CS 362, HW 9

Prof. Jared Saia, University of New Mexico

1. Problem 5 from the Midterm.

2. Suppose we can insert or delete an element into a hash table in O(1) time. In order to ensure that our hash table is always big enough, without wasting a lot of memory, we will use the following global rebuilding rules:

   - After an insertion, if the table is more than 3/4 full, we allocate a new table twice as big as our current table, insert everything into the new table, and then free the old table.
   - After a deletion, if the table is less than 1/4 full, we allocate a new table half as big as our current table, insert everything into the new table, and then free the old table.

   Show that for any sequence of insertions and deletions, the amortized time per operation is still O(1). Hint: Do not use potential functions.

3. Suppose we are maintaining a data structure under a series of operations. Let $f(n)$ denote the actual running time of the nth operation. For each of the following functions f, determine the resulting amortized cost of a single operation.

   (a) $f(n) = n$ if n is a power of 2, and $f(n) = 1$ otherwise.

   (b) $f(n) = n^2$ if n is a power of 2, and $f(n) = 1$ otherwise.

4. Describe and analyze a data structure to support the following operations on an array $A[1 \ldots n]$ as quickly as possible. Initially, $A[i] = 0$ for all i.

   - **SetToOne(i)** Given an index i such that $A[i] = 0$, set $A[i]$ to 1.
   - **GetValue(i)** Given an index i, return $A[i]$

- **GetClosestRightZero(i)** Given an index i, return the smallest index $j \geq i$ such that $A[j] = 0$, or report that no such index exists.

The first two operations should run in worst-case constant time, and the amortized cost of the third operation should be as small as possible.