Gorilla Report

Adam Tovatt

May 10, 2025

1 Results

All test cases passed when running the provided `check_solution.sh` script. The program runs efficiently, completing even the largest test case (secret/4huge.in) in around 209 milliseconds on a modern desktop machine (intel i9-9900k). The majority of the time is spent computing the dynamic programming tables and reconstructing the alignments for each query.

Section	**Time (ms)**	**Percent of Total**
readInput	0.31	0.1%
solve	209	99.9%

2 Implementation details

The solution is based on a classic dynamic programming algorithm for sequence alignment. The input is parsed into a reusable alignment context (containing the character set and cost matrix), and a list of query string pairs. Each alignment query is independent and solved using an iterative algorithm that fills a 2D DP table and a backtrack table to reconstruct the optimal alignment. The implementation is fully parallelized across queries using `AsParallel` to reduce overall runtime.

The core data structures used are:

- A 2D `int[,]` DP table for tracking the maximum gain up to each prefix
- A 2D `Direction[,]` backtrack table to reconstruct the alignment path
- An `AlignmentContext` object that maps characters to indices and stores the cost matrix

The overall running time for a single query is $O(NM)$, where N and M are the lengths of the two strings. This is due to filling a DP table of size $N \times M$ and doing constant-time work per cell. Because all queries are independent, the solution can be parallelized safely across them.

Discussion Questions

Is your solution recursive or iterative?
The solution is iterative. Both the DP table and the backtrack reconstruction are implemented with loops, not recursion.

What is the time complexity, and more importantly why?
The time complexity per alignment is $O(NM)$ because each of the $N \times M$ table entries is computed once using three constant-time comparisons.

What would the time complexity of a recursive solution without cache be?
It would be exponential, specifically $O(3^{N+M})$, because each step could branch into up to three recursive calls without remembering results.

Can you think of any applications of this type of string alignment?
Yes. Bioinformatics to align DNA or protein sequences, and in natural language processing for fuzzy matching or edit distance. Not sure how git works but git probably uses some kind of string alignment too.

What could the costs represent in the applications?
In biology, costs can represent biological likelihoods of mutation, substitution, or gaps in evolutionary history. In text matching, they might represent character substitution penalties or edit weights.