To address the problem of slow execution time in the existing implementation, we rewrote the code in C and implemented a number of new features, specific to the mapping of sRNAs. We introduced a novel hash function which allowed a novel, one-pass, method for reading the reference sequence. Implementing in C allowed finer control of memory management and we were able to reduce memory consumption significantly.

The user is able to choose the number of bins in the hash table to suit their memory requirements. Although one would expect the execution time to increase with the number of bins due to a reduced number of collsions, we found that for larger bin numbers total execution time increased in response to the time required for the creation of the hash table itself. The fasta reference sequence file is also read one sequence at a time, so as to reduce the memory footprint of the program.

The novel hash function uses a direct two-bit representation of the sequence in an unsigned 64 bit integer as the hash value. This restricts the size of a sequence that can be hashed to 32 base pairs and requires that all sequences are of the same length. While this would be too restrictive for general hashing of sequences it is reasonable for sRNAs which are usually of length 24 or 21 and are compared only with sRNAs of identical length in the current application.

Usually, after the sequence window is built the hash function needs to iterate through the sequence to generate the hash. A benefit of the hash function that we developed is that it can be progressively built, forward and reverse one base pair at a time. This allowed us to develop a one-pass method for querying the reference sequence for matches. The existing hash is bit-shifted and the two bit representation of the next character is added on the first and last two bits for the reverse and forward hashes respectively. A further benefit is that sequence equality checking in the hash table is improved. Because the hash is a perfect representation of the sequence it can be stored in the hash table for equality comparison instead of the character array. This allows greatly increased speed of equality checking in the case of hash collisions because the entire series of characters of the query sequence and existing sequence don’t need to be checked for equality, just the single integer. It also slightly reduces the size of the hash table because instead of having 8 bits per character stored there are a fixed 64 bits stored per element in the hash table.