Week 1 Quiz

TOTAL POINTS 10

1. According to video lesson 1.1.1, a hash table consists of three things. Which of these was NOT one of those three things? 1 point

   - Encryption
   - An array
   - Collision handling
   - A hash function

2. Given a hash function h(key) that produces an index into an array of size N, and given two different key values key1 and key2, the Simple Uniform Hashing Assumption states which of the following? 1 point

   - The probability that h(key1) == h(key2) is 0.
   - If h(key1) == h(key2) then h needs a running time of O(N) to complete.
   - If h(key1) == h(key2) then h needs a running time of O(lg N) to complete.
   - The probability that h(key1) == h(key2) is 1/N.

3. According to video lesson 1.1.2, which of the following is a *good* hash function h(key) that translates any 32-bit unsigned integer key into an index into an 8 element array? 1 point

   (Note that an expression like "2 & 3" uses the bitwise-AND operator, which gives the result of comparing every bit in the two operands using the concept of "AND" from Boolean logic; for example, in Boolean logic with binary numbers, 10 AND 11 gives 10: for the first digit, 1 AND 1 yields 1, while for the second digit, 0 AND 1 yields 0. An expression like "4 % 8" uses the remainder operator that give the remainder from integer division; for example, 4 % 8 yields 4, which is the remainder of 4/8. In some cases, these two different operators give similar results. Think about why that is.)

   ```
   int h(uint key) {
       int index = 5;
       while (key--)
           index = (index + 5) % 8
       return index;
   }
   ```

   ```
   int h(uint key) { return key & 7; }
   ```

   ```
   int h(uint key) { return rand() % 8; }
   ```

   ```
   int h(uint key) { return max(key,7); }
   ```

4. Suppose you have a good hash function h(key) that returns an index into an array of size N. If you store values in a linked list in the array to manage collisions, and you have already stored n values, then what is the expected run time to store a new value into the hash table? 1 point

- O(n)
- O(N)
- O(n/N)
- O(1)

5. Suppose you have a good hash function h(key) that returns an index into an array of size N. If you store values in a linked list in the array to manage collisions, and you have already stored n values, then what is the expected run time to find the value in the hash table corresponding to a given key? 1 point

- O(n/N)
- O(N)
- O(n)
- O(1)

6. Which one of the following four hashing operations would run faster than the others? 1 point

- Finding a value in a hash table of 4 values stored in an array of 8 elements.
- Finding a value in a hash table of 20 values stored in an array of 100 elements.
- Finding a value in a hash table of 100 values stored in an array of 1,000 elements.
- Finding a value in a hash table of 2 values stored in an array of 2 elements.

7. When storing a new value in a hash table, linear probing handles collisions by finding the next unfilled array element. Which of the following is the main drawback of linear probing? 1 point

- There may not be an available slot in the array.
- If the hash function returns an index near the end of the array, there might not be an available slot before the end of the array is reached.
- Even using a good hash function, contiguous portions of the array will become filled, causing a lot of additional probing in search of the next available unused element in the array.
- The array only stores values, so when retrieving the value corresponding to a key, there is no way to know if the value at h(key) is the proper value, or if it is one of the values at a subsequent array location.

8. When using double hashing to store a value in a hash table, if the hash function returns an array location that already stores a previous value, then a new array location is found as the hash function of the current array location. Why? 1 point

- Only one additional hash function is called to find an available slot in the array whereas linear probing requires an unknown number of array checks to find an available slot.
- Double hashing reduces the chance of a hash function collision on subsequent additions to the hash table.
- Double hashing reduces the clumping that can occur with linear probing.
- Since the hash function runs in constant time, double hashing runs in O(1) time.

9. Which of the following data structures would be the better choice to implement a memory cache, where a block of global memory (indicated by higher order bits of the memory address) are mapped to the location of a block of faster local memory. 1 point

- An AVL tree.
- A hash table implemented with double hashing

A hash table implemented with separate chaining, using an array of linked lists.

A hash table implemented with linear probing.

10. Which of the following data structures would be the better choice to implement a dictionary that not only returns the definition of a word but also returns the next word following that word (in lexical order) in the dictionary. 1 point

A hash table implemented with double hashing.

An AVL tree.

A hash table implemented with separate chaining, using an array of linked lists.

A hash table implemented with linear probing.

I, **Jiarong Yang**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit