CS466: Introduction to Bioinformatics Name: Payal Mantri

Problem Set 1

Handed out: September 7, 2022 *Due: September 15, 2022 (11:59pm, CT)*

Instructions: This homework assignment consists of five questions worth a total of 50 points. In addition, there is a bonus question on the third page worth an additional 6 points. These questions are based on the material covered in Lectures 1 to 5. **Do not forget to write your name at the top!**

1. **Asymptotic Running Time** [5 points]

Consider the following running time functions, where $n > 0$.

$$\begin{array}{cccccccc} n^2 & n^3 & \sqrt{n} & n^2\log(n) & n\log(n) & n! & 2^n & n \\ n(n+1)-n^2 & n+n^2 & n\log(n^2) & n^3-n^2 & 1 & n^2-n & n^n & 10{,}000{,}000 \end{array}$$

a. Identify groups of functions such that for any pair $(f(n), g(n))$ of functions in the same group it holds that both $f(n) = O(g(n))$ and $g(n) = O(f(n))$. Note that some groups will contain a single function. [3 points]

Hint: For example, $f(n) = 3n$ and $g(n) = n$ would be in the same group, as $f(n) = 3n = O(n) = O(g(n))$ and $g(n) = n = O(3n) = O(f(n))$.

- g_1(x) = {1 , 10000000}
- g_2(x) = $\{\sqrt{n}\}$
- g_3(x) = $\{n \text{ , } n(n+1) - n^2 \}$
- g_4(x) = $\{n\log(n) \text{ , } n\log(n^2) \}$
- g_5(x) = $\{n^2 \text{ , } n+n^2 \text{ , } n^2 - n \}$
- g_6(x) = $\{n^2\log(n) \}$
- g_7(x) = $\{n^3 \text{ , } n^3 - n^2 \}$
- g_8(x) = $\{ 2^n \}$
- g_9(x) = $\{n! \}$
- g_{10}(x) = $\{n^n \}$

.

b. Arrange the resulting Big Oh running time groups in order from fastest to slowest. [2 points]

g_1 , g_2, g_3, g_4, g_5, g_6 , g_7, g_8, g_9, g_{10} .

2. **Sequence Alignment** [20 points]

Consider two DNA sequences **v** = TAGATA and **w** = GTAGGCTTAAGGTTA. In this exercise, we will align the two sequences using a score of +1 for a match, -1 for a mismatch, and -1 for a insertion/deletion (i.e. a gap penalty of 1). We will use three different alignment algorithms. In each case, follow the specific instructions to provide requested information about the dynamic programming table or optimal alignment.

a. Consider the following global alignment of **v** with **w**.

v	-	T	A	G	-	-	-	-	A	-	-	-	T	-	A
w	G	T	A	G	G	C	T	T	A	A	G	G	T	T	A

Give the score for this global alignment and fill out the dynamic programming table with the corresponding backtrace (i.e. highlight the corresponding path through this table and fill in cells on path with alignment scores). [5 points]

	-	G	T	A	G	G	C	T	T	A	A	G	G	T	T	A
-	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15
T	-1	-1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13
A	-2	-2	-1	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
G	-3	-1	-2	0	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
A	-4	-2	-2	-1	1	1	0	-1	-2	-1	-2	-3	-4	-5	-6	-7
T	-5	-3	-1	-2	0	0	0	1	0	-1	-2	-3	-4	-3	-4	-5
A	-6	-4	-2	-2	-1	-1	-1	0	2	1	0	-1	-2	-3	-4	-3

Alignment score: -3

b. Consider the following fitting alignment (grayed out entries are not part of the alignment). That is, an alignment of **v** and a substring of **w** with maximum global alignment score.

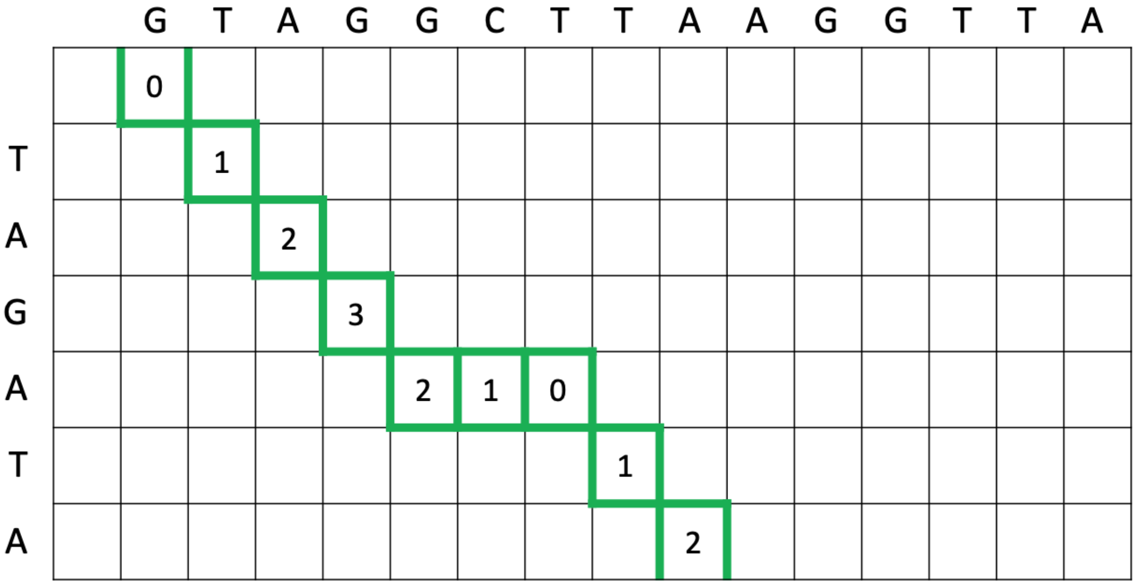
v	-	-	-	-	-	-	-	T	A	-	G	A	T	-	A
w	G	T	A	G	G	C	T	T	A	A	G	G	T	T	A

Give the score for this fitting alignment and fill out the dynamic programming table with the corresponding backtrace (i.e. highlight the corresponding path through this table and fill in cells on path with alignment scores). [5 points]

	-	G	T	A	G	G	C	T	T	A	A	G	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	-1	-1	1	0	-1	-1	-1	1	1	0	-1	-1	-1	1	1	0
A	-2	-2	0	2	1	0	-1	0	0	2	1	0	-1	0	0	2
G	-3	-1	-1	1	3	2	1	0	-1	1	1	2	1	0	-1	1
A	-4	-2	-2	0	2	2	1	0	-1	0	2	1	1	0	-1	0
T	-5	-3	-1	-1	1	1	1	2	1	0	2	1	0	2	1	0
A	-6	-4	-2	0	0	0	0	1	1	2	1	0	0	1	1	2

Alignment score: 2

c. Consider the following dynamic programming table produced when finding an optimal fitting alignment. That is, an alignment of **v** and a substring of **w** with maximum global alignment score.



		G	T	A	G	G	C	T	T	A	A	G	G	T	T	A
		0														
T			1													
A				2												
G					3											
A						2	1	0								
T									1							
A										2						

Give the fitting alignment corresponding to the highlighted path. [5 points]

Corresponding alignment:

-	T	A	G	A	-	-	T	A	-	-	-	-	-	-
G	T	A	G	G	C	T	T	A	A	G	G	T	T	A

d. Consider the following local alignment (grayed out entries are not part of the alignment). That is, an alignment of a substring of **v** and a substring of **w** with maximum global alignment score.

v	-	T	A	G	-	-	-	-	-	-	-	-	-	-	-
w	G	T	A	G	G	C	T	T	A	A	G	G	T	T	A

Give the score for this local alignment and fill out the dynamic programming table with the corresponding backtrace (i.e. highlight the corresponding path through this table and fill in cells on path with alignment scores). [5 points]

	-	G	T	A	G	G	C	T	T	A	A	G	G	T	T	A
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0
A	0	0	0	2	0	0	0	0	0	2	1	0	0	0	0	2
G	0	1	0	0	3	1	0	0	0	1	0	2	1	0	0	1
A	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0
T	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0
A	0	0	0	2	0	0	0	0	0	2	1	0	0	0	0	2

Alignment score: 3

3. **Linear Space Alignment** [10 points]

Consider two sequences $\mathbf{v}$ = CT and $\mathbf{w}$ = GCAT of length $m = |\mathbf{v}| = 2$ and $n = |\mathbf{w}| = 4$, respectively. In this exercise, we will compute an optimal global alignment of the two sequences using the Hirschberg algorithm. We will use a score of +1 for a match, -1 for a mismatch, and -1 for a insertion/deletion (i.e. a gap penalty of 1).

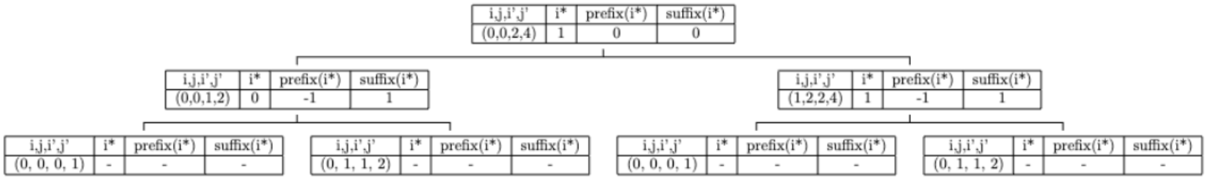
a. The initial call is Hirschberg$(0, 0, m = 2, n = 4)$. We need to identify the middle vertex $(i^*, n/2 = 2)$. Fill out the following table for this initial call and indicate i^*. [2 points]

i	prefix(i)	suffix(i)	wt(i)
0	-2	0	-2
1	0	0	0
2	-1	-2	-3

b. What are the two recursive calls that are made in this initial invocation Hirschberg$(0, 0, m, n)$? [1 point]

Hirschberg(0, 0 , 1, 2) and
Hirschberg(1, 2, 2, 4)

c. Give the recursion tree, where each vertex corresponds to an invocation of Hirschberg. See Lecture 1 for an example of a recursion tree. Label each vertex of this tree by the used arguments (i, j, i', j'). In addition, label each *internal* vertex by the value of i^*, prefix(i^*) and suffix(i^*). [5 points]


i,j,i',j' | i* | prefix(i*) | suffix(i*)
(0,0,2,4) | 1 | 0 | 0
i,j,i',j' | i* | prefix(i*) | suffix(i*)
(0,0,1,2) | 0 | -1 | 1
i,j,i',j' | i* | prefix(i*) | suffix(i*)
(1,2,2,4) | 1 | -1 | 1
i,j,i',j' | i* | prefix(i*) | suffix(i*)
(0, 0, 0, 1) | - | - | -
i,j,i',j' | i* | prefix(i*) | suffix(i*)
(0, 1, 1, 2) | - | - | -
i,j,i',j' | i* | prefix(i*) | suffix(i*)
(0, 0, 0, 1) | - | - | -
i,j,i',j' | i* | prefix(i*) | suffix(i*)
(0, 1, 1, 2) | - | - | -

d. Indicate the reported vertices in the table and give the final alignment. [2 points]

Reported vertices (‘X’):

	0	G	C	A	T
0	X	X			
C			X	X	
T					X

Final alignment:

-	C	-	T
G	C	A	T

4. **Banded Alignment** [5 points]

Global pairwise sequence alignment of two sequences of length m and n takes $O(mn)$ time using dynamic programming. This corresponds to quadratic time. In this question, we consider banded alignment, which was introduced in Lecture 5. Briefly, the idea is to restrict alignments, which are paths from $(0, 0)$ to (m, n), to only occur in a band of width k around the diagonal. Modify the recurrence of global sequence alignment to achieve this change. Assume $m = n$ and use 0-based indexing, i.e. with $k = 0$ the allowed region consists of only the diagonal, $k = 1$ has an allowed region of length 3, etc. [5 points]

a coordinate will be present in the band only if it satisfies following condition
$|i - j|$=k

The recurrence can be calculated as :-

$$s[i,j] = \max \begin{cases} 0, & \text{if i=0, j=0,} \\ s[i-1,j] + \delta(v_i, -), & \text{if j} \leq \text{i+k} \\ s[i,j-1] + \delta(-, w_j), & \text{if j} \geq \text{i-k} \\ s[i-1,j-1] + \delta(v_i, w_j), & \text{otherwise} \end{cases}$$

5. **BLOSUM** [10 points]

Consider the following three blocks on the alphabet $\Sigma = \{A, B, C, D\}$.

```
ABCDA      BBC     AAAA
ABCDA      BBC     DBBB
BBCDA      BCC     BAAA
AACDA      CBC     ADBA
CBADA      BBD
AACAA
```

Using $L = 0$, such that the above three blocks are not pruned down, compute the BLOSUM0 scoring matrix. Use $\lambda = 0.5$, the natural logarithm and round up to the nearest integer (i.e. take the ceiling). **Give q_x and $p_{x,y}$ for each $x, y \in \Sigma$. Clearly indicate the denominator used for computing these two quantities.**

q_A= 23/61 = 0.377
q_B= 18/61 = 0.295
q_C= 12/61 = 0.196
q_D=8/61 =0.131

$q_{A,A}$= 22/129 = 0.170
$q_{A,B}$= 23/129 =0.178
$q_{A,C}$= 9/129 = 0.069
$q_{A,D}$= 9/129 = 0.069
$q_{B,B}$= 19/129 =0.147
$q_{B,C}$= 9/129 = 0.069
$q_{B,D}$= 2/129 = 0.016
$q_{C,C}$= 16/129 = 0.124
$q_{C,D}$= 4/129 = 0.031
$q_{D,D}$= 10/129 = 0.077

$$s(A, B) = \frac{1}{\lambda} \ln \frac{p_A p_B}{q_{A,B}}$$

The BLOSUM scoring Matrix is as follows

$$\begin{array}{ccccc}
A & 1 & & & \\
B & 1 & 2 & & \\
C & 0 & 1 & 3 & \\
D & 1 & -1 & 1 & 4 \\
 & A & B & C & D
\end{array}$$

.

6. **Bonus: Total Number of Global Alignments** [6 points]

In this bonus question, we are going to determine the total number of *global alignments* that exist given two strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$. We will assume without loss of generality that $m \leq n$. Recall the matrix representation of an alignment. This is a $2 \times k$ matrix where $k \in \{\max\{m, n\}, \ldots, m + n\}$ such that there is no column with two gaps. Thus, the number k of columns varies from $\max\{m, n\}$ to $m + n$.

a. Explain why, in general (i.e. no prior knowledge on how m and n are related), the number k of columns varies from $\max\{m, n\}$ to $m + n$. [1 point]

Since alignment is way to convert one sequence to another, it needs minimum number of spaces to accommodate the sequence of larger length, which is given by **max{m,n}**.
Also we say that there shouldn't be any column with two gaps, and in the worst case, when no letters in v and w match, we need to delete all elements of sequence v and insert all letters of sequence w, which will need a space of m+n letters (n gaps in row 1 corresponding to n inserts and m gaps in row 2 corresponding to n deletes). Hence **maximum** number of columns required will be **m+n**
Thus, the number k of columns varies from **max{m,n}** to **m+n**
.

b. Explain why $k \in \{n, \ldots, m + n\}$ for the case where $m \leq n$. [1 point]

Using the above explanation in 6b , we know that the number k of columns varies from **max{m,n}** to **m+n**.
If m ≤ n , then **max { m,n }** = n .
Hence in that case the number of k of columns varies from **n** to **m+n** , that is k ∈{ $n, ..., m + n$} .

c. Suppose that the alignment has length $k \geq n$. In how many different ways can we insert $k - n$ gaps in the second sequence $\mathbf{w}$, yielding gapped sequence $\mathbf{w}'$? [1 point]

Hint: Observe that $\mathbf{w}'$ has length k.

kC (k-n) = kC n .

d. Let $\mathbf{w}'$ be a gapped sequence of length $k \in \{n, \ldots, m + n\}$ such that removing the gaps yields the original sequence $\mathbf{w}$. In how many ways can we insert gaps in $\mathbf{v}$ to obtain an alignment with $\mathbf{w}'$ of length k? [1 point]

Hint: Recall that an alignment does not contain columns with two gaps. In how many different ways can we insert gaps in $\mathbf{v}$ subject to this condition?

.

e. How many alignments of $\mathbf{v}$ and $\mathbf{w}$ are there of a given length $k \in \{n, \ldots, m+n\}$? How many alignments are there of any length? [1 point]

Hint: Combine your answers to the previous two questions.

Number of alignments of of any length

$$\binom{m+n}{n} = \frac{(m+n)!}{m!n!}$$

f. Give an example of a scoring function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \to \mathbb{R}$ such that the number of optimal global alignments equals your answer to the previous question. [1 point]

Hint: Think of a border case.

.