CSC263: Problem Set 5

November 12, 2019

1 Problem 1

(a) We assign the following amortized costs to our operations:

BINOMIAL-HEAP-INSERT: \$3
BINOMIAL-HEAP-EXTRACT-MIN: \$4$log(n)$

We want to show that the amortized cost of BINOMIAL-HEAP-INSERT is $O(1)$ and BINOMIAL-HEAP-EXTRACT-MIN is $O(log(n))$.

Credit Invariant: The number of credits stored at the root of each tree is \$1.

First, we consider the cost of BINOMIAL-HEAP-INSERT. With \$3, it costs \$1 to create a new heap with just one binary tree, B_0 in it. And \$1 is stored at the root of the tree, thus preserving the credit invariant. Then we add this heap to our existing heap. This uses up the \$1 credit stored at the root of the tree to link together two B_i's (2nd bullet point from the question sheet). Each time this linking occurs, one B_i disappears, so this step is paid for with the \$1 credit stored at the root. Next, we need to update enough pointers to move a tree from one linked list to another (3rd bullet point from the question sheet), and this is paid for with the remaining \$1. Thus, our cost of \$3 works for BINOMIAL-HEAP-INSERT.

Next, we consider the cost of BINOMIAL-HEAP-EXTRACT-MIN. We spend \$1$log(n)$ to extract the root with the minimum value and transform its children into a heap (there are at most $\lfloor log(n) \rfloor + 1$ binomial trees, from notes). We place a \$1 credit at each root in this heap, using \$1$log(n)$ (although this does not matter since the union operation does not use the credits at the root of each tree). And we spend \$2$log(n)$ for the union operation, which altogether amounts to \$4$log(n)$. For the union operation we claim that it costs \$1$log(n)$ and it uses \$1$log(n)$ to merge together two root lists and the remaining \$1$log(n)$ cost to store \$1 credit at the root of each tree in the resulting heap. Note that is this is an overcharge for union, since the cost can be reduced by using the saved credits stored at the root of each tree and the remaining cost of operation can be used to store \$1 at the root of each tree again to preserve the credit invariant, but this does not matter for asymptotic complexity.

Note, that the number of credits stored will be non-negative because by default we start with an empty data structure and we need to insert before extracting. So the number of INSERTs will always be greater than or equal to the number of EXTRACT-MINs.