Assignment 4, 6.006, Introduction to Algorithms

Bryan Zhang

June 19, 2018

1 Problem 4-1. Hash Functions and Load

1.1 a

Answer: 3
Hashing is to distribute every element equally randomly into a sub-slot.

1.2 b

Answer: 4 Both are necessary because without collision resolution resize the hash-table can not preclude two elements with the same hash values from colliding into the same slot. While only chaining but without resizing the hash table, the load factor will bigger and bigger. Thus the running time is bigger and bigger, which drags the performance slow.

1.3 c

Answer: 6 correct answer: 7
Since m' > m, then the time complexity is $O(n + m')$

Algorithm 1 Risize the Hash-table

```
function RESIZE($oldTable, m'$)
    allocate new memory of size m'                                  ▷ $O(1)$
    for slots in the newTabale do slots = NIL                        ▷ $O(m')$
    for slots in the oldTable do make new slot in the newTable
        for item in the old slots do copy(item)                     ▷ $O(1)$
                                                                    ▷ $O(n + m)$
                                                                    ▷ $O(n + m + m')$
     =0
```

1.4 d

Answer: Theoretically the time complexity is $O(n)$. Since for n insertions, the cost is $O(k + k + k.... + k)$ while the the number of the terms is n/k. So the total cost is $O(n^2)$, thus $O(n)$ amortized.
Practically, the memory system is binary. It is more convenient to allocate memory blocks doubly.

2 Problem 4-2. Python Dictionary

2.1 a

Answer: 1 The ”Member Testing” use cases states that the dictionary rarely changes not can’t change.

2.2 b

Answer: 1 **correct answer: 3**

We need to first insert so many items that we need a large minimum size. And Since we need quick performance for dictionary member testing, a sparse hash table is preferred. Thus the growth factor is 4.

3 Problem 4-3. Matching DNA Sequences