CSCI 2600 - Homework 3
Problem 2

Jordan Ye

March 13, 2025

Problem 2 Written Answers

(a) We have chosen the array representation of a polynomial: `RatNum[] coeffs`, where `coeffs[i]` stores the coefficient of the term of exponent i. An alternative data representation is the list-of-terms representation: `List<Term> terms`, where each Term object stores the term's RatNum coefficient and integer exponent. The beauty of the ADT methodology is that we can switch from one representation to the other without affecting the clients of our RatPoly. Briefly list the advantages and disadvantages of the array representation versus the list-of-terms representation.

Answer:

The array representation offers O(1) direct access to coefficients by exponent, making it efficient for operations that need to access terms by degree. It is memory-efficient for dense polynomials where most exponents have non-zero coefficients. However, it wastes space for sparse polynomials with large gaps between terms, requires allocating space for all terms from 0 to the highest degree, and operations like addition may require resizing arrays.

The list-of-terms representation is more space-efficient for sparse polynomials with few terms, making it easier to maintain terms in sorted order without wasting space for zero coefficients. Its disadvantages include O(n) access time to specific terms (requiring linear search), more complex implementation for operations needing coefficients by degree, and the overhead of storing the exponent with each term.

(b) Where did you include calls to `checkRep()` in RatPoly (at the beginning of methods, the end of methods, the beginning of constructors, the end of constructors, some combination)? Why?

Answer:

I included calls to `checkRep()` at the end of all constructors and at the end of methods that create new RatPoly objects. This approach is sufficient because RatPoly is an immutable class (all fields are final), so once a valid object is constructed, it cannot be modified to violate the representation invariant.

Since methods like `add()`, `sub()`, `mul()`, etc., don't modify existing objects but return new ones, checking the representation at the end of these object-creating operations ensures the invariant is preserved for all RatPoly objects. Including checkRep() at the beginning of methods would be redundant since the existing objects have already been verified at construction time and cannot be modified afterward.