1. Design

    1.1. IdealParScoring.java

        1.1.1. My design for IdealParScoring involved me creating a matrix of data driven futures, s, in the constructor. And, in the function scoreSequences, wrapped inside a finish, I used an async-await inside two nested for loops (for xLength and yLength) to calculate the first value, second value, and third value needed to compute the input for an entry in the matrix using the Smith-Waterman algorithm.

    1.2. UsefulParScoring.java

        1.2.1. My design for UsefuParScoring involved me creating a matrix of data driven futures, s, in the constructor. And, in the function scoreSequences, wrapped inside a finish, I used an async-await inside two nested for loops (for xLength and yLength) to calculate the first value, second value, and third value needed to compute the input for an entry in the matrix using the Smith-Waterman algorithm.

2. UsefulParScoring.java Measurements

    2.1. Execution time, SeqScoring.java:

    2.2. Execution time, UsefulParScoring.java: 1min, 41 seconds