CS 561, HW 8

Prof. Jared Saia, University of New Mexico

1. Problem 2 on the midterm.
2. Problem 4 on the midterm. Please also create the table from your recurrence for the following staircase: $[10, 1, 5, 20, 20, 1]$, and give the optimal value and the sequence of steps taken to get that value.
3. Problem 5 on the midterm. Please also create the table for input $(1, 3, 4)$ using your recurrence and say whether Player 1 can always win.
4. Walt is making a device for his friend Hector that counts how many times Hector rings a bell. The software for the device requires a binary counter data structure with INCREMENT and RESET operators.

   In class we discussed an INCREMENT algorithm for incrementing a binary counter in $O(1)$ amortized time. Now we want to include a RESET algorithm that sets all the bits in the counter to 0. Below are the algorithms for INCREMENT and RESET. They use an array B of bits and an integer m giving the largest index in B set to 1.

Algorithm 1 INCREMENT(B, m)

```
1: i ← 0
2: while B[i] = 1 do
3:     B[i] ← 0
4:     i ← i + 1
5: end while
6: B[i] ← 1
7: if i > m then
8:     m ← i
9: end if
```

Algorithm 2 RESET(B, m)

```
1: for i ← 0 to m do
2:     B[i] ← 0
3: end for
```

   Let n be the number of operations on this binary counter. Give the following costs as a function of n.

(a) What is the worst-case run time of INCREMENT?

(b) What is the worst-case run time of RESET?

(c) Prove that in an arbitrary sequence of calls to INCREMENT and RESET, each call has amortized cost O(1). Hint: Use the accounting method and save up dollars during INCREMENT for future calls to RESET.

5. 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.