Declaration

We made use of AI tools for general guidance and clarification while preparing this assignment, including help with refining mathematical notation, parts of the Python implementation, and occasional grammar checks or rephrasing.

Problem 1

how changes to the scoring function affect the maximum score alignment.

a

> Prove formally that if the score of a gap is more than half the score of a substitution, then a maximum score alignment of any two sequences does not contain mismatches. In this question you may assume that all gap scores are equal and all substitution scores are equal.

Denote Q as the score function such that:
For every x, y: $Q(x,-) = Q(-,y) > 1/2 * Q(x,y)$

Denote T_m as a sequence T of length m.

We will prove by contradiction.
Assume that exists a maximum score alignment of two sequences T_m, U_n such that exists an index i, j where the alignment chose substitution (There can be a case where $i \neq j$ if we chose more gaps in one of the sequences then the other) ($Q(T[i], U[j])$)
We will denote the maximum score as M (for maximum)

We will then choose the exact same sequence, but replacing the choice of substitution $Q(T[i], U[j])$ with two gaps $Q(T[i], -)$, $Q(-, U[j])$
The sum of any two gaps is larger then any single substitution so S' (The new score)
Is equal to the equation:
$S' = S - Q(T[i], U[j]) + Q(T[i], -) + Q(-, U[j]) >$ (replacing two gaps)
$S - Q(T[i], U[j]) + Q(T[i], U[j]) >$ (canceling terms out) S
$S' > S$
So we proved that if exists a score alignment containing a substitution there exists a new score alignment with higher value in contradiction to our assumption.

This means that if the statement in the question is true then any maximum score alignment of any two sequences will contain only gaps, and no substitution.

b

> Demonstrate using an example that offsetting a scoring function by an additive constant can change the maximum score local alignment. The score function Q' is said to be an E-offset of Q if $Q'(x,y) = Q(x,y) + E$ for all $x, y \in \Sigma \bigcup \{-\}$. Provide an explicit and fully detailed example

Because we only need an example, we can choose an easy case:
Take sequence $T = \{a\}$, $U = \{b\}$ for language $\{a, b\}$
Scoring function:
$Q(a,a) = Q(b,b) = 1$
$Q(a,b) = Q(b,a) = 1$
$Q(-,a) = Q(a,-) = Q(-,b) = Q(b,-) = -1$
Offset:
$E = 100$

The score function Q will give the obvious solution:
$T' = T = U' = \{a\}$
With the score 1
And the choices:
$Q(a,b)$
For the sequence.

The score function Q' on the other hand will give a different solution (still in an obvious way):
$T' = \{a, -\}$
$U' = \{-, b\}$
With the score 198
And the choices:
$Q'(a,-), Q'(-,b)$
For the sequence.

Just to show that it does in fact give different solutions (Those are the only two possible)
$Q(a,-), Q(-,b) = -2 < Q(a,b)$
$Q'(a,b) = 101 < Q'(a,-) + Q'(-,b)$

c

> Can offsetting a scoring function by an additive constant influence the identity of the maximum score global alignment? If you think so, provide an explicit and detailed example to demonstrate this. If you think offsetting cannot influence the maximum score global alignment, then provide a proof of this claim.

Yes, the same example from before holds true,
We can notice that in the example from question 1_b, the global alignment and local alignment are equal. (In both cases of Q and Q')

Problem 2

a

> Explain shortly why the technique used for global alignment cannot be used **as is** in the case of local alignment.

The technique used in the NW algorithm assumes that the optimal alignment path must cross the middle section of the DP matrix, because it always begins at $(0,0)$ and ends at (m,n). In local alignment this assumption fails: the optimal solution may be completely contained in an internal sub-matrix, never touching the mid column or any boundary. Therefore, the divide-and-conquer strategy of Hirschberg cannot be applied as is.

---

b

> Suggest a linear-space version for the Smith-Waterman algorithm. Describe your algorithm in full detail.

The idea of our algorithm is to first locate the sub-matrix that contains the optimal local alignment, and then run NW with linear space only on that region.

---

1. Run SW while maintaining only the last 2 rows. Track the location (i,j) of the maximum cell and denote it by $(i_{\text{end}}, j_{\text{end}})$. This gives the bottom-right endpoint of the optimal local alignment.
2. Take $S_{0..i_{\text{end}}}$ and $T_{0..j_{\text{end}}}$, reverse both, and run SW again in the same linear-space fashion. Let (i,j) be the new maximum. Convert this to the original coordinates:

$$i_{\text{start}} = |S| - i + 1, \quad j_{\text{start}} = |T| - j + 1.$$

3. Consider $S_{i_{\text{start}}..i_{\text{end}}}$ and $T_{j_{\text{start}}..j_{\text{end}}}$, and run NW with Hirschberg's trick to recover the alignment in linear space.

---

c

> Prove the correctness of your suggested algorithm and analyze its complexity. You may assume correctness and use the complexity bounds proven for all algorithms taught in class.

Steps **1** and **2** correctly identify the sub-matrix containing the optimal local alignment, with path starting at $(i_{\text{start}}, j_{\text{start}})$ and ending at $(i_{\text{end}}, j_{\text{end}})$.

Step **1** is immediate: the maximum cell of the Smith–Waterman matrix is exactly the end-point of the optimal local alignment.

The key part is step **2**. After reversing both sequences, the same local alignment appears reversed, and the original starting point becomes the ending point of a local alignment in the reversed prefixes. Running SW again therefore reveals precisely where the optimal alignment must begin in the original orientation. Converting back from reversed indices yields $(i_{\text{start}}, j_{\text{start}})$.

Conceptually, this produces a matrix of the following form:

$$\begin{array}{cccccccc}
. & . & . & . & . & . & . & . \\
. & \text{start} & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . \\
. & . & . & . & \text{end} & . & . & . \\
. & . & . & . & . & . & . & .
\end{array}$$

Once both endpoints are known, the problem reduces to a global alignment inside this sub-region, and NW with Hirschberg reconstructs the alignment.

For the complexity: each SW pass takes $O(|S| \cdot |T|)$ time and only $O(|S| + |T|)$ space. The final NW reconstruction is also linear space. Thus, the entire algorithm runs in time

$$O(|S| \cdot |T|)$$

and space

$$O(|S| + |T|).$$

Problem 3

a

> Suggest a DP algorithm which takes a genome sequence, T, a paired-end read, $S = (S_1, l, S_2)$, and an alignment scoring function σ, and computes in $O(|S||T|)$ complexity a maximum score mapping.

Give T and $S = (S_1, l, S_2)$.

1. We will run global alignment with the first row being set to 0, as we can start from any point in T but we must align all of S_1 and S_2
2. For S_1, run SW on (S_1, T) and denote as H_{s1}. Then, we are interested to find in every $0 ... len(T)$ the best local alignment score that ends at $T[j]$. This array is the last row in H_{s1} meaning for the sequence S_1 the best local alignment for $T[1..j]$ ending in $T[j]$ is $H_{s1}[|S_1|, j]$
3. To enforce the that A_2 will start exactly l bases after, we will run SW on the reversed sequences S_2^{rev} and T^{rev}, noted as H_{s2}, and compute look at the same row storing the best local alignment score of S_2 that ended at $|T| - j + 1]$
4. We will couple every cell $H_{s1}[|S_1|, j]$ with the cell $H_{s2}[|S_2|, |T| - j - l]$.
5. From those couples we will choose the one with the highest sum.

Matrix Cell

- Holds both the score of the local alignment, with update formula of

$$H[i, 0] = 0$$

$$H[i, j] = \max\Big(0,\ H[i-1, j-1] + \sigma(S[i], T[j]),\ H[i-1, j] + \text{gap},\ H[i, j-1] + \text{gap}\Big)$$

  and track of the previous step that was taken beforehand

Each cell $H_{s2}[|S_2|, |T| - j - l]$ holds the global alignment for S_2^{rev} with suffix of $T^{rev}[0..|T| - j - l]$
When unreversing the sequences we get:
global alignment of S_2 with prefix of $T[j + l..|T|]$

b

> consider a scenario where the distance between alignments, l, is not known.

We reuse parts 1–3 from (a), computing H_{s1}, H_{s2}, $\max_{s1}$, $\max_{s2}$ as before, and add additional steps

1. Compute rolling max_{s1}, left arg_{s1} (prefix maxima of $\max_{s1}$), and rolling max_{s2}, right arg_{s2} (suffix maxima of $\max_{s2}$).
2. For each $k = 0 ... |T| - 1$, compute

$$\text{Score}(k) = \text{rolling max}_{s1}[k] + \text{rolling max}_{s2}[k+1]$$

   and take the maximal score.
3. Let k^* be the maximizing cut.
   Set

$$i^* = \text{left arg}_{s1}[k^*], \qquad j^* = \text{right arg}_{s2}[k^* + 1].$$

Using these indices, perform traceback in H_{s1} and H_{s2} as in part (a) to output A_1 and A_2.

Problem 4

Written solution

```
    0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  180
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
1    0   0   0   0   2   0   2   0   0   0   0   0   0   0   0   2   0   0   0
2    0   0   0   0   0   4   1   0   0   2   0   0   0   0   2   0   0   0   0
3    0   0   0   0   2   1   6   3   0   0   0   0   0   0   0   4   1   0   0
4    0   0   0   0   2   0   3   4   1   0   0   0   0   0   0   2   2   0   0
5    0   0   0   0   0   0   0   5   6   3   2   0   2   2   0   0   4   1   0
6    0   0   0   0   0   0   0   2   7   4   5   2   2   4   1   0   2   2   0
7    0   2   2   2   0   0   0   0   4   5   2   7   4   1   2   0   0   4   4
8    0   0   0   0   4   1   2   0   1   2   3   4   5   2   0   4   1   1   2
9    0   0   0   0   1   6   3   0   0   3   0   1   2   3   4   1   2   0   0
10   0   0   0   0   0   3   4   1   0   2   1   0   0   0   5   2   0   0   0
11   0   0   0   0   0   0   1   6   3   0   4   1   2   2   2   3   4   1   0
12   0   0   0   0   2   0   2   3   4   1   1   2   0   0   0   4   1   2   0
13   0   2   2   2   0   0   0   0   1   2   0   3   0   0   0   1   2   3   4
14   0   2   4   4   1   0   0   0   0   0   0   2   1   0   0   0   0   4   5
15   0   0   1   2   2   0   0   2   2   0   2   0   4   3   0   0   2   1   2
16   0   0   0   0   0   4   1   0   0   4   1   0   1   2   5   2   0   0   0
17   0   0   0   0   2   1   6   3   0   1   2   0   0   0   2   7   4   1   0
18   0   0   0   0   0   4   3   4   1   2   0   0   0   0   2   4   5   2   0
19   0   0   0   0   0   2   2   1   2   3   0   0   0   0   2   1   2   3   0
20   0   0   0   0   0   0   0   4   3   0   5   2   2   2   0   0   3   0   1
21   0   2   2   2   0   0   0   1   2   1   2   7   4   1   0   0   0   5   2
22   0   2   4   4   1   0   0   0   0   0   0   4   5   2   0   0   0   2   7
23   0   0   1   2   6   3   2   0   0   0   0   1   2   3   0   2   0   0   4
24   0   0   0   0   4   4   5   2   0   0   0   0   0   0   1   2   0   0   1
Best score: 7 at indexes: 6 8
First optimal alignment:
S: ATAAGG
T: ATA-GG
S idx: [0, 1, 2, 3, 4, 5]
T idx: [3, 4, 5, '-', 6, 7]
   0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 180

      .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
1   .  .  .  .  D  .  .  .  .  .  .  .  .  .  .  .  .  .  .
2   .  .  .  .  .  D  .  .  .  .  .  .  .  .  .  .  .  .  .
3   .  .  .  .  .  .  D  .  .  .  .  .  .  .  .  .  .  .  .
4   .  .  .  .  .  .  U  .  .  .  .  .  .  .  .  .  .  .  .
5   .  .  .  .  .  .  .  D  .  .  .  .  .  .  .  .  .  .  .
6   .  .  .  .  .  .  .  .  D  .  .  .  .  .  .  .  .  .  .
7   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
8   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
9   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
10  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
11  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
12  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
13  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

```
14  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
15  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
16  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
17  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
18  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
19  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
20  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
21  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
22  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
23  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
24  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Second optimal alignment (non-overlapping):
S: TGACCGTA
T: TG-CGGTA
S idx: [9, 10, 11, 12, 13, 14, 15, 16]
T idx: [8, 9, '-', 10, 11, 12, 13, 14]
```

d

For finding another alignment, we just looked for another max score of 7

```
S: TGACCGTA
T: TG-CGGTA
S idx: [9, 10, 11, 12, 13, 14, 15, 16]
T idx: [8, 9, '-', 10, 11, 12, 13, 14]
```

e

```
Alignment 1:
S: A T A A G G (0 - 5)
T: A T A - G G (3 - 7)

Alignment 2:
S: A T A A G G (0 - 5)
T: A T - A G G (3 - 7)
```

Those alignment are completely overlapping as the gaps can be either in the first A or the second A