CSE 326: Data Structures
Assignment #5
May 14, 1999
Due: Friday, May 21

Reading Assignment: Weiss, Sections 6.1– 6.4, 7.5, 9.1– 9.3, 9.6.1

Problems:

1. Suppose the following actions are taken in the following order:

   - A set S of n keys and another key $k \notin S$ are selected by an adversary.
   - A hash function h is selected uniformly at random from a universal class of hash functions H.
   - The keys in S are hashed to a hash table of size m using the hash function h, where separate chaining is used as the collision resolution strategy.
   - A lookup on the key k is performed.

   What is the expected cost for the (unsuccessful) lookup of key k? Prove your answer.

2. (a) Show the heaps that result from inserting 6,5,2,4,7,1 and then 3 into an initially empty heap.

   (b) Show the heaps that result from three successive `DeleteMins` on the heap resulting from part (a).

3. (a) Suppose that H_1 and H_2 are two heaps of size n_1 and n_2, that $n_1 \geq n_2$, and that every element of H_2 is greater than every element of H_1. Show how to merge H_1 and H_2 into a single heap in time $O(n_2)$, assuming that the array storing the heap H_1 has room in it for $n_1 + n_2$ elements. Give an algorithm that performs the minimum possible number of key comparisons.

   (b) Why is the condition $n_1 \geq n_2$ needed, and how quickly can the heaps be merged if this condition is violated?

4. (**Extra Credit**) Heaps permit one to find the minimum element in constant time, to insert an item in $O(\log n)$ time, and to delete an element (given its index in the array) in $O(\log n)$ time. Explain how to modify the heap data structure and algorithms to provide an implementation of a *double-ended priority queue* with the following characteristics: the data structure can be constructed in $O(n)$ time, a key can be inserted or deleted (given its index) in $O(\log n)$ time and either the minimum *or the maximum* can be found in constant time. (Hint: Arrange the keys so that each node at *even* depth has key value *greater* that that of any of its descendents, and each node at *odd* depth has key value *less* than any of its descendents. This data structure is called a **min-max heap**.

5. Prove that if the Breadth-First-Search algorithm starting at v visits vertex w_1 before vertex w_2, then the distance from v to w_1 is less than or equal to the distance from v to w_2.

6. (a) Describe a graph on n vertices and a particular starting vertex v such that during a breadth-first search starting from v, $\Theta(n)$ nodes are simultaneously in the queue.

   (b) Describe a graph on n vertices and a particular starting vertex v, such that during a depth-first search starting from v $\Theta(n)$ nodes are simultaneously in the state of having started but not completed the execution of the `RecursiveDFS` starting at that node.

   (c) Describe a graph on n vertices and a particular starting vertex v, such that during a depth-first search starting from v, there is a time when $\Theta(n)$ nodes have not yet been encountered, while $\Theta(n)$ nodes have completed the execution of the `RecursiveDFS` starting at that node.