CSE 100 Advanced Data Structures Fall 2018

Homework 3

Instructors: *Leo Porter (Sec. A)*
& *Debashis Sahoo (Sec. B)*

Due on: Thursday 11/08 *(40 points)*

Name: _____ PID: _____ *Date:* 11 /____ / 2018

Instructions

1. Answer each problem in the boxes or circles provided. Any writing outside of the boxes *will NOT be graded*. Do not turn in responses recorded on separate sheets.
2. Handwritten or typed responses are accepted. In either case, make sure all answers are in the appropriate boxes.
3. All responses *must* be neat and legible. Illegible answers will result in zero points.
4. Make sure to *scan in portrait mode* and to *select the corresponding pages* on Gradescope for each question.
5. You may use code from any of the class resources, including Stepik. You may not use code from other sources.

---

1. (8 points) *Linear Probing*: The following hash table was created using linear probing:

   Hash function: $h(x) = (5 * x)\%8$

0	1	2	3	4	5	6	7
11	3	2	7	12	23		19

   a. (3 points - **Correctness)** List all of the elements that could have been inserted into the hash table *first*

   2, 7, 12, 19

   b. (3 points - **Correctness)** List all of the elements that could have been inserted into the hash table *last*.

   2, 3, 23

   c. (2 points - **Completeness)** Provide a possible insertion order of the elements.

   2, 7, 12, 23, 19, 11, 3
   Make sure:
   That this ordering is present: …19,..,11,..,3,…
   That 2, 7 and 12 are inserted (in any order) before 23
   That there is no illegal first/last number

2. (6 points **- Correctness**) *Bloom Filter*: Given the following bloom filter:

- hash function 1: $h_1(x) = (x^3)\%13$
- hash function 2: $h_2(x) = (x * 5)\%13$

0	1	2	3	4	5	6	7	8	9	10	11	12
	1	1			1			1		1		1

a. What are 4 possible distinct values between 1 and 13 (inclusive) which could be inserted to create the bloom filter above?

inserted: 1,2,3,5. false positives: 8, 12
inserted: 2,3,5,8. false positives: 1, 12
inserted: 1,2,3,12. false positives: 5, 8
inserted: 2,3,8,12. false positives: 1, 5

b. List all the false positives for integers between 1 and 13 (inclusive) in your bloom filter given inserted elements you chose above.

inserted: 1,2,3,5. false positives: 8, 12
inserted: 2,3,5,8. false positives: 1, 12
inserted: 1,2,3,12. false positives: 5, 8
inserted: 2,3,8,12. false positives: 1, 5

3. (7 points **- Correctness**) *Cuckoo Hashing*: The following hash tables and their respective hash functions are used for cuckoo hashing.

- Hash function 1: $h_1(x) = (x\%7)\%5$

0	1	2	3	4
7			17	46

- Hash function 2: $h_2(x) = (x + 3)\%5$

0	1	2	3	4
47	33			

a. Draw the 2 tables listed above after inserting 10. If you think it leads to an infinite cycle write "infinite cycle".

[47, _, _, 17, 46] and [7, 33, _, 10, _]

b. Provide an insert value that will cause an infinite cycle. Show the cycle that is created.

Insert: 3
Cycle: 3 ->17->47->7->17-> 3 ->33->47->7->33-> 3

Other valid solutions:
12, 28, 38, 42, 52, 63, 68, 73, 77, 82, 87, 98

Make sure show the full cycle. A point is deducted if you only show 3->17->47->7->17->3 instead of 3->17->47->7->17->3->33->47->7->33->3

4. (6 points - **Correctness**) *Know Your Facts*:

a. What is the average-case time complexity for insert in a Hash Table.

● $O(1)$ ○ $O(\log n)$ ○ $O(n)$ ○ $O(n \log n)$ ○ $O(n^2)$

b. What is the average-case time complexity for find in a Hash Table.

● $O(1)$ ○ $O(\log n)$ ○ $O(n)$ ○ $O(n \log n)$ ○ $O(n^2)$

c. What is the worst-case time complexity for insertion in a Hash Table using Separate Chaining (assume a Linked List is used in the bucket implementation)

○ $O(1)$ ○ $O(\log n)$ ● $O(n)$ ○ $O(n \log n)$ ○ $O(n^2)$

(resize & rehash!)

d. What is the worst-case time complexity for find in a Hash Table using Separate Chaining (assume a Linked List is used in the bucket implementation)

○ $O(1)$ ○ $O(\log n)$ ● $O(n)$ ○ $O(n \log n)$ ○ $O(n^2)$

e. What is the worst-case time complexity for insertion in a Hash Table using a Hash Table using Separate Chaining (assume a BST is used in the bucket implementation)

○ $O(1)$ ○ $O(\log n)$ ● $O(n)$ ○ $O(n \log n)$ ○ $O(n^2)$

f. What is the worst-case time complexity for find in a Hash Table using Separate Chaining (assume a BST is used in the bucket implementation)

○ $O(1)$ ○ $O(\log n)$ ● $O(n)$ ○ $O(n \log n)$ ○ $O(n^2)$

g. What is the load factor of a Hash Table.

○ The minimum number of elements you need to insert before the Hash Table performs optimally

● The ratio #inserted elements/current size of hash table

○ It represents the maximum number of elements you can insert in your hash table

○ The ratio #inserted elements/#collisions

○ The ratio #occupied hash values/current size of hash table

h. At which load factor do you generally want to increase the size of your hash table?

- O 0.9
- O 0.7
- O 151

(we gave you half the points for .7 because this was mentioned in class according to some students https://en.wikipedia.org/wiki/Hash_table#/media/File:Hash_table_average_insertion_time.png)

- ● It depends on your collision resolving strategy
- O It depends on the current size of your hash table

i. What is the worst-case time complexity for insertion in a Hash Table using Separate Chaining (assume an AVL tree is used in the bucket implementation)

O $O(1)$ O $O(\log n)$ ● $O(n)$ O $O(n \log n)$ O $O(n^2)$

Resize & Rehash!

j. What is the worst-case time complexity for find in a Hash Table using Separate Chaining (assume an AVL tree used in the bucket implementation)

O $O(1)$ ● $O(\log n)$ O $O(n)$ O $O(n \log n)$ O $O(n^2)$

k. What is the worst-case time complexity for insertion using Cuckoo Hashing?

O $O(1)$ O $O(\log n)$ ● $O(n)$ O $O(n \log n)$ O $O(n^2)$

Infinite cycle => Resize & Rehash!

l. What is the worst-case time complexity for find using Cuckoo Hashing?

● $O(1)$ O $O(\log n)$ O $O(n)$ O $O(n \log n)$ O $O(n^2)$

5. (7 points **- Correctness**) *Load Factor*: You implemented a hash table using linear probing and the following hash function:

$$h(x) = (x + 4)\%5$$

You defined 0.7 to be the maximum load factor for your hash table and when exceeded you want to approximately double the size of your hash table.
This is the current state of your hash table:

0	1	2	3	4
6		13	24	

a. What is the current load factor of your hash table?

0.6

b. Show your hash function and hash table after inserting 5.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|24| | 5 | 6 | | | | | | | 13 |

These hash functions were all counted correct:
h(x) = (x+4) % 11
h(x) = (x+8) % 11
h(x) = (x+10) % 11
Note: you have to round up to the nearest prime (11) after doubling the size

c. What is the current load factor of your new hash table?

0.36

(if you used a table of length 10 for 5b, we counted 0.4 correct here)

6. (6 points **- Correctness**) *Collision Handling*: Consider two hash functions:
   - $h_1(x) = x\%7$
   - $h_2(x) = 5 - (x\%3)$

   Insert the following keys into a hash table of size 7:
   `13, 501, 128, 37, 2`

   a. Insert the keys using linear probing (using h_1)

0	
1	
2	128
3	37
4	501
5	2
6	13

   b. Insert the keys using separate chaining (using h_1)

0	
1	
2	128
3	
4	501
5	
6	13

-> 37 -> 2

c. Insert the keys using double hashing (use h_1 as primary and h_2 as secondary hash function)

0	
1	
2	128
3	37
4	501
5	2
6	13