CS 251 Homework 2 (100 points)

Out: February 1st, 2019 (8:00 pm)
Due: February 8th, 2019 (8:00 pm)

Important: Each question has only one correct answer. Additionally, you must provide an **explanation** for your answer (1 sentence for multiple-choice questions and 2,3 sentences for short answers). Answers without proper explanations, even though it is correct, will not get full credit.

Select the correct option for the following multiple choice questions or provide short answers accordingly:

1. Consider a hash table with chaining, of size N. Assume in each of the following scenarios the table starts empty. Further assume K pairs of <key, element> are added and K = N. Then, in which of the following situations will a find operation for a particular key have a potential worst-case runtime of O(N).
   a) The keys of all elements are powers of two and the hash function returns a unique index for each key mod N.
   b) The hash function returns a fixed number mod N, but the keys of all elements inserted are unique.
   c) Although the hashing function computes a good distribution of key values into the hash table indices, the hash function returns only even numbers.
   d) The find operation in this problem is never near O(N).

   B
   Explanation: When the hash function returns a fixed number with unique integers, after the mapping by hash function, the searching still need to go over a list with length N, which is O(N) in our case.

2. You have a hash table of size N = 65. You have total 10,000 elements to insert into the table (e.g., N << 10,000). The keys are all unique and are all 64 bit positive integers. You have to find a suitable hash function h(k) for key 'k'. Which of the following hash functions will NOT work with the given parameters?
   a) h(k) = k mod N
   b) h(k) = (number of bits of k which are '1') mod N
   c) $h(k) = \lfloor \log_2(k) \rfloor$
   d) $h(k) = \lfloor \log_2(k^2) \rfloor$

   D

Explanation: For function in d), since there can be integer up to 2^{64}, h(k) can up to 128. However, the size is only N = 65, where it cannot satisfy the range of this specific hash function.

3. Consider the same setting as above. The only change is, now the keys are all ASCII strings of length 32, instead of integers. Which of the following hash function will NOT work?
   a) h(k) = (sum of the ASCII values of all the characters) mod N
   b) h(k) = (Σ_i k[i] * 128^i) mod N , where $0 \le i \le 31$, and k[i] represent the i^{th} character in string k.
   c) h(k) = (ASCII value of k[i]) / 4 , where 'i' is randomly chosen, $0 \le i \le 31$, k[i] represent the i^{th} character in string k, and ASCII values are in the range [0, 255].
   d) All the hash functions defined above will not work (in this problem).
   e) All the hash functions defined above will work (in this problem).

   C
   Explanation: For method b), the same string might have different hash value. In this case, it would be impossible to perform searching.

4. You are given a hash table which allows collision. Each slot of the hash table actually keeps a linked list, and whenever a collision occurs the new element is added to the tail of the list. Suppose, you have total M elements, and the hash table is of size N. Let's assume the hash function generates hash values i=h(k) which look almost random. What is the expected running time (not the worst-case running time) for a search operation?
   a) O(log M)
   b) O(log N)
   c) O(M/N)
   d) O(1)

   C
   Explanation: We expect the elements uniformly distributed on the hash table. Therefore, there should be M/N elements for every value in hash table to perform searching.

5. Explain in one sentence why each of the following is a property of a good hash function:
   a) Fast to compute
   For every search, the hash function need to be called. A hash function with low cost of computation can help to reduce the operation time.
   b) Rarely causes collisions
   When the hash function rarely causes collisions, optimally, the searching can be performed with O(1). Otherwise, the complexity of searching can reach O(n) as the worst case.
   c) The same keys hash to the same value

If the same keys can hash to different values, it will be hard for the algorithm to perform insertion. This is also confusing to perform the searching.

6. Assume that you have an array of Strings representing words from the English language. The length of the array is N. You are tasked with finding the word with the maximum and minimum frequency. Describe an algorithm where this can be done in O(N) using a hash table.

   Suppose the hash table is HT, the hash function is H(x), the word in array is w.
   We loop the word array, update HT with HT[H(w)]++ to get the count of each words.
   O(N).
   We loop the word array, compare for the largest element count.
   O(N)
   We loop the word array, compare for the lowest element count.
   O(N)
   Therefore, in total, this algorithm can be done in O(N).

7. Assume that you have an array of unique integers. The length of the array is N. You are tasked with finding two integers in this array such that their sum is equal to a provided target integer. Describe an algorithm where this can be done in O(N) using a hash table. You can assume that exactly one pair of elements in the array sums to the provided target integer.

   Suppose the Hash function is represented by H(x). The target value is T. The values are initialized by 0.
   For an input number n, we first check if H(n) is 0 or 1.
   If H(n) = 0, we set H(T - n) = 1.
   If H(n) != 0, we find the pair we want.

8. The maximum number of nodes that can be present in a binary tree with depth n (Assume root is at depth 0) are:
   NOTE: You need to provide the proof of your result (i.e., you need to derive the result).

   When n = 1, the number of nodes is 1.
   When n = 2, the number of nodes is 3.
   When n = 3, the number of nodes is 7.
   Suppose when n = k, the number of nodes is 2^k-1.
   For n = k + 1:
   Since every previous leaf can create two nodes as new leafs, the added nodes are 2^k.
   The number of nodes will be 2^{k+1}-1.
   Therefore, the number of nodes is 2^n-1.

9. You have N elements equally distributed in K lists sorted in ascending order. Further the K lists are initially sorted in ascending order as well (based on the minimum element present at the start of each list). However, there is no guarantee of the relative values of the elements in the K lists below the current top of each list. Assuming no extra data structure is built:
   a) What is the tightest big-oh notation for removing the smallest element?
   O(1). Since the K lists are sorted and lists itself are sorted, the smallest element is located in list0 with index of 0. To remove this item, O(1) is the complexity.
   b) What is the tightest big-oh notation for re-sorting the K lists once the smallest element of one of the lists is removed?
   O(K). Since there are K lists in total, when removing one of the smallest element, the K lists need to be re-sorting. Since the K lists are already sorted, it is the same to consider an insertion to a sorted list. Therefore, O(K) is the complexity.
   c) What is tightest big-oh notation for removing all N elements?
   $\boldsymbol{O(NKlog(\frac{N}{K}))}$***. For removing one element, we need search K lists, as a sorted list, we need*** $\boldsymbol{log\left(\frac{N}{K}\right)}$ ***for searching. For searching K lists, we need*** $\boldsymbol{Klog\left(\frac{N}{K}\right)}$***. For removing N elements, the complexity should be*** $\boldsymbol{O(NKlog\left(\frac{N}{K}\right)}$***. We assume the size is*** $\boldsymbol{\frac{N}{K}}$ ***since it is uniformly distributed.***

10. How many comparisons are needed by insertion sort to sort an array of 100 integers in the best possible case.
    NOTE: Provide appropriate justification on when you might be able to see the best case.
    a) 100
    b) 4950
    [✓ c) 99]
    d) 98

    C
    Explanation: When inserting the first integer, we do not need to perform any comparison. In the best case, the array to be sorted is already with the order we want. We will only need to perform 100 – 1 = 99 times of searching.

11. You run BubbleSort on a list of 10,000 items that are listed in reverse order. How many swaps will happen when running this program?
    [✓ a) 49,995,000]
    b) 50,000,000
    c) 50,005,000
    d) 100,000,000

    A

Explanation: For the worst case of bubble sort, if the array length is n, it need to swap for (n-1) + (n-2) + … + 2 + 1 times. When n = 100, it results in 49995000.

12. Suppose that you have a pre-sorted linked list of size N, and you want to insert a new item into the structure in a way that keeps the list in order. What is the runtime of this operation?
    a) O(logN)
    b) O(N)
    c) O(1)
    d) O(NlogN)
    B
    Explanation: Since the list is already sorted, we only need to go over the list for once to insert into correct position, which the complexity is O(N).