Week 5 feedback and quiz answers

(read only after passing the programming assignment and the quiz)

Keep up the great work!

Congratulations on completing Week 5 of Data Structures: Measuring and Optimizing Performance! To complete this quiz it requires having an understanding of Hash Tables and how to deal with collisions! If you struggled with Week 5's quiz, that's okay because there are no more quizzes left in this course! I hope you feel accomplished.

So what now? Well first off, be proud of what you have achieved because this course was difficult and was designed to get you to think in new and creative ways. Secondly, feel free to go back to previous weeks and further explore topics. Thirdly, join us for the next course in this specialization as we learn about graphs! Lastly, refer a friend to this course and have them embark on the same exciting journey you did.

The rest of this reading contains the answers, as well as some explanations, for this week's quiz questions. Correct answers to the quiz questions are shown in **bold**. Incorrect answers are shown in *italics*.

Good luck and once again congratulations!

It has been a pleasure working with you and hopefully see you in the next course!

Sincerely,

Our MOOC Team

Quiz Answers and Explanations

Correct answers are in **bold**. Incorrect answers are shown in *italics*.

1. Which of the following is true of searching for an element in a Hash Table that is implemented using linear probing (you may assume a Hash Table that is ~70% full)?

- *Worst case is O(1) (Incorrect)*

With linear probing, a long consecutive list of values could be created whose length is equal to the # of elements in the hash table. Hence, the worst case would require looking at O(n) elements.

- *Average case is log(n) (Incorrect)*

Average case for a hash table is actually better than log n.

- **Average case is O(1) (Correct)**

Great!

2. When a hash table becomes too full, which of the following is required to resize the table?

- *Create a new table which is larger than the original table. Then bulk copy keys from one table to the other (i.e. if a key is in location X in the original table, it remains in position X in the new table). (Incorrect)*

When you change the size of the hash table, you need to change the hash function. So elements will need to be reinserted based on the new table's hash function.

- **Create a new table which is larger than the original table. Then individually reinsert each key from the old table into the new table (i.e. if a key is in location X in the original table, it may or may not be in position X in the new table). (Correct)**

Great!

- *Add extra space at the end of the existing hash table and leave the old keys in place. (Incorrect)*

When you change the size of the hash table, you need to change the hash function. So elements will need to be reinserted based on the new table's hash function. Also, adding extra space at the end of a contiguous data structure (like an array) is not always feasible.

3. In the algorithm for spelling suggestions in this week's assignment, there was a hash set to mark Strings as visited. What is the BEST ANSWER for why we used the visited set?

- **To avoid exploring all the possible String mutations for an individual String more than once. (Correct)**

Great! This is the best answer. For any String of length n, there are (26*(n+1))+(26*n)+n possible mutations. Exploring the same mutations of the same String more than once can significantly hurt run-time.

- *To avoid testing to see if a mutated String was a real word (and hence a spelling suggestion) more than once. (Incorrect)*

This is close, it did help us avoid testing to see if the same String was a dictionary word more than once, but the average case lookup of the hashset is just O(1), so that's not the primary reason.

- *So that our exploration was of a tree of words, and if we had a word repeated, it wouldn't be a tree anymore. (Incorrect)*

Trees can have duplicates and there's a better reason why we had the visited hash set.

4. Which of these is an invalid word path (assume all words below are valid dictionary words)?

- *From broom to mop: broom -> boom -> boo -> moo -> mop (Incorrect)*

This is a valid path. Each mutation is just one character.

- *From cake to stake: cake -> sake -> stake (Incorrect)*

This is a valid path. Each mutation is just one character.

- **From shy to chime: shy -> she -> he -> hide -> chide -> chime (Correct)**

Great! This is invalid. You cannot go from "he" to "hide" with one mutation.