DSA Week 3 Problem Set's solution

by Elina Kuzmichyova

Week 3. Problem set (solutions)

1. Consider a hash table with 15 slots and the hash function $h(k) = \big(k^2 - 2k + 7\big)$ mod 13 . Show the state of the hash table after inserting the keys (in this order)

$$5, 28, 19, 15, 20, 33, 12, 17, 10, 13, 3, 34$$

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

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Key	3		17	20	33	19	34	28	15	5	12	10	13

2. 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 positive integers 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 double hashing [Cormen, Section 11.4].
(e) For all keys k, if a value in M at k exists, then it is smaller than k.

```
secret(M, k):
    total := 0
    v := M.get(k)
    for i=1 to v
        if    M.hasKey(i)
            total    :=    total    +    M.get(i)
    return    total
```

Compute the average case time complexity of secret. The answer must use O-notation and may depend on n, k, and α. For the average case analysis, use independent uniform permutation hashing. Assume worst case for the contents of the input map M.

Briefly justify your answer ($2-3$ sentences). Detailed proof for the asymptotic complexity is not required for this exercise.

Line	Time	Cost	Total cost
3	1	C_1	C_1
4	1	C_2	C_2
5	1	$\frac{1}{\alpha}\log\frac{1}{1-\alpha}$	$\frac{1}{\alpha}\log\frac{1}{1-\alpha}$
6	k	C_4	C_4*k
7	k-1	$\frac{1}{1-\alpha}$	$\frac{1}{1-\alpha}$*(k-1)
8	0	$\frac{1}{\alpha}\log\frac{1}{1-\alpha}$	0
9	1	C_7	C_7

So when sum all total costs: $C_1 + C_2 + C_4$*k + $C_7 + \frac{1}{\alpha}\log\frac{1}{1-\alpha} + 0 + \frac{1}{1-\alpha}$*(k-1) as we count big O notation, we don't count constants, so we will have $\frac{1}{\alpha}\log\frac{1}{1-\alpha}$ + $\frac{1}{1-\alpha}$*(k-1) as first expression won't grow up as second one will be, we count only second, so the answer will be O($\frac{1}{1-\alpha}$*(k-1))

P.S.: We have the worst case map and we use the thing that trying to find and not finding is the most time consuming variant with map M. (as hasKey works longer in unsuccessful case)

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? Specify the data structures and methods involved.

Methods: Deletion
Data structures: Double linked list, hashtable
Description of deletion methods: Let assume that we use double linked list for collision resolution. To delete key-value pair we need to get a link or a pointer to this pair in double linked list. After we have an access we should rewrite head and tail for pair's neighborhoods, tail for previous element will be equal to tail of our pair and head of the next element will be equal to head of our pair. This two operation takes constant time, so it means $O(1)$ worst case time complexity.