Dylan Mackey

## 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.

 $^{7}$  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

The beginning of the bash table would be extendly full as a low number

b) (k) = (number of bits which are '1') mod N of '1''s would be common

 $c) h(k) = L log_2(k) \rfloor$ 

d)  $h(k) = L \log_2(k^2) J$ 

- 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)  $h(k) = (\sum_i k[i] * 128^i) \mod N$ , where  $0 \le i \le 31$ , and k[i] represent the  $i^{th}$ character in string k.

ć),h(k) = ( ASCII value of k[i] ) /2 , where 'i' is randomly chosen, 0 ≤ i ≤ 31. And

k[i] represent the ith character in string k.

If" is randomly chosen pach

d) None of the above are valid. e) All of the above are valid.

time, then you will not know where the item is hashed to

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

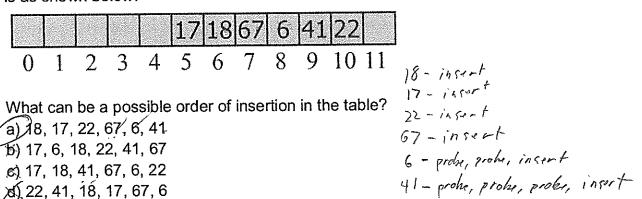
a) All keys hash to same index
b) All keys hash to different indices
c) All keys hash to different even-numbered indices

the samed index if will
c) All keys hash to different even-numbered indices

take at most in prohis to

Essuming the quadratic probin Circles hack around to the

10 124 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:



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)

worst case. Inserting to a hash table

c) O(N)

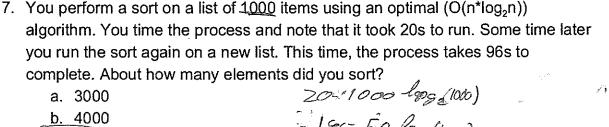
with linked lists is constant time.

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 If every hash was even, then each hash slot will have in elements, or nodus, that need to be traverced



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?

Assuming by returning st List daysort (List da) of **€6** we print the items os well, Heap cheap; co this can worke. Each element for divery delement ded not of will be inserted, and then put ce can cheap; co up-heapool. Then, the largest levy will be removed and added to 54. To 18 sort, it <del>رو ۲</del> let where ength wif wi; wi ListeL; 6 for distinctions of removed argest developed the rest; of takes alog(n), as up/ ょう as to remove fing returnedL; el So. 2nlog(n) or Olaton } 6

- a) This can work, and runs in O(n2).
- b) This can work, and runs in O(n\*log(n))
- c) This can not work, but runs in O(n²).
- d) This can not work, but runs in O(n\*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

 $\frac{n^2}{2} - \frac{n}{2}$ . In a worst case sciences a buble sort will perform  $\frac{n^2}{2} - \frac{n}{2}$  sugps. In this case, 49, 995, 000 when n 

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

There are 2 swaps in 654 3

this worst-case, 1514326-

123456-

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

ellowert will not 1 4 3 2 5

c. ceiling(N/2)

red to be supposed, 1 2 3 4 5

el. floor(N/2)

d. floor(N/2)

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))

Inserting is O(1), however finding the

node to insert is O(1) therefore

0 (n)