Algorithm:

For our main algorithm, we chose the greedy search (with some modifications). First, we compare every motif-length substring in sequence 0 with every motif-length substring in sequence 1. We score each substring by giving it 1 point for each matching character in both sequences. Only substring that score above a threshold are saved (in order to save time later). We take this list of possible motifs and apply them to the rest of the sequences. At each iteration, we take each motif and score like before (compare character in motif with character in the sequence and give it 1 point if it matches). Once again, we toss any motif that that scores below the threshold. Throughout, for each motif, we store the sites associated with each sequence. At the end, we sort the remaining motifs by score and pick the top one.

Alternatively, instead of dropping scores below a threshold, we can simply drop the bottom half scoring motifs. This would most likely be more efficient for bigger scale inputs.