CS466: Introduction to Bioinformatics **Name:**

Problem Set 2

Handed out: September 21, 2022 *Due: September 29, 2022*

Instructions: This homework assignment consists of three questions worth a total of 50 points. In addition, there is a bonus question worth an additional 5 points. These questions are based on the material covered in Lectures 6 to 9. **Do not forget to write your name at the top!**

1. **Subquadratic Time Alignment** [10 points]

   a. In class we learned how to solve the block alignment problem in time $O(n^2/\log n)$ using the Four Russians Technique. Specifically, in the case of an alphabet of $|\Sigma| = 4$ letters, we pre-computed *all* pairwise alignments of *all* strings of length $t = \log_2(n)/4$. Protein sequences have an alphabet of $|\Sigma| = 20$ letters. How should we choose length t to achieve the time bound of $O(n^2/\log n)$ if $|\Sigma| = 20$? Motivate your answer. [5 points]

$$t = \frac{\log_{20} n}{2} = \frac{\log_2 n}{2\log_2 20}$$

Reasoning For protein sequences our alphabet set $|\sum| = 20$ letters. So number of strings/sequences of length t that can be generated are

$$20^t = 20^{\frac{\log_{20} n}{2}} = \sqrt{n}$$

Now number of alignments to be precomputed is $20^t * 20^t = \sqrt{n} * \sqrt{n} = n$

Assuming each alignment takes $O(t^2)$ time, the total time required for precomputation step is $O(nt^2)$, that is given by

$$\begin{aligned} O(nt^2) &= O\left(n * \frac{\log_2 n}{2\log_2 20}^2\right) \\ &= O(n\log^2 n)\text{.....Ignoring the constant } \frac{1}{4\log^2 20} \end{aligned}$$

Now computing the optimal block alignment requires $\frac{n}{t} * \frac{n}{t}$ lookups and each lookup takes O(log n) time.
So total time for block alignment is

$$O\left(\frac{n}{\log n/2\log 20} * \frac{n}{\log n/2\log 20} * \log n\right) = O\left(\frac{n^2}{\log n}\right)$$

This running time dominates the time needed to precompute S (which took$O(nlog^2 n)$ time). Hence, we have a total running time of $O(n^2/logn)$.

Hence the t value given above helps to achieve the time bound of $O(n^2/\log n)$.

b. In their STOC 2015 paper, Backurs and Indyk proved that the edit distance problem cannot be solved in time $O(n^{2-\epsilon})$ where $\epsilon > 0$ under the Strong Exponential Time Hypothesis. To show that the same result also holds for pairwise global sequence alignment, choose an appropriate scoring function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R}$ such that an optimal edit distance alignment is also an optimal global sequence alignment. [5 points]

Bonus: +5 points if you give an actual proof of why the reduction works.

Let σ be the scoring function of the edit distance problem, and δ be the score matrix of the global alignment problem.

If $\delta(x, y) = -\sigma(x, y)$ for all $x, y \in \sum \bigcup \{-\}$, then the solution to the edit distance problem is equivalent to the solution to the global alignment problem.

Since we know that the , edit distance[Levenshtein distance] uses +1 for insertion, deletion and mismatch and 0 for match.
To obtain same global alignment, $\delta(x, y)$ ={-1 for indels, mismatch and 0 for match }.
This is obtained by fact that maximising a function is equivalent to minimising the negative of a function.
.

2. **Carrillo-Lipman** [20 points]

We consider the WEIGHTED SP-EDIT DISTANCE problem, where we are given sequences $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \Sigma^*$ each with length n and a scoring function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \to \mathbb{R}$. The task is to find a multiple alignment A such that $\mathrm{SP}(A)$ is minimum. We use the Carrillo-Lipman algorithm. Let $\mathbf{v}_{i,j}$ denote the prefix $v_{i,1} \ldots v_{i,j}$ of sequence $\mathbf{v}_i$ of length j. Briefly, $D(i_1, \ldots, i_k)$ denotes the minimum cost of aligning the k prefixes $\mathbf{v}_{1,i_1}, \ldots, \mathbf{v}_{k,i_k}$. On the other hand, $D^+_{a,b}(i,j)$ denotes the minimum cost of the pairwise alignment of suffixes $\mathbf{v}_{a,i}$ and $\mathbf{v}_{b,j}$. In Lecture 7, we considered the $k = 3$ case. We learned that given a heuristic solution with cost z, we know that the optimal alignment does *not* pass through vertex (i_1, i_2, i_3) if

$$D(i_1, i_2, i_3) + D^+_{1,2}(i_1, i_2) + D^+_{1,3}(i_1, i_3) + D^+_{2,3}(i_2, i_3) > z. \tag{1}$$

a. Consider the case with $k = 4$ sequences. Let $(i_1, i_2, i_3, i_4) \in [n]^4$ and let $D(i_1, i_2, i_3, i_4)$ be the optimal cost for aligning prefixes $\mathbf{v}_{1,i_1}, \ldots, \mathbf{v}_{4,i_4}$. Let z be the cost of an alignment of $\mathbf{v}_1, \ldots, \mathbf{v}_4$. Under which condition do we know that the optimal alignment does *not* pass through vertex (i_1, i_2, i_3, i_4)? [5 points]

Hint: Update Equation (1).

$$\begin{aligned} D(i_1, i_2, i_3, i_4) + D^+_{1,2}(i_1, i_2) + D^+_{1,3}(i_1, i_3) + D^+_{1,4}(i_1, i_4) \\ +D^+_{2,3}(i_2, i_3) + D^+_{2,4}(i_2, i_4) + +D^+_{3,4}(i_3, i_4) > z. \end{aligned} \tag{2}$$

.

b. Consider the general case with $k \in \mathbb{N}$ sequences. Let $(i_1, \ldots, i_k) \in [n]^k$ and let $D(i_1, \ldots, i_k)$ be the optimal cost for aligning prefixes $\mathbf{v}_{1,i_1}, \ldots, \mathbf{v}_{k,i_k}$. Let z be the cost of an alignment of $\mathbf{v}_1, \ldots, \mathbf{v}_k$. Under which condition do we know that the optimal alignment does *not* pass through vertex $(i_1, \ldots, i_k)$? [10 points]

Hint: Update Equation (1).

The optimal alignment does not pass through vertex the$(i_1, i_2, ..., i_k)$ if following holds true

$$D(i_1, \ldots, i_k) + \sum_{p=1}^{k} \sum_{q=p+1}^{k} D^+_{p,q}(i_p, i_q) > z$$

.

c. Finally, consider the SP-Global Alignment problem, where we have the same input $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \Sigma^*$ but aim to find an alignment A with *maximum* score SP(A). Suppose that we are given $k = 3$ sequences. Let z be the score of an alignment of $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$. Under which condition do we know that the optimal alignment does *not* pass through vertex (i_1, i_2, i_3)? [5 points]

(i_1, i_2, i_3) doesn't pass through the optimal alignment if it satisfies following equation

$$D(i_1, i_2, i_3) + D^+_{1,2}(i_1, i_2) + D^+_{1,3}(i_1, i_3) + D^+_{2,3}(i_2, i_3) < z$$

Here $D(i_1, i_2, i_3)$ is the *maximum* SP-cost of aligning prefixes v1[1..i], v2[1..j], v3[1..k]

$D_{p,q}(i, j)$ be the maximum cost of aligning prefixes vp[1..i], vq[1..j](where $1 \leq p < q \leq 3$).

$D^+_{p,q}(i, j)$ be the maximum cost of aligning suffixes vp[i..n], vq[j..n] (where1 $\leq p < q \leq 3$).

.

3. **Star Alignment** [20 points]

Create a star alignment of the four strings s_1, s_2, s_3, s_4 using the pre-computed optimal pairwise alignments provided below. Present your answer by identifying the center sequence s_c, the quantity $\sum_{i=1}^{4} d(s_c, s_i)$ and the final multiple alignment of the strings. **Show your work.** Include all intermediate multiple alignments that you generate.

s_1 :	`ACCCTCGCT`
s_2 :	`ACGGTCCCT`
s_3 :	`ACGGCCT`
s_4 :	`TCGGCCCTT`

s_1 :	`ACCCTCGCT`	s_1 :	`ACCCTCGCT`	s_1 :	`AC--CCTCGCT`
s_2 :	`ACGGTCCCT`	s_3 :	`A-CGGC-CT`	s_4 :	`TCGGCC-C-TT`
$d(s_1, s_2) =$	3	$d(s_1, s_3) =$	4	$d(s_1, s_4) =$	6
s_2 :	`ACGGTCCCT`	s_2 :	`ACGGTCCC-T`	s_3 :	`ACGG-CC-T`
s_3 :	`ACGG--CCT`	s_4 :	`TCGG-CCCTT`	s_4 :	`TCGGCCCTT`
$d(s_2, s_3)$:	2	$d(s_2, s_4) =$	3	$d(s_3, s_4) =$	3

Hint: This is an instance of SP-Edit Distance.

Given $d(v_i, v_j)$ is the optimal (weighted) edit distance between v_i and v_j, we know that $d(v_j, v_i) = d(v_i, v_j)$.
Also $d(v_i, v_i)$ =0

(a) Let s_1 be center star.

$$\begin{aligned}\sum_{i=1}^{4} d(s_1, s_i) &= d(v_1, v_1) + d(v_1, v_2) + d(v_1, v_3) + d(v_1, v_4)\\ &= 0 + 3 + 4 + 6\\ &= 13\end{aligned}$$

(b) Let s_2 be center star.

$$\begin{aligned}\sum_{i=1}^{4} d(s_2, s_i) &= d(v_2, v_1) + d(v_2, v_2) + d(v_2, v_3) + d(v_2, v_4)\\ &= 3 + 0 + 2 + 3\\ &= 8\end{aligned}$$

(c) Let s_3 be center star.

$$\begin{aligned}\sum_{i=1}^{4} d(s_3, s_i) &= d(v_3, v_1) + d(v_3, v_2) + d(v_3, v_3) + d(v_3, v_4)\\ &= 4 + 2 + 0 + 3\\ &= 9\end{aligned}$$

(d) Let s_4 be center star.

$$\begin{aligned}\sum_{i=1}^{4} d(s_4, s_i) &= d(v_4, v_1) + d(v_4, v_2) + d(v_4, v_3) + d(v_4, v_4)\\ &= 6 + 3 + 3 + 0\\ &= 12\end{aligned}$$

.

The minimum sum is obtained when s_2 is the center sequence. So we will use $s_c = s_2$

Aligning s_2 with s_1

```
s2 :  ACGGTCCCT
s1 :  ACCCTCGCT
```

Aligning s_3 with earlier obtained alignment

```
s2 :  ACGGTCCCT
s1 :  ACCCTCGCT
s3 :  ACGG-CC-T
```

Aligning s_4 to obtain final alignment with earlier obtained alignment.

```
s2 :  ACGGTCCC-T
s1 :  ACCCTCGC-T
s3 :  ACGG--CC-T
s4 :  TCGG-CCCTT
```