Algorithms – CMSC 37000

Amortized cost

Instructor: László Babai Ry-164 e-mail: laci@cs.uchicago.edu

1. *(Increment and double data structure)* A data agent maintains a nonnegative integer X and serves two requests: $X := X+1$ and $X := 2X$. Internally, the agent maintains X as a linked list of bits, so his actual expenses are as follows: unit cost for $X := 2X$ (appending a zero at the end); and k units of cost where k is the number of bits switched in the case of $X := X + 1$. For instance, the cost of adding 1 to $X = 23$ is 4 units (23 in binary is 10111, 24 is 11000, so 4 bits are switched).

   Initially, $X = 1$. The agent charges $O(1)$ for each operation ("amortized cost"). Show how the agent will break even, for any sequence of queries. State the actual values of the $O(1)$ charges (how many units the agent charges for each type of request). If the amount charged exceeds the actual cost, the surplus goes into an escrow account; if the amount charged is less than the actual cost then the difference has to be made up from the escrow account. The escrow account is not permitted to slip into the red (must always keep nonnegative balance). Initially the escrow balance is zero.

   (a) Prove that under the stated charges, the escrow balance will never become negative, regardless of the sequence of requests.

   (b) Formalize your proof as follows: State a simple invariant which will guarantee that that the escrow account will never become negative. (An "invariant" is a predicate on the variables that, if true before receiving a request, remains true after the execution of the request.)

2. We simulate queue Q (a FIFO list) using two stacks R and S as follows:

   - simulate enqueue(Q, x) by push(R, x)
   - simulate dequeue(Q) by the following program:

```
00   if S empty then clear(R, S)
01   end(if)
02   x := pop(S)
03   return x
```

   where clear(R, S) is the procedure

```
10   while R not empty
11       x := pop(R)
12       push(S, x)
13   end(while)
```

(a) Verify that this is a correct simulation.

(b) Define a simple invariant which can be used to verify the correctness of the simulation.

(c) Now we add the operation "k-dequeue," meaning that we dequeue Q k times or until the queue becomes empty. Add this operation to the simulation by two stacks. Prove that the following amortized cost table works: enqueue: $O(1)$; k-dequeue: free. State the charge for enqueue (how many units). Define a simple invariant which will guarantee that the escrow balance will never become negative.