

CSC 263 Homework 3

Due July 27, 2004

1. (16 points)
 - (a) What is the worst-case running time for INSERT in an open-addressing hash table with n items and m slots ($m > n$)? Give an exact expression in terms of the number of slots of the array that are visited.
 - (b) Specify a sequence of items, a hash function and a type of probing that achieves this worst-case. That is, INSERTing the n -th item should take the amount of time given in part (a).
 - (c) What is the amortized cost of each INSERT in the sequence from (b)?
 - (d) Change just the hash function so that every INSERT from the sequence in part (b) takes constant time.

2. (16 points) This question relates to implementing an open addressing hash table using a dynamic array (see the programming part of the assignment). The hash table will be a dynamic array that doubles whenever it becomes 3/4 (or more) full and halves whenever it becomes 1/4 (or less) full (note: these numbers are slightly different from the programming question). More precisely, when the array grows, we have to create a new array of twice the size, go through every slot in the old array and, whenever we find a nonempty slot, rehash the item into the new array. We handle the shrinking case similarly. Assume that the array starts out empty. The hashing will be done by some arbitrary hash function with some arbitrary type of probing. Throughout the question, we will measure the cost of each INSERT and DELETE by the number of array slots that we need to access (read or write).
 - (a) Recall from lecture 5 that the expected number of probes needed to INSERT or DELETE an item from a hash table with n elements and m slots is $\frac{1}{1-a}$ where $a = \frac{n}{m}$ is the load factor. In the scheme described above, what is a_{max} , the biggest that the load factor ever becomes? For simplicity, assume from now on that every INSERT or DELETE requires exactly $\frac{1}{1-a_{max}}$ probes.
 - (b) Let m be the current size of the array. What is the cost of doing an INSERT in the case where the array needs to double? What is the cost of doing a delete in the case where the array needs to halve? Briefly explain your answers.
 - (c) Describe a small modification to the accounting scheme from lecture 7 so that we can always cover all the costs. Make sure to specify how much to charge for INSERT and DELETE, what the credit invariant will be and, briefly, how to maintain the credit invariant.

3. (18 points) A bipartite graph $G = (V, E)$ is a graph where V can be partitioned into two sets V_1 and V_2 (one of which may be empty) such that there are no edges between any two nodes in V_1 or between any two nodes in V_2 .
- (a) Show that any graph that contains an odd cycle is not bipartite.
 - (b) Show how to modify DFS so that, given any graph, it creates a partition like the one described above (that is, it assigns to each node either a 1 or a 2) when the graph is bipartite, and it outputs “not bipartite” when the graph is not bipartite.
 - (c) Is it true that any graph that does not contain an odd cycle is bipartite?