Solutions for Data Structures and Algorithms Spring 2023 — Problem Sets

By Dmitriy Okoneshnikov, B22-DSAI-04

February 06-07, 2023

Week 3. Problem set

1. Consider the following algorithm (see pseudocode conventions in [Cormen, Section 2.1]). The inputs to this algorithm are a map M and a key k. Additionally, assume the following:

   (a) The map M uses the same data type both for keys and for values.

   (b) The map M is not empty and contains n distinct keys.

   (c) The map M is represented as a hashtable with load factor α.

   (d) The map M resolves collisions via chaining [Cormen, Section 11.2].

   (e) The set of keys stored in M is equal to the set of values stored in M.

```
1   /* M is a map with a load factor α and size n,
2    * k is a key that may or may not be in M */
3   secret(M, k):
4       counter := 0
5       last_key := k
6       repeat
7           last_key := M.get(last_key)
8           counter := counter + 1
9       until last_key = k
10      return counter
```

   Compute the average case time complexity of secret. The answer **must** use Θ-notation. For the average case analysis, use *independent uniform hashing*. For full grade, you may assume the worst case for the arrangement of values stored in M.

   For extra credit, assume the average case also for the arrangement of values stored in M.

   Optionally, you may provide details for the computation of the running time $T(n)$. Proof for the asymptotic bound is not required for this exercise.

   Answer.

   The average case time complexity of secret: $\Theta(n(1+\alpha))$.

   The worst case for the arrangement of values stored in M is when hash table size is 1 and therefore, all the elements are stored in one linked list of size n. Then each time you would have to traverse through at worst n elements (time complexity $\Theta(n^2)$).

2. Consider a hash table with 13 slots and the hash function $h(k) = (k^2 + k + 3) \mod 13$. Show the state of the hash table after inserting the keys (in this order)

$$5, 28, 19, 15, 20, 33, 12, 17, 10, 13, 3, 33$$

with collisions resolved by linear probing [Cormen, Section 11.4].

Answer.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Key	10	33	3	12	13		19	5	20	28	15	33	17

3. In your own words, explain how it is possible to implement deletion of a key-value pair from a hashtable with $O(1)$ worst case time complexity if collision resolution is implemented using chaining?

   Answer.

   It is possible to remove a key-value pair from a hashtable with separate chaining collision resolving with $O(1)$ worst case time complexity by using doubly linked lists and passing to the REMOVE method a pointer to the node containing the key-value pair.

   This way we can, without traversing through the list, directly access (with time complexity of $O(1)$) the node we want to remove. Let us assume this node has index of i. Then we just need to assign the $i-1$ node's next element pointer to $i+1$ node and the $i+1$ node's previous element pointer to $i-1$ node.

   Therefore, the time complexity of this removal is always $O(1)$.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms, Fourth Edition.* The MIT Press 2022