6.897 Advanced Data Structures (Spring'05)

Prof. Erik Demaine TA: Mihai Pătraşcu

$\boxed{\text{Problem 3}}$ *Due: Wednesday, Feb. 23*

Timing. As you may know, next Monday is free, and next Tuesday is an MIT Monday (Monday classes are held on Tuesday). To let you enjoy the long weekend, we are making the problem due on Wednesday, which is the day before next lecture. Nonethless, problem 4 will be posted on Tuesday as usual, and solutions will be due on the following Monday.

The Problem. A classic topic in data structures is deamortization, i.e. making amortized data structures worst-case. This problem is about a well-known and extremely useful tool: deamortization of global rebuilding.

Assume we have a data structure, which can run various operations. Running an operation takes time t in the worst-case. However, after *at most* m operations, the data structure must undergo a "global rebuilding" phase, which always takes time mt. Assume m and t do not vary in time. The global rebuilding algorithm acts in place: during its execution, it modifies the data structure it is run on so that it can handle another m operations.

One possible scheduling for global rebuilding is to have it done once after every m operations, causing the running time of every operations to be $2t$, in an amortized sense.

Prove the following: there exists a data structure with worst-case running time $O(t)$ per operation, which handles the same operations as the data structure from above.

Assume that the global rebuilding algorithm can be paused (e.g. you can run 3 steps of the algorithm, pause, run 7 more steps, pause etc). However, a data structure which is undergoing global rebuilding cannot handle operations until the rebuilding is complete (you cannot pause rebuilding and run an operation). Your solution must be online: the next operation is not available until the result of the current operation is known.