Homework 3

Carl Eastlund

Due **Wed., Feb. 27** at **9:00pm**.

> **Collaboration Policy:** Your work on this assignment must be your own. You *may not* copy files from other students in this class, from people outside of the class, from the internet, or from any other source. You *may not* share files with other students in this class.
>
> You *may* discuss the problems, concepts, and general techniques used in this assignment with other students, so long as you do not share actual solutions.
>
> If you are in doubt about what you *may* and *may not* do, ask the course instructor before proceeding. If you violate the collaboration policy, you will receive a zero as your grade for this entire assignment and you will be reported to OSCCR (`northeastern.edu/osccr`).

You will implement five new datatypes in this assignment. Three are concrete datatypes; I will tell you what sort of representation to use, and you will implement that in Racket. Two are abstract datatypes; I will only tell you what operations they need to support, and you must both design the representation and implement your design in Racket.

For each datatype, you will be assigned a set of operations that the datatype must support, and an upper-bound on the running time of the operations. In each case, you will be asked to describe your design and analyze its efficiency in LaTeX (in `solution.tex` and `solution.pdf`) and implement it in Racket (in `solution.rkt`).

For the concrete datatypes, the *only* compound data structures you may use in Racket are lists, arrays, boxes, and `struct` definitions. The *only* built-in operations you may use on these data structures are those that run in $\Theta(1)$ time, plus `make-vector`, which runs in $\Theta(n)$ time. You must implement all other operations yourself. For reference, all the functions defined by `struct` are in $\Theta(1)$, as are `empty?`, `cons?`, `cons`, `first`, `rest`, `vector-length`, `vector-ref`, `vector-set!`, `box?`, `box`, and `unbox`.

For the abstract datatypes, you may use lists, arrays, boxes, and `structs` as described above; you may also use any of the other datatypes you have implemented.

1. Concrete datatypes:
   (a) Doubly-linked lists.
   (b) Balanced binary trees.
   (c) Hash tables.
2. Abstract datatypes:
   (a) Sets.
   (b) Catenable deques.