November 13, 2017

Matrix multiplication in softly quadratic time

3. Rational matrix multiplication

Approach

We don't have strong results on this front.

The worst case time lines up with rational brute-force. The main cause for concern is blow-up of numerators and denominators given that intermediate numerators and denominators end up being co-prime. The most straightforward approach is to find LCM over all $O(n^2)$ input values, but this would lead to an extra factor of $O(n^2)$ for running time, as the number of bits needed for finding LCM for p co-prime m-bit denominators is $p \cdot m$.

There is some consolation to be had, however. Blow-up of intermediate term numerators and denominators happens with Strassen-likes, as well. If we assume that it is difficult to anticipate the best order for multiplying and adding rational numbers for, e.g., brute-force approach, then we get an extra factor of $O(n)$ for running time, as we deal with $O(n)$ input values at a time for a dot product. Also of note is the fact that for Strassen-likes, we tend to substitute multiplications for more additions. Consider Strassen 1969; 8 multiplications and four additions turns into 7 multiplications and 18 additions and subtractions. This means that we have more intermediate terms, which without further insight into the likelihood of rational simplification for varying situations (e.g. Strassen-likes or our custom approach to be detailed later), bodes badly for minimizing largest blow-up in terms of numerator or denominator size in bits. It may be possible to characterize the difficulty of an MM problem w.r.t. rational term numerator/denominator blow-up, but we do not address this currently.

We can describe an approach that leads to an extra factor of $O(n)$ for running time, though we do not promise that the blow-ups for numerator and denominator match for our approach and rational brute-force.

We know that based on average size of a subtree from a balanced tree is $O(1)$ (again), so average numerator or denominator for memoize phase for a node takes around an extra factor of $O(1 \cdot n) = \mathrm{O(n)}$ more bits. We note that each point is for a row or a column, each of which would be made up of n components that overall contribute O(n) separate terms for a rational dot product. This is in contrast to having an extra factor for worst-case time of n^2. We leave for later the answering of the question of whether this experimentally leads to time that is an improvement on rational brute-force. We also will have to consider whether the speed-up between our rational approach and rational brute-force is comparable to speed-up between, say, our floating-point approach and floating-point brute-force. We think this will be the case. However, we would also have to take care to have enough degeneracy for our custom rational MM approach, because otherwise we have unavoidable slow-down.

We need more changes, however, for our rational MM approach. We can say that we memoize by storing rational numbers, but we also must take care to change how we interact with memoized values. When we perform "downward filtering", we need to make sure denominators match before removing one-bits, and then we need to simplify. On average, for our balanced upper-layer (for x) tree and lower-layer (for y) trees, a subtree is size $O(1)$; this is given that we amortize across nodes in upper-layer tree or lower-layer trees. So, while bit-wise OR and AND for avoiding intersection for downward filter can involve a large subtree and thus huge lowest common denominator, the work can be considered efficient enough overall. This would make the sixth time (including certain parts of approach for floating-point MM) that we make use of the fact that average size of a subtree for a balanced tree is $O(1)$.

We have yet to establish that worst-case times for rational brute-force and custom rational approaches loosely match. For a query, we care about a specific row and a specific column. We will ignore time for memoization. As part of union-intersection arithmetic, we subtract union from sum of weighted one-bits (i.e. rational numbers) for row components and for column components. The value from which we subtract is akin to having sum of dot product of row with one-vector and dot product of column with one-vector. We note that in the worst case, these size-n row and column vectors lead to another $O(2 \cdot n) = O(n)$ for coefficient for number of bits necessary for blown-up numerator or denominator for an output cell, leaving the coefficient unchanged at $O(n)$.

The extra factor of n is better than an extra factor of n^2; this is also slightly pessimistic, because there are many opportunities for simplification to occur, though we likely cannot map directly to sum and multiplication operations for rational brute-force.

It is worth noting that for each of four subproblems, we don't overshoot auxiliary dot product, which doesn't overshoot main dot product. However, this doesn't affect the size of intermediate numerator and denominator.

We don't claim to support rational arithmetic s.t. we maximize simplification of numerator relative to denominator over all valid addition or multiplication or bit-wise OR or AND operation sequences.

One alternative approach is using Chinese remainder theorem with rational number reconstruction to break problem into those of around p bits at a time s.t. we assume that worst-case blow-up of denominators and numerators occurs; we get an extra factor of n^2 for this approach.

A second alternative approach is somehow using adaptive rational arithmetic; at the very least, we know we could via modifying brute-force.

When ignoring an extra n factor for worst-case time that is inherent for any rational MM approach, ω can still be said to be softly two for rational MM for rings.

We need to mention that we must choose bit bounds for nodes based on height and mention that we have $O(2 \cdot n) = O(n)$ contributions for upward union and downward filtering each.

Running time

$O(n \cdot (\frac{m}{m'})^2 \cdot n^2 \cdot \log^3(n))$ with co-prime denominators assumed.

This time could be reduced by avoiding Demmel-Hida reliance for query phase; i.e., if we handle merging of superaccumulators in a more sophisticated manner. Presumably, we could get a time with two logarithms.

Note

m is the size of a input numerator or denominator in terms of number of bits.

m' is the size of a word in terms of number of bits.

References

- Strassen - Gaussian elimination is not optimal (1969)