Ve281 Data Structures and Algorithms
Written Assignment Three

This assignment is announced on Oct. 17th, 2017. It is due by 5:40 pm on Oct. 27th, 2017. The assignment consists of eight problems.

1. Define a family H of hash functions mapping keys from a universe U to the set $\{0, 1, \ldots, n-1\}$ to be **ϵ-universal** if for all pairs of distinct keys $k, l \in U$,

   $$Pr(h(k) = h(l)) \leq \epsilon,$$

   where the probability is over the random choice of the hash function h from the family H. Show that an ϵ-universal family of hash functions must have

   $$\epsilon > \frac{1}{n} - \frac{1}{|U|}.$$

2. In lecture, we show a few examples of universal family of hash functions which satisfy that for all pairs of distinct keys $k, l \in U$,

   $$Pr(h(k) = h(l)) = \frac{1}{n},$$

   where n is the size of the hash table and the probability is over the random choice of the hash function h from the family H. Does there exist any example with $|U| > n$ so that for all pairs of distinct keys $k, l \in U$,

   $$Pr(h(k) = h(l)) < \frac{1}{n}?$$

   (Hint: consider this problem together with the claim in Problem 1.)

3. Suppose we want to design a hash table containing at most 600 elements using linear probing. We require that an unsuccessful search needs no more than 8.5 compares and a successful search needs no more than 3 compares on average. Please determine a proper hash table size.

4. A full node in a binary tree is a node with two children. Prove that the number of full nodes plus one is equal to the number of leaves in a nonempty binary tree.

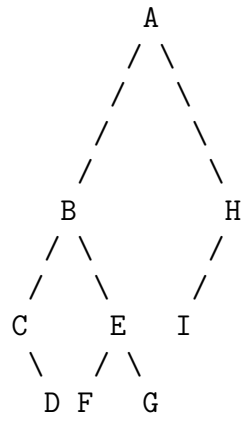
5. For the following tree, show the order in which the nodes are visited during the following tree traversals (The nodes in the tree are from A to I):

   (a) Pre-order depth-first traversal.

   (b) Post-order depth-first traversal.

   (c) In-order depth-first traversal.

   (d) Level-order traversal.



6. In class, we showed a recursive way to realize in-order depth-first traversal of a binary tree. In this problem, we ask you to design a **nonrecursive** algorithm that performs in-order depth-first traversal. Assume the tree is stored using a linked structure with node as

```
struct node {
  int key;
  node *left, *right;
}
```

   You can either describe your algorithm in plain English or write pseudo-code. If you choose to write pseudo-code, you should write in a way that can be easily understood. Otherwise, you will get a zero for the problem. (Hint: consider using a stack as an auxiliary data structure.)

7. Min Heap

   (a) Suppose that we are inserting the keys 3, 1, 4, 1, 5, 9, 2, 6, 5, 4 **one by one** into an initially empty min heap. Show the resulting min heap in the form of a tree.

   (b) For the min heap you obtained for Problem (7a), show the resulting heap after calling two `dequeueMin` operations.

8. Min heap initialization

   Given a sequence of keys 3, 9, 7, 2, 5, 2, 8, 6, 1, 4, show the resulting min heap if we initialize it with the efficient algorithm we talked in lecture that takes $O(n)$ time complexity on an array of n elements. Show the intermediate steps in the form of a tree. Do not just write the final result.