CS472 Sorting Algorithm Proof Project

- Aarav Surkatha
- Rahin Jain
- Emilio Kiryakos

As a team, we started on the CS472 project with the goal of formally verifying sorting algorithms using the Dafny verification tool. At the outset, we were motivated by the idea of combining our theoretical understanding of algorithms with the proof methods we learnt from class. However, despite our persistent efforts throughout the semester, we were ultimately unable to complete formal proofs for the sorting algorithms within the given timeframe. This outcome, though disappointing, became a source of learning and insight into both formal verification and problem-solving.

In the early phases of the project, we thought of trying to prove both Radix Sort and Bucket Sort. We began by reviewing Dafny's syntax and its support for inductive proofs and loop invariants. Our strategy was to start with simpler algorithms, create basic helper functions and gradually build up to more complex ones, by decomposition where possible. We consulted documentations of dafny tutorials, watched YouTube methods, and dissected example proofs available on github (1,2,3)for other sorting algorithms to get ideas.

What We Tried

We started with Insertion Sort, aiming to prove both partial and total correctness. We first encoded the algorithm and attempted to define preconditions, postconditions, and loop invariants. Early attempts revealed the nuances of ghost variables and the need for carefully crafted invariants. We then moved on to Bucket Sort, hoping that its structure would help us with modular verification. Along the way, we experimented with lemmas to prove properties like sortedness, permutations, and bounds preservation.

We made use of Dafny's built-in features such as multiset equality and sorted predicates, but struggled to align the operational behavior of the sorting algorithms with the required logical assertions. We also incorporated feedback from peers and the instructor, such as simplifying our invariants, trying recursive formulations, and incrementally proving properties of helper functions.

To support our implementation, we wrote several Dafny methods:

- ArrayMaxNumber and ArrayMinNumber to determine range bounds.
- BucketIndex to map values into specific buckets.
- DistributeToBuckets to collect values into bucket sequences while preserving permutation.
- MergeBuckets to sort each bucket using InsertionSort and merge them back into a single array.

The full Dafny code includes specifications and invariants for each step. While most helper methods were verified in isolation, composing them into a fully verified BucketSort method remained out of reach. The final method includes verification for properties like sortedness and permutation, but fails when combining all helpers in sequence due to complex interactions between loop invariants and memory state.

Challenges and Roadblocks

One of the biggest challenges we encountered was maintaining the delicate balance between algorithmic complexity and Dafny's logical constraints. For instance, even small modifications to our implementation would cause previously verified invariants to break, requiring us to repeatedly refactor and reassess our assumptions. Loop invariants were especially difficult to get right; they needed to be strong enough to support our proof goals but not so strong that they became impossible to maintain inductively.

Another recurring issue was the lack of detailed feedback from Dafny's verification engine when proofs failed. Often, we would receive only a generic failure without insight into which part of the proof was at fault. This made debugging time-consuming, especially as our project timeline narrowed.

Additionally, while we made genuine attempts to divide the work efficiently, it quickly became clear that verification required contextual understanding of each component in depth. This made parallelization difficult and often led to rework when integrating individual components.

Feedback and Lessons Learned

Throughout the semester, we received valuable and technical feedback during check-ins and workshop sessions. One particularly helpful comment was that although our structure was reasonable, we had not yet developed the right loop invariants for BucketSort. It was pointed out that we needed to pay close attention to reasoning about how many elements reside in different parts of the structure at each stage.

We were encouraged to closely investigate the causes of index-out-of-bounds errors and to explicitly prove, even on paper, that all index accesses were safe. A major insight was recognizing that the consistency between the number of buckets and the array length was crucial for expressing and maintaining useful invariants. These suggestions pushed us to rethink how we model element distribution and indexing in Dafny, and led us to temporarily isolate and verify components like DistributeToBuckets and MergeBuckets independently.

This feedback also shifted our mindset from attempting full proofs end-to-end to proving properties incrementally, and to iteratively refining loop invariants using test failures as debugging aids. Though we still struggled with some of the more complex invariants, we grew to better understand Dafny's expectations for inductive reasoning.

From a technical standpoint, we gained a reasonable understanding of Dafny's verification model, including its limitations and strengths. We learned how to structure specifications, the importance of ghost code, and how subtle changes in preconditions or function structure could significantly impact provability.

Final Thoughts

While we did not complete a fully verified proof of the sorting algorithms, we walk away from this project with an understanding of formal methods and the effort required to verify even seemingly simple algorithms. We are appreciative of the little progress we made, and the intellectual curiosity that drove us forward despite obstacles.

In conclusion, the project was challenging but intriguing. We are grateful for the opportunity to tackle such a complex task and for the feedback and support from you throughout the journey.