15-451 Assignment 2

Karan Sikka
ksikka@cmu.edu
Recitation: A
September 9, 2014

1: Max Stacks and Quacks

(a) Queues from stacks:

To prove that a sequence of n ops costs at most $3n$, we set up a hypothetical situation where we spend 3 coins on each operation,
impose a cost of 1 coin for each push and pop, and assuming this, show that the algorithm's bank balance is never negative.
Then we will know that the cost to run the algorithm will not exceed $3n$.

For an insert operation, the algorithm will spend 1 coin on pushing onto S1, and save 2 coins in the bank.

For a dump operation where the stack S1 has size k, the algorithm with spend 2 coins on each of the k elements,
popping each element off S1 and pushing on to S2.

The only way to get an element onto S1 is via an insert, and the only way to get an element off of S1 is a dump.

Each element on the stack S1 must have contributed 2 coins to the bank, and those 2 coins will be used to get it popped off and pushed onto S1.
So we see that the funds saved from an insert are always sufficient to perform the dump.

For a remove operation, there are 2 cases. In one case, S1 is not empty, and we need only spend 1 out of 3 coins received on popping off an element.
In the other case, S1 is empty and a dump needs to be performed.
The dump is funded from money saved during inserts, as shown above.
After the dump, we can reduce to the problem to the first case where S1 is not empty, and we need only spend 1 out of 3 coins received on popping off an element.

We showed that the insert, dump, and remove operations can always be funded by money in the bank assuming we spend 3 coins on each operation.
Therefore n operations cost at most $3n$.

(b) Max-queue:

A max-queue is simply a queue that supports the additional operation of return-max.

We can implement a max-queue using two stacks as done in part a, and we know the push and pop operations are $O(1)$ amortized.

To implement the return-max function, we will add some data structures so that the max of S1 and S2 can each be found in constant time, and then we will take max(S1, S2) to find the max of the Q.

How to turn a Stack "S" into a Max Stack so that its max can be found in constant time:

Create another stack, call it M. When pushing "e" onto S, also push max(e,max(M)) onto M. Max(M) is always at top of M inductively. This takes $O(1)$ time so it doesn't affect the cost of the push.

On a pop from S, also pop the max off of M. This maintains the invariant that the element at the top of the M-stack is the current max of S. This is also done in $O(1)$ time.

Now the max of S can always be found in constant time by looking at the top of the stack M.

Notice that the dump is linear in the number of pushes and pops, so it's time is unaffected since the time of the pushes and pops is unaffected.

To return max, simply take the max((max(s1), max(s2)) which is computed in constant amortized time.

2: Little Gauss's Formula

(a) Recall *Little Gauss's formula*:

(b) Now, equation **??** can be proven by induction as follows: