CS 61B Spring 2019

LLRBs, Hashing, Heaps

Exam Prep 8: March 4, 2019

1 Hashtable Runtimes (Fall 2016 MT2: Q3)

Consider a hash table that uses external chaining and also keeps track of the number of keys that it contains. It stores each key at most once; adding a key a second time has no effect. It takes the steps necessary to ensure that the number of keys is always less than or equal to twice the number of buckets (i.e., that the load factor is ≤ 2). Assume that its hash function and comparison of keys take constant time. All bounds should be a function of N, the number of elements in the table.

1. Give $\Theta()$ bounds on the worst-case times of adding an element to the table when the load factor is 1 and when it is exactly 2 before the addition.

   Bound for load factor 1:

   Bound for load factor 2:

2. Assume that the hashing function is so good that it always evenly distributes keys among buckets. What now are the bounds on the worst-case time of adding an element?

   Bound for load factor 1:

   Bound for load factor 2:

3. Making no assumption about the goodness of the hashing function, suppose that instead of using linked lists for the buckets, we use some kind of binary search tree that somehow keeps itself "bushy." What bound can you place on the worst-case time for testing to see if an item is in the table?

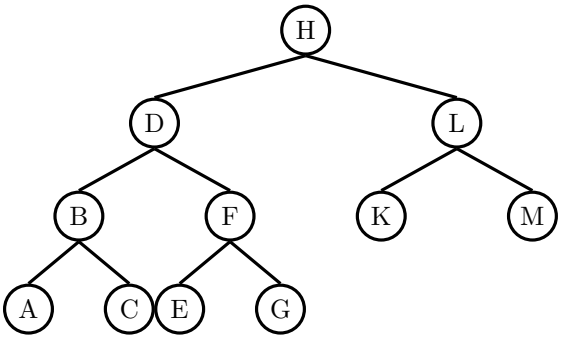
   Bound:

4. Using the same representation as in part (c), but with a very good hash function, as in part (b), what bound can you place on the worst-case time for testing to see if an item is in the table?

   Bound:

2 Min Heaps (Spring 2018 MT2 Q6)

Consider the min heap below, where each letter represents some value in the tree. For each question, indicate which letter(s) correspond to the specified value. Assume each value in the tree is unique.



Assuming values are inserted into the heap in **increasing** order, indicate all letters which could represent the following values:

Smallest value:

Median value:

Largest value:

Assuming values are inserted into the heap in **decreasing** order, indicate all letters which could reprsent the following value:

Smallest value:

Median value:

Largest value:

Assuming values are inserted into the heap in an **unknown** order, indicate all letters which could represent the following values:

Smallest value:

Median value:

Largest value: