Let  $M$  be the maximum number of keys in any slot after all the keys have been inserted. Your mission is to prove an  $O(\lg n / \lg \lg n)$  upper bound on  $E[M]$ , the expected value of  $M$

a. Argue that the probability  $Q_k$  that exactly  $k$  keys hash to a particular slot is given by

$$Q_k = \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \binom{n}{k}$$

b. Let  $P_k$  be the probability that  $M = k$ , that is, the probability that the slot containing the most keys contains  $k$  keys. Show that  $P_k \leq nQ_k$ .

c. Use Stirling's approximation to show that  $Q_k < e^k / k^k$ .

d. Show that there exists a constant  $c > 1$  such that  $Q_{k_0} < 1/n^3$  for  $k_0 = c \lg n / \lg \lg n$ . Conclude that  $P_k < 1/n^2$  for  $k \geq k_0 = c \lg n / \lg \lg n$

e. Argue that

$$E[M] \leq \Pr \left\{ M > \frac{c \lg n}{\lg \lg n} \right\} \cdot n + \Pr \left\{ M \leq \frac{c \lg n}{\lg \lg n} \right\} \cdot \frac{c \lg n}{\lg \lg n}$$

Conclude that  $E[M] = O(\lg n / \lg \lg n)$

**Solution:**



- **Exercise 3. D-select**

In the deterministic linear-time selection algorithm, instead of partitioning all the inputs into a number of groups of size 5, we partition the inputs into a number of groups of size 7. The rest steps of the algorithm are similar to the original version. Will the runtime of this new algorithm still be  $O(n)$ , where  $n$  is the input size? Prove your claim.

**Solution:**

- **Exercise 4. Silly Blue Tiger**

1. Imagine that an algorithm requires us to hash strings containing English phrases. Knowing that strings are stored as sequences of characters, Blue Tiger decides to simply use the sum of those character values (modulo the size of its hash table) as the string's hash. Will the performance of the implementation match the expected value shown in lecture? (Yes/No and the brief reasoning are what we expect.)
2. Blue Tiger decides to implement both collision resolution and rehashing for its hash table. However, it does not want to do more work than necessary, so it wonders if it needs both to maintain the correctness and performance it expects. After all, if it has rehashing, it can resize to avoid collisions; and if it has collision resolution, collisions don't cause correctness issues. Which statement about these two properties true? (Choose one or two of the statements and briefly explain your choice.)
3. Blue Tiger wants to implement rehashing. Assuming it is enlarging a table of size  $m$  into a table of size  $m'$ , and the table contains  $n$  elements, what is the best time complexity to achieve this rehashing step (expanding a  $m$ -slot table with  $n$  elements into a  $m'$ -slot table)? Answer in big theta notation in terms of the variables in the context. Intermediate results of your complexity analysis should be displayed.
4. In lecture, we discussed approximately doubling the size of our hash table. Blue Tiger begins to implement this approach (that is, it lets  $m'$  be a prime close to but greater than  $2m$ ) but stops when it thinks that it might be able to avoid wasting half of the memory the table occupies on empty space by letting  $m'$  be a prime number close to  $(m + k)$  instead, where  $k$  is some constant. Does this work equally well comparing to picking a prime number close to  $2m$ ? Explain your answer.

**Solution:**

• **Exercise 5. Rehashing**

Given a sequence of inputs 4371, 1323, 6173, 4199, 4344, 9679, 1989, insert them into a hash table of size 10. Suppose that the hash function is  $h(x) = x \bmod 10$ . Hash table uses the double hashing with the second hash function as  $g(x) = (7 - x) \bmod 7$ . (Be careful with this mod calculation)

If any given input cannot be inserted correctly, do rehashing to guarantee a successful insertion, using a new table size as introduced in class (the first prime number after doubling the original hash table size) and a new hash function  $h(x) = x \bmod \text{new\_size}$ . Note that the order in rehashing depends on the order stored in the old hash table, not their previous inserting order.

Solve this question step by step and show intermediate results.

**Solution:**

The initial hash table:

	$T_0$	$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$T_6$
<i>entry0</i>							
<i>entry1</i>							
<i>entry2</i>							
<i>entry3</i>							
<i>entry4</i>							
<i>entry5</i>							
<i>entry6</i>							
<i>entry7</i>							
<i>entry8</i>							

Does the hash table need rehashing? If it does, when and why (answer in 1-2 sentences)? Also finish the rehashing process by drawing your own progress table that is similar to the one above:

- **Exercise 6**

Suppose we want to design a hash table containing at most 1162 elements using quadratic probing. We require it to guarantee successful insertions,  $S(L) < 2$  and  $U(L) < 4$ . Please determine a proper hash table size and show all intermediate steps.

- **Exercise 7**

In a few sentences (in an itemized fashion), explain why a hash table with open addressing and rehashing (doubling the size every time when the load factor exceeds a threshold) achieves insertion in  $O(1)$  time.

Answer the following questions: 1) When rehashing is not triggered, is the insertion time  $O(1)$  (does it have an upper bound)? Why? 2) Show that at any given point, the amortized (distributed to all items in the hash table) cost of rehashing is  $O(1)$  for each item in the hash table.

What if during rehashing, the hash table is expanded by a factor  $x$  with  $x > 2$ ? Is the amortized cost of rehashing still  $O(1)$ ? If no, explain. If yes, show your work. If it depends, show both.