DNA Sequencing

1. Description of algorithms

The following extracted cells (markdown or code) appear related to algorithm descriptions or implementations. I include the content excerpts and any detected function definitions / comments.

- **Needleman–Wunsch Algorithm:** Performs global alignment, meaning it aligns two sequences from end to end, introducing gaps where necessary to maximize similarity across their entire lengths.
    - Initialize a scoring matrix with penalties for gaps along the first row and column.
    - Fill the matrix using dynamic programming, with scoring rules:
        - Match: +2
        - Mismatch: −1
        - Gap penalty: −2
    - Store traceback directions (diagonal, up, left) to reconstruct the alignment.
    - Perform traceback from the bottom-right corner to reconstruct the globally aligned sequences.

- **Smith–Waterman Algorithm:** Performs local alignment, finding the best matching subsequence between two sequences rather than forcing end-to-end alignment.
    - Initialize the scoring matrix with zeros.
    - Fill the matrix using dynamic programming:
        - Match: +2
        - Mismatch: −1

        - Gap penalty: −2
        - Negative values are reset to zero
    - Track the maximum score and its position.
    - Perform traceback from the cell with the maximum score until a zero is encountered, yielding the best local alignment.

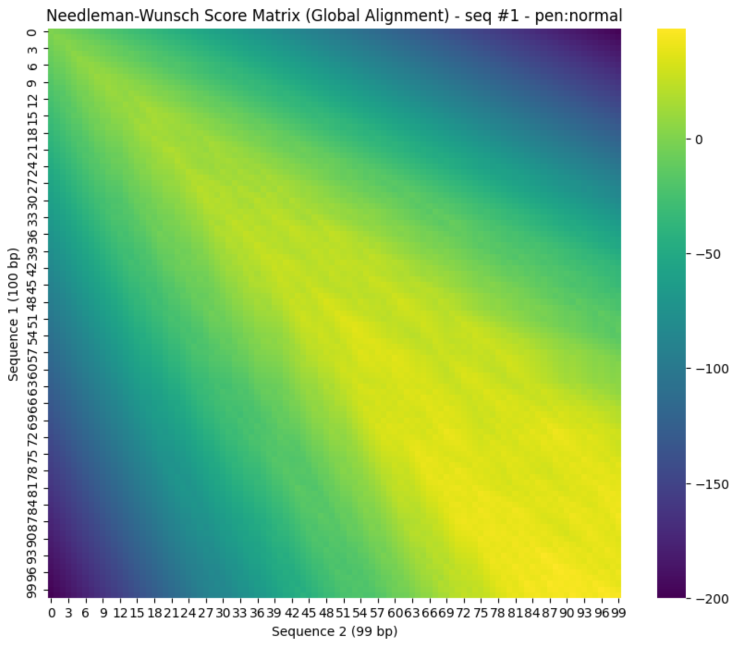
2. Discussion of alignment outputs

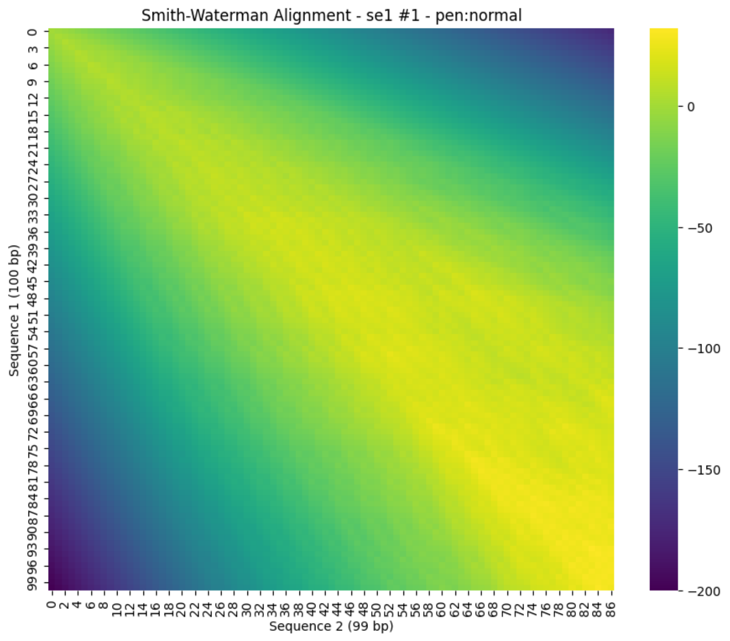
The alignments we got with Needleman–Wunsch (global) and Smith–Waterman (local) gave us two different but useful ways of looking at how similar the original sequences are to their variants.

- Global alignments (Needleman–Wunsch): This method tries to line up the entire length of both sequences from start to finish. In our case, it gave a moderate score (for example, 46 for sequence 1 vs. its variant). We could see long stretches where the sequences matched, but also many gaps and differences. The heatmaps looked like diagonal lines, which shows that the sequences are similar, even though some parts don't line up perfectly.
- Local alignments (Smith–Waterman): This method only looks for the best-matching pieces of the sequences, not the whole thing. The alignments were shorter, but the matching parts were much stronger. The heatmaps showed "islands" of high scores, pointing to regions where the sequences really are very similar, while ignoring the rest that didn't match as well.
- Comparison:
    - Global alignment is best when you want to see how similar two sequences are overall.
    - Local alignment is better when you just care about the strongest matching regions, like conserved patterns.

- In our results, Needleman–Wunsch gave us longer alignments but also more penalties for differences, while Smith–Waterman zoomed in on the most meaningful matching sections.

Both approaches are helpful. Needleman–Wunsch shows the big picture of how related two sequences are, while Smith–Waterman highlights the most important local similarities. Together, they give a more complete view of the relationship between the sequences.





3. Analysis of parameter effects

The alignment results depend heavily on the scoring parameters we choose: the rewards for matches and the penalties for mismatches and gaps.

Match score: If we give more points for a match, the algorithm will try harder to line up long stretches of matching bases, even if it means slipping in more gaps. If the match score is lower, it won't "cling" to matches as much, so alignments are looser.

Mismatch penalty: A bigger penalty makes the algorithm less forgiving of differences—only really similar sequences will line up well. A smaller penalty lets more mismatches slide, so even less-related sequences can get aligned.

Gap penalty: If gaps are high, the algorithm avoids them, so alignments are tight but might miss real insertions or deletions. If gaps are low, the algorithm will add more, stretching the alignment to find similarities—but it might also introduce "fake" matches.

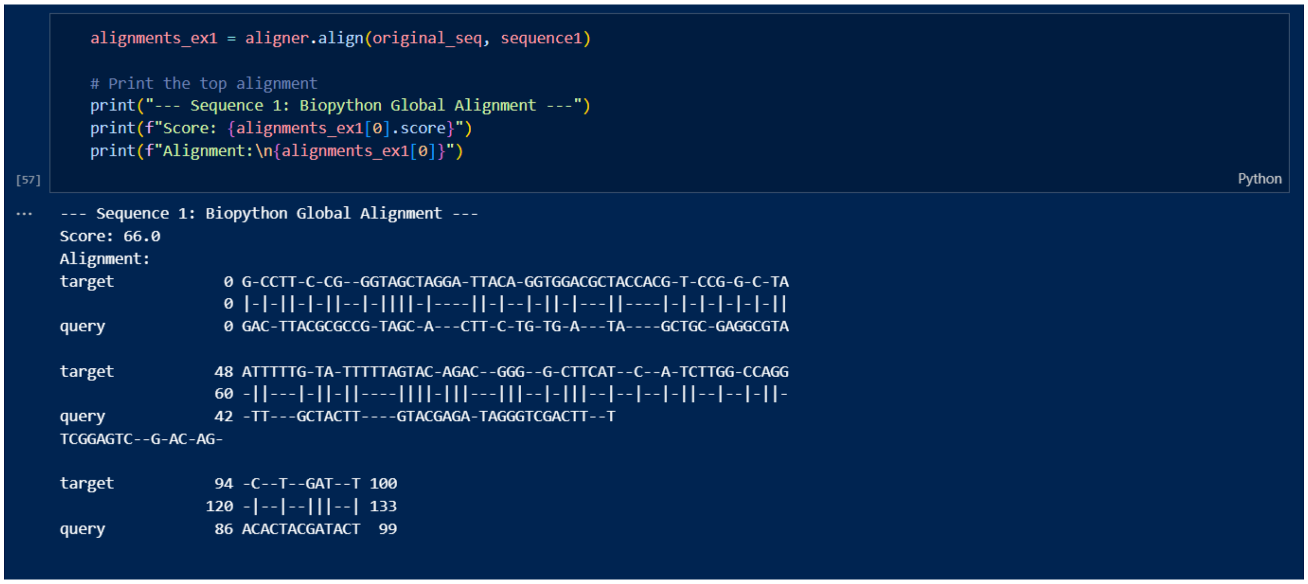
What we used in the practice:

- High gap penalties → Cleaner diagonals in the heatmaps but shorter alignments.

- Low gap penalties → We saw longer alignments with more gaps, sometimes aligning regions that might not be truly related.

  Changing mismatch penalties → Adjusting mismatch penalties changed how many substitutions were tolerated, shifting the balance between sensitivity (finding more similarities) and specificity (only keeping the strongest ones).

4. Use of the Biopython library



```python
alignments_ex1 = aligner.align(original_seq, sequence1)

# Print the top alignment
print("--- Sequence 1: Biopython Global Alignment ---")
print(f"Score: {alignments_ex1[0].score}")
print(f"Alignment:\n{alignments_ex1[0]}")
```

[57] Python

```
--- Sequence 1: Biopython Global Alignment ---
Score: 66.0
Alignment:
target            0 G-CCTT-C-CG--GGTAGCTAGGA-TTACA-GGTGGACGCTACCACG-T-CCG-G-C-TA
                  0 |-|-||-|-||--|-||||-|----||-|--|-||-|---||----|-|-|-|-|-|-||
query             0 GAC-TTACGCGCCG-TAGC-A---CTT-C-TG-TG-A---TA----GCTGC-GAGGCGTA

target           48 ATTTTTG-TA-TTTTTAGTAC-AGAC--GGG--G-CTTCAT--C--A-TCTTGG-CCAGG
                 60 -||---|-||-||----||||-|||---|||--|-|||--|--|--|-||--|--|-||-
query            42 -TT---GCTACTT----GTACGAGA-TAGGGTCGACTT--T
TCGGAGTC--G-AC-AG-

target           94 -C--T--GAT--T 100
                120 -|--|--|||--| 133
query            86 ACACTACGATACT  99
```