http://dx.doi.org/10.7124/bc.000AF0

Evaluation of performance of the same analysis in different programming languages

I.-A. Ripan[1], K. Brzezinska[2], Ye. Makedon[3]

[1] Stefan cel Mare University of Suceava
13, University Str., Suceava, Romania, 720229
[2] FH Campus Wien, Austria
226, Favoritenstrasse, Vienna, Austria, 1100
[3] Constructor University Bremen gGmbH
1, Campus Ring, Bremen, Germany, 28759
mail@kbrzezinska.com

Aim. This study evaluates the performance of the Needleman-Wunsch algorithm across Python, C++, and C#, focusing on execution time and memory usage. The algorithm, based on dynamic programming with a time complexity of O(mn), ensures optimal global sequence alignment, making it essential in bioinformatics. An existing repository was used for implementation, and despite its age, the core logic remains valid for performance comparison. While newer algorithms like Smith-Waterman and BLAST offer different advantages, Needleman-Wunsch is still the gold standard for global alignments due to its simplicity and accuracy. **Methods.** In this study, the algorithm was implemented in Python, C++, and C#. Execution time and memory usage were measured for small and large data sets using standardized tools for consistency. Validation used sequences CTCGCAGC and CATTCAC, with scoring criteria of 10 points for a match, –2 for a mismatch, and –5 for a gap. All implementations yielded the aligned sequences CAT-TCA-C and C-TCGCAGC. Testing included *E.coli* reference and randomly mutated sequences, extracting 1000 nucleotides from the first codon of both fasta files. The algorithm was applied 10 times to measure peak memory and execution time, with scalability evaluated on 10,000 nucleotide samples. Two implementations for each language ensured independent assessment of memory management and execution time, validating the algorithm's theoretical complexity and analyzing operational overhead. **Results.** The results showed varying performance characteristics among the languages. C++ had the fastest execution and lowest memory usage, proving most efficient for large-scale analysis. C# performed well in speed but had higher memory consumption. Python, though easier to use and faster to develop in, had significantly higher execution times and memory usage. These results were expected, given that C++ is compiled and Python is interpreted. The complexity difference led to significant disparities in memory and time performance, especially as sample sizes increased. **Conclusions.** This study highlights the trade-offs between ease of use, speed, and resource consumption. C++ is ideal for performance-critical applications, while Python offers accessibility for rapid development despite higher resource needs.

K e y w o r d s: Performance evaluation, Needleman-Wunsch algorithm, Python, C++, C#.