1 Names and Emails

- Aleksandar Makelov — `amakelov@college.harvard.edu`
- Ben Wetherfield — `bwetherfield@college.harvard.edu`
- Chan Kang — `chankang@college.harvard.edu`
- Michael Fountaine — `mfount@college.harvard.edu`

2 Overview

Problem: Verify Timsort, python's preferred sorting algorithm!

Solution sketch: We will take an incremental approach: We're going to start with an abstract "implementation" of Timsort, possibly black-boxing certian difficult-to-implement features such as heuristics used in the algorithm, and implement a verified version of that implementation to sort lists of natural numbers (defined inductively). Timsort is a hybrid sorting algorithm, requiring other sorting sub-algorithms and some simple data structures, so we will divide our project into interfaces for each of these, along with interfaces for proof tactics.

Goals: Primarily, we'd like to verify Timsort as a way to learn more about Coq and certified programming. (Or, if Timsort proves to be too complicated (by our estimation by the time the final spec is due), we hope to verify another hybrid/adaptive sorting algorithm, with that same learning outcome in mind.)

3 Prioritized Feature List

Core Features

Cool Extensions

4 Technical Specification

5 Next Steps