Team Members: Zheng Chen, Daniel Martin

# PSEUDOCODE:

```
sequence_alignment_dp(costs, string1, string2):

        n = length of string1
        m = length of string2

        S = empty 2d list with size n+1 * m+1

        S[0][0] <- 0

        #Base Case
        for i <- 1 to n:
                S[i][0] = S[i-1][0] + costs[string1[i]]['-']
        for j <- 1 to m:
                S[0][j] = S[0][j-1] + costs['-'][string2[j]]

        #Tabulation
        for i <- 1 to n:
                for j <- 1 to m:

                        S[i][j] = min { S[i-1][j-1] + costs[string1[i]][string2[j]],
                                        S[i-1][j] + costs[string1[i-1]]['-'],
                                        S[i][j-1] + costs['-'][string2[j-1]]
                                }


        #Backward Pass to find completed strings
        seq1 <- empty string
        seq2 <- empty string
        a = n
        b = m
        while a > 0 or b > 0:
                if a = 0:
                        Add '-' to seq1
                        Add string2[b] to seq2
                        b = b - 1
                else if b = 0:
                        Add string1[a] to seq1
                        Add '-' to seq2
                        a = a - 1
```

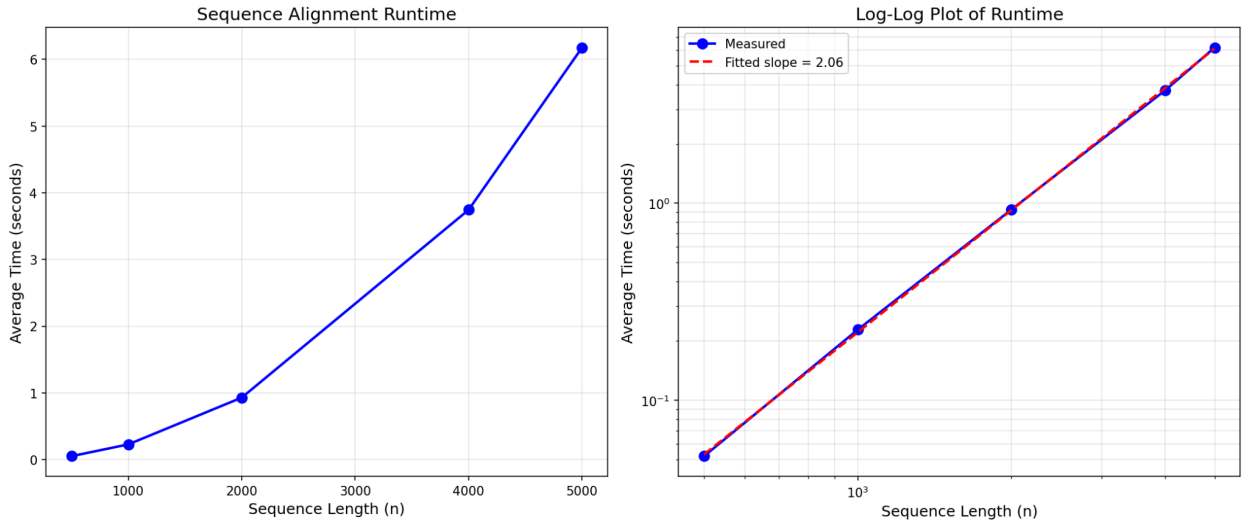
```
else if S[a][b] = S[a-1][b-1] + costs[string1[a]][string2[b]]:
        Add string1[a] to seq1
        Add string2[b] to seq2
        a = a - 1
        b = b - 1
else if S[a][b] = S[a-1][b] + costs[string1[a]]['-']:
        Add string1[a] to seq1
        Add '-' to seq2
        a = a - 1
else:
        Add '-' to seq1
        Add string2[b] to seq2
        b = b - 1

Reverse seq1
Reverse seq2

min_cost = S[n][m]

Return seq1, seq2, min_cost
```

# ANALYSIS OF RUNTIME

This dynamic programming implementation has a worst-case runtime of O(n * m) where n is the length of string 1 and m is the length of string 2. The algorithm consists of a nested for loop where the outer loop runs n times and the inner loop runs m times and the inner code runs at O(1) time, therefore the inner constant-time operations will run n*m times giving this algorithm O(n*m) runtime.

# REPORTING AND ANALYZING EMPIRICAL RUNTIME

EXPERIMENT SETUP:
Runtime experiments were conducted on a MacBook Pro with Apple Silicon M3. For each of five sequence lengths (500, 1000, 2000, 4000, 5000), 10 random pairs of DNA sequences were generated using the alphabet {A, G, T, C}. The timing was measured using Python's time.time() capturing only the execution of the sadp() function itself, which is excluding file I/O, sequence generation, and output of the aligned strings. The reported time for each length is the average over the 10 trials.




The left plot shows average runtime as a function of sequence length on a linear scale. The right plot shows the same data on a log-log scale, with a fitted line.
The fitted slope on the log-log plot is 2.06. Since the runtime is of the form O(n^s) and the log-log plot produces a straight line with slope m, the experimental runtime is inferred to be: O(n^2.06) = O(n^2)

TABLE

| Length | Avg Time (s) | Min (s) | Max (s) |
|---|---|---|---|
| 500 | 0.0519 | 0.0513 | 0.0536 |
| 1000 | 0.2278 | 0.2224 | 0.2476 |
| 2000 | 0.9293 | 0.9084 | 0.9776 |
| 4000 | 3.7440 | 3.6945 | 3.7904 |
| 5000 | 6.1760 | 5.9445 | 6.7453 |

# INTERPRETATION AND DISCUSSION

The empirical results closely match the theoretical asymptotic analysis. The algorithm fills an n×m DP table where both sequences have length n, requiring O(n^2) work. The log-log plot confirms this because the measured data points fall almost perfectly on the fitted line with slope 2.06, which is very close to the theoretical slope of 2.

The small deviation from exactly 2.0 is expected and can be attributed to real-world factors such as memory access patterns and Python interpreter overhead, all of which introduce minor variation but do not change the fundamental quadratic growth rate.

The min/max times across trials are also deviating very little (5.9445s to 6.7453s at length 5000), indicating the results are stable and not heavily influenced by background system operations. Overall, the empirical growth curve matches the O(n^2) expectation from asymptotic analysis very well.