CS6033 Design and Analysis of Algorithms I
Homework Assignment 6 Part 2
26.11.2022

Group Members:
rk4305 (Sai Rajeev Koppuravuri)
sg7372 (Sriharsha Gaddipati)
vt2182 (Venu Vardhan Reddy Tekula)
vt2184 (Veeravenkata Raghavendra Naveenkumar Tata)

---

1. (10 points) Consider a hash table with 9 slots and the hash function $h(k) = k \mod 9$. Demonstrate what happens upon inserting the keys 5, 28, 19, 15, 20, 33, 12, 17, 10 with collisions resolved by chaining.

   Ans:

   The given hash function is $h(k) = k \mod 9$.

   Keys to be inserted: 5, 28, 19, 15, 20, 33, 12, 17, 10

   We have to insert the keys by substituting the key in the function and placing the key with the value of the function as the index in the hash table.

   For example, for the key 5.
   $h(5) = 5 \mod 9 = 5$

   Insert the key 5 in the 5th index of the hash table.

   If any two keys have the same value from the function (index), this is called collision and given that the collision should be resolved by chaining. So, the keys should be stored as a list (linked list) in the hash table with the newest key as the first element.

   For example, the two keys 28 and 19 share the same functional value.

   $h(28) = 28 \mod 9 = 1$
   $h(28) = 19 \mod 9 = 1$

   So, the 28 will be inserted first into the list $[28]$

   And then 19 will be inserted next as the first element $[19, 28]$

   After inserting all the keys in the similar way, the hash table would look like the below

i	T[i]
0	[]
1	[10,19,28]
2	[20]
3	[12]
4	[]
5	[5]
6	[33,15]
7	[]
8	[17]

2. (10 points) Consider inserting the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table of length $m = 11$ using open addressing. Illustrate the result of inserting these keys using linear probing with $h(k, i) = (k + i) \mod m$ and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \mod (m - 1))$.

Ans:

Given,

Table size: m = 11

Keys to be inserted: k = 10, 22, 31, 4, 15, 28, 17, 88, 59

The standard hash function is
$h(k) = k \mod m$
$h(k) = k \mod 11$

Linear probing is used, so the hash function is
$h(k) = (k + i) \mod m$
$h(k) = (k + i) \mod 11$

where i = 1, 2, 3, ... , 10

Inserting the keys into the hash table

10: $10 \mod 11 = 10$, insert 10 at the index 10

22: $22 \mod 11 = 0$, insert 22 at the index 0

31: $31 \mod 11 = 9$, insert 31 at the index 9

4: $4 \mod 11 = 4$, insert 4 at the index 4

15: $15 \mod 11 = 4$, collision
$(15 + 1) \mod 11 = 5$, insert 15 at the index 5

28: $28 \mod 11 = 6$, insert 28 at the index 6

17: $17 \mod 11 = 6$, collision
$(17 + 1) \mod 11 = 7$, insert 17 at the index 7

88: $88 \mod 11 = 0$, collision
$(88 + 1) \mod 11 = 1$, insert 88 at the index 1

59: $59 \mod 11 = 4$, collision
$(59 + 1) \mod 11 = 5$, again collision
$(59 + 2) \mod 11 = 6$, again collision
$(59 + 3) \mod 11 = 7$, again collision
$(59 + 4) \mod 11 = 8$, insert 59 at the index 8

The hash table would look like

0	22
1	88
2	
3	
4	4
5	15
6	28
7	17
8	59
9	31
10	10

Double hashing is used

$h_1(k) = k$
$h_2(k) = 1 + (k \mod (m-1)) = 1 + (k \mod (11-1)) = 1 + (k \mod 10)$
$h(k) = (h_1(k) + i * h_2(k)) \mod m = (h_1(k) + i * h_2(k)) \mod 11$

Inserting the keys into the hash table

10: $10 \mod 11 = 10$, insert 10 at the index 10

22: $22 \mod 11 = 0$, insert 22 at the index 0

31: $31 \mod 11 = 9$, insert 31 at the index 9

4: $4 \mod 11 = 4$, insert 4 at the index 4

15: $15 \mod 11 = 4$, collision
$h_1(15) = 15; h_2(15) = 1 + (15 \mod 10) = 6$
$(15 + 1 * 6) \mod 11 = 21 \mod 11 = 10$, again collision
$(15 + 2 * 6) \mod 11 = 27 \mod 11 = 5$, insert 15 at the index 5

28: $28 \mod 11 = 6$, insert 28 at the index 6

17: $17 \mod 11 = 6$, collision
$h_1(17) = 17; h_2(17) = 1 + (17 \mod 10) = 8$
$(17 + 1 * 8) \mod 11 = 25 \mod 11 = 3$, insert 17 at the index 3

88: $88 \mod 11 = 0$, collision
$h_1(88) = 88; h_2(88) = 1 + (88 \mod 10) = 9$
$(88 + 1 * 9) \mod 11 = 97 \mod 11 = 10$, again collision
$(88 + 2 * 9) \mod 11 = 106 \mod 11 = 7$, insert 88 at the index 7

59: $59 \mod 11 = 4$, collision
$h_1(59) = 59; h_2(59) = 1 + (59 \mod 10) = 10$
$(59 + 1 * 10) \mod 11 = 69 \mod 11 = 3$, again collision
$(59 + 2 * 10) \mod 11 = 79 \mod 11 = 2$, insert 88 at the index 2

0	22
1	
2	59
3	17
4	4
5	15
6	28
7	88
8	
9	31
10	10

3. (5 points) What is a bound on the probability that you use at most 16n space for the secondary hash tables? (i.e. $\sum_{i=0}^{m-1}(n_j)^2 \leq 16n$).

   What is a bound on the probability (in terms of k) that you use at most kn space (for some constant k)?

   Ans:

   $P(\sum_{i=0}^{m-1}(n_j)^2 < 16n) + P(\sum_{i=0}^{m-1}(n_j)^2 \geq 16n) = 1$

   As we try to solve the above equation,

   $P(\sum_{i=0}^{m-1}(n_j)^2 \geq 16n) \leq \frac{E(\sum_{i=0}^{m-1}(n_j)^2)}{16n} < \frac{2n}{16n} < \frac{1}{8}$

   So, we can get

   $P(\sum_{i=0}^{m-1}(n_j)^2 \leq 16n) + P(\sum_{i=0}^{m-1}(n_j)^2 \geq 16n) = 1$

   $P(\sum_{i=0}^{m-1}(n_j)^2 \leq 16n) + \frac{1}{8} = 1$

   $P(\sum_{i=0}^{m-1}(n_j)^2 \leq 16n) = 1 - \frac{1}{8}$

   $P(\sum_{i=0}^{m-1}(n_j)^2 \leq 16n) \geq \frac{7}{8}$

   Similarly, for at-most kn space

   $P(\sum_{i=0}^{m-1}(n_j)^2 \leq kn) \geq 1 - \frac{2}{k}$

4. (10 points) Suggest how to implement a direct-address table in which the keys of stored elements do not need to be distinct and the elements can have satellite data. All three dictionary operations (INSERT, DELETE, and SEARCH) should run in O(1) time. (Don't forget that DELETE takes as an argument a pointer to an object to be deleted, not a key.)

   Ans: If the key does not have to be unique, we can use hashing via chaining to create a direct address table. There can be a double linked list node with satellite data against each duplicate key.

   Insert
   As new node will always be added to the head of the linked list, insert operations will always be $\mathcal{O}(1)$.

   Delete
   As the node's address will be provided, the delete operation will be $\mathcal{O}(1)$.

Search
As a result of the Load Factor (α) = # elements per slot / # slots in the table, the search operation will be $\mathcal{O}(1)$. This number will remain the same, hence the search for duplicate keys will continue to be $\mathcal{O}(1)$.

5. (15 points) Trying to finance your education, you decide to invest your D dollars in the stock market. The problem is you don't know which stocks to invest in. You decide ask your friends for stock tips and invest in every stock any of your friends suggests. If you receive n unique stock names you will invest D/n in each of the n stocks. Since some of your friends have given you the same stock name, you need to find a way to remove the duplicates.

   Design an efficient algorithm to perform this task. Your algorithm must run in O(n) average case time where n is the number of names given to you by your friends.

   Demonstrate that your algorithm runs in O(n) average case time.

   Ans:

   Here we can use a hash map to identify and remove duplicates from a list of given n stock names in O(n) time complexity

   Pesudo code

```
def remove_duplicates():
    stock_list = list(input()) # list of stock items

    hash_map = dict()

    for i in stock_list:
        if i not in hash_map:
            hash_map[i] = 1

    return hash_map.keys()
```

   Algorithm runs in O(n) in average case because we are iterating on the given list in a single loop and able to delete duplicates.