CS251 - Homework 2: Hashing/sorting

Out: February 05, 2016 @ 9:00 pm
Due: February 12, 2016 @ 9:00 pm

Important: Each question has only one correct answer. Additionally, you must provide an explanation on each question. Any choice without an explanation, even though it is correct, will be graded with 0 points.

1. You have a hash table of size N (N = 65). You have total 10,000 elements (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 you NOT choose?

   a) h(k) = k mod N
   [✓ b) h(k) = (number of bits which are '1') mod N]
   c) h(k) = $\lfloor \log_2 (k) \rfloor$
   d) h(k) = $\lfloor \log_2 (k^2) \rfloor$

   Different binary numbers can have the same number of 1's (i.e. 1 [001] and 2 [010])

2. Consider the same setting as question 1. The only change is, now the keys are all (non-extended) ASCII strings of length 32, instead of integers. Which of the following hash function you will NOT choose?

   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]) /2 , where 'i' is randomly chosen, $0 \le i \le 31$. And k[i] represent the i th character in string k.
   [✓ d) None of the above are valid.]
   e) All of the above are valid.

   A is not valid because 'ab' and 'ba' have the same hash. C is not valid because you might not pick the same letter each time

3. A hash table uses open addressing with quadratic probing. Which of the following scenarios leads to linear running time to search for a random key?

   [✓ a) All keys hash to same index]
   b) All keys hash to different indices
   c) All keys hash to different even-numbered indices
   d) None of the above

   This ensures that each index is looked at once.

4. A hash table of length 12 uses open addressing with hash function h(k)=k mod 12, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below:

					17	18	67	6	41	22	
0	1	2	3	4	5	6	7	8	9	10	11

What can be a possible order of insertion in the table?
[✓ a) 18, 17, 22, 67, 6, 41]
b) 17, 6, 18, 22, 41, 67
c) 17, 18, 41, 67, 6, 22
d) 22, 41, 18, 17, 67, 6

18%12 =6, 17%12 = 5, 22%12 = 10
67%12 =7, 6%12 = 6, 41%12 = 5
6 and 5 were already full, so 6 and 41 were moved to the next open space

5. You are given a hash table which allows collision. So, each slot of the hash table actually keeps the head and tail pointer of a linked list, and whenever a collision occurs the new element is added to the tail of the list. Suppose, you have a hash table of size N containing M elements (N < M). Assume the hash function generates hash values i=h(k), which looks almost random. What is the average (or expected) running time (i.e., not the worst case running time) for an insert operation?
a) O(log M)
b) O(log N)
c) O(N)
[✓ d) O(1)]

Hasing the key and inserting at the end of the linked list are both constant time operations.

6. In the same setting as question 5, which of the following options is closest to the average number of linked list nodes that will need to be traversed during a search operation?
a) log M
b) log N
[✓ c) M/N]
d) 1

The average would be the length of the list divided by the number of lists thus the average time would be M/N.

7. You perform a sort on a list of 1000 items using an optimal ($O(n*\log_2 n)$) algorithm. You time the process and note that it took 20s to run. Some time later you run the sort again on a new list. This time, the process takes 96s to complete. About how many elements did you sort?
   a. 3000
   b. 4000
   c. [✓ 5000]
   d. 10,000

   The second run takes ~4.8 times longer, so 5000 items were sorted.

8. Imagine that you have a flexible implementation of a heap data structure. You want to sort a list of (key, data) pairs by putting the list into the heap and then printing it out in decreasing sorted key order. The pseudocode for this is below. Could the below pseudocode work and what is the runtime of this function?

```
List mysort(List L)
{
     Heap heap;
     for every element e in L
          put e in heap;

     let N be length of L;
     List sL;
     for N times
          remove largest key and put in sL;

     return sL;
}
```

a) This can work, and runs in $O(n^2)$.
b) [✓ This can work, and runs in $O(n*\log(n))$]
c) This can not work, but runs in $O(n^2)$.
d) This can not work, but runs in $O(n*\log(n))$

The top loop is O(N) and the bottom loop is O(log(N))

9. You run BubbleSort on a list of 10,000 items that are listed in reverse order. Exactly how many swaps (not comparisons) will happen when running this program?
    a. 49,995,000
    b. 50,000,000
    c. 50,005,000
    [✓ d. 100,000,000]

bubble sort is O(n^2) so worst case is 100 million

10. Now consider a Selection Sort (as could be implemented with a sequence based priority queue) with the same list of 10,000 items from problem 9. Exactly how many swaps (not comparisons) will happen when running this program?
    a. 10,001
    b. 10,000
    c. 4999
    [✓ d. 5000]

The number of swaps that are done is at most n/2 because each swap should get the list closer to being sorted.

11. Generalize the result from question 10. Given a reversed list of N objects, exactly how many swaps will selection sort do on the list?
    a. N + 1
    b. N
    [✓ c. ceiling(N / 2)]
    d. floor(N / 2)

On sorting an odd number of items you may have to swap the last item.

12. Suppose that you have a pre-sorted singly linked list and no other auxiliary data structure. 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(log(n))
    [✓ b. O(n)]
    c. O(1)
    d. O(n*log(n))

Worst case, you have to make N comparisons and insert the item at the end of the list.

Submit Instructions:
The homework must be turned in by the due date and time using the turnin command. Follow the next steps:

1. Please make sure your submission is legible! (No cellphone pictures please! All ITAP labs have scanners for you to use)
2. Login to data.cs.purdue.edu (you can use the labs or a ssh remote connection).
3. Make a directory named with your username and copy your solution (**in pdf format**) there.
4. Go to the upper level directory and execute the following command:
   turnin -c cs251 -p hw2 your_username
   (**Important:** previous submissions are overwritten with the new ones. Your last submission will be the official and therefore graded).
5. Verify what you have turned in by typing **turnin -v -c cs251 -p hw2**
   (**Important:** Do not forget the **-v** flag, otherwise your submission would be replaced with an empty one). If you submit the wrong file you will not receive credit.