Midterm Practice Exam

CS 4390/5390

October 13, 2019

1. Given the table below which was created using the Smith-Waterman algorithm for local alignment, (a) identify the local alignment score, and (b) perform trace-back to find the optimal alignment.

		T	T	A	C	T	G	T	G	T
	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	↖5	←4.5	←4	←3.5	←3	←2.5
A	0	0	0	↖5	←↑4.5	↖←↑4	↖←↑3.5	↖←↑3	↖←↑2.5	↖←↑2
C	0	0	0	↑4.5	↖10	←9.5	←9	←8.5	←8	←7.5
C	0	0	0	↑4	↖↑9.5	↖←↑9	↖←↑8.5	↖←↑8	↖←↑7.5	↖←↑7
C	0	0	0	↑3.5	↖↑9	↖←↑8.5	↖←↑8	↖←↑7.5	↖←↑7	↖←↑6.5
C	0	0	0	↑3	↖↑8.5	↖←↑8	↖←↑7.5	↖←↑7	↖←↑6.5	↖←↑6
T	0	↖5	↖5	←4.5	↑7.5	↖13.5	←13	↖←12.5	←12	↖←11.5
G	0	↑4.5	↑4.5	↖←↑4	↑7	↑13	↖18.5	←18	↖←17.5	←17
T	0	↖5	↖9.5	←9	←8.5	↑12.5	↑18	↖23.5	[illegible]	↖←22.5
G	0	↑4.5	↑9	↖←↑8.5	↖←↑8	↑12	↖↑17.5	↑23	↖28.5	←28

Optimal Local Alignment Score:

Optimal Local Alignment (note not all of the spaced will be used)

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

2. Given the Needleman-Wunsch table below, find the optimal global alignment for the two sequences.

		T	T	A	C	T	G	T	G	T
	0	←-0.5	←-1	←-1.5	←-2	←-2.5	←-3	←-3.5	←-4	←-4.5
C	↑-0.5	↖←↑-1	↖←↑-1.5	↖←↑-2	↖3.5	←3	←2.5	←2	←1.5	←1
A	↑-1	↖←↑-1.5	↖←↑-2	↖3.5	←↑3	↖←↑2.5	↖←↑2	↖←↑1.5	↖←↑1	↖←↑0.5
C	↑-1.5	↖←↑-2	↖←↑-2.5	↑3	↖8.5	←8	←7.5	←7	←6.5	←6
C	↑-2	↖←↑-2.5	↖←↑-3	↑2.5	↖↑8	↖←↑7.5	↖←↑7	↖←↑6.5	↖←↑6	↖←↑5.5
C	↑-2.5	↖←↑-3	↖←↑-3.5	↑2	↖↑7.5	↖←↑7	↖←↑6.5	↖←↑6	↖←↑5.5	↖←↑5
C	↑-3	↖←↑-3.5	↖←↑-4	↑1.5	↖↑7	↖←↑6.5	↖←↑6	↖←↑5.5	↖←↑5	↖←↑4.5
T	↑-3.5	↖2	↖←1.5	←↑1	↑6.5	↖12	←11.5	↖←11	←10.5	↖←10
G	↑-4	↑1.5	↖←↑1	↖←↑0.5	↑6	↑11.5	↖17	←16.5	↖←16	←15.5
T	↑-4.5	↖↑1	↖6.5	←6	←↑5.5	↖↑11	↑16.5	↖22	←21.5	↖←21
G	↑-5	↑0.5	↑6	↖←↑5.5	↖←↑5	↑10.5	↖↑16	↑21.5	↖27	←26.5

Optimal Global Alignment (note not all of the spaced will be used)

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

3. (a) Compute the Z-Values for `ACTAACTAAC`. (b) how are the values of $Z_2, Z_3, ...Z_{i-1}$ used in computing Z_i. (c) what does the value of Z_i mean?

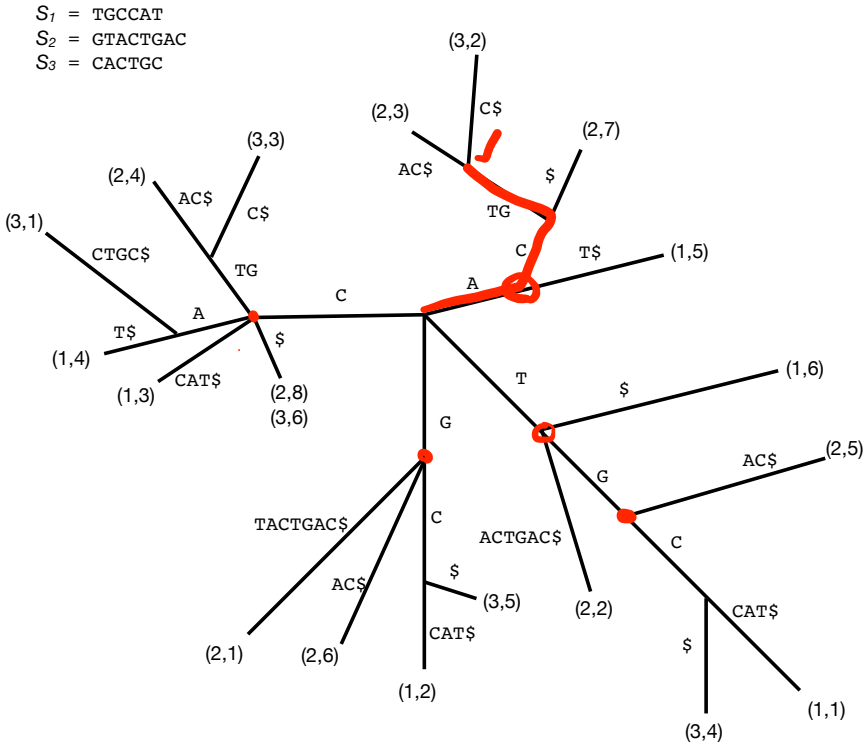
(a)

A C T A A C T A A C
Zi - 0 0 1 6 0 0 1 2 0

(b) When computing say Z6, we know Z5 + 5 > 6, so we can look to see if Z2 is less than Z5 + 1, if so we know that the prefix match is contained in the current Z-box (starting at 5) and that the value can simply be copied.

(c) The Z value is the longest prefix of S[1…n] and S[i…n] that match.

4. From the suffix tree below: (a) determine if the string `ACTG` is in the input set of sequences, and explain your reasoning; and (b) find the longest common substring between the set of sequences, and explain your reasoning.



(a) Yes, ACTG is contained in the set because the path from the root following that sequence (highlighted) exists in the suffix tree.

(b) "TG" is the longest common substring, of the internal nodes in the tree with leaves in their subtrees labeled by all 3 sequences (circled), the node representing the string "TG" is the deepest.

7. What is the sum-of-pairs score of the following multiple sequence alignment using the global scoring with affine scoring model with the following parameters:

match	10
mismatch	-3
indel	-1
gap	-3

```
ACCTGCC
-C-TGCA
AGCGGCA
ACCT--A
```

Mt 3 3 3 3 3 3 3 = 21
Ms 0 3 0 3 0 0 3 = 9
Id 3 0 3 0 3 3 0 = 12
Gp = 9

(10 x 21) - (3 x 9) - (1 x 12) - (3 x 9)

210 - 27 - 12 - 27

210 - 66

144

8. Given the pairwise alignments between the 4 sequences, and using sequence B as the star-center, create the multiple alignment using the center-star method.

A: GATG-TGCCG B: CCTGCTGCAG	B: CCTGCT-GCAG C: CC-GCTAGCAG	B: CCTGCT-GCAG D: CCTG-TAG--G

```
B:   CCTGCT-GCAG
A:   GATG-T-GCCG
C:   CC-GCTAGCAG
D:   CCTG-TAG--G
```

9. How would we modify the Smith-Waterman algorithm if we wanted to find a disjoint set of substrings of S to align to a substring of T.

   For example when aligning S =GGAGCGGCTTGG with T =AAAACCTTTT, an optimal alignment would align $S[3..5] \cdot S[8..10]$ to $T[3...8]$:

   AGCCTT
   AACCTT.

   The concept can be though of as "skipping" $S[6..7]$ when computing the optimal local alignment. Note that the $\cdot$ operator is for concatenation.

Update the recursion formula to the following:

```
V(i,j) = max {
      0,                                 // this is local alignment, empty align okay
      V(i-1,j-1) + delta(S[i],S[j]) // match mismatch as normal
      V(i, j-1) + delta(-, T[j])      // all insertions are still counted
      V(i-k, j), k<i                     // look for all substrings that ended at j in T
}
```

during the traceback follow any jumps to reconstruct the alignment

3. (2 point) Given the following partially completed computation of the Z-value algorithm, compute the rest of the values using the $O(n)$ time algorithm we discussed in class. Describe how you arrived at each value.

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

copied Z_2

copied Z_3

copied Z_5

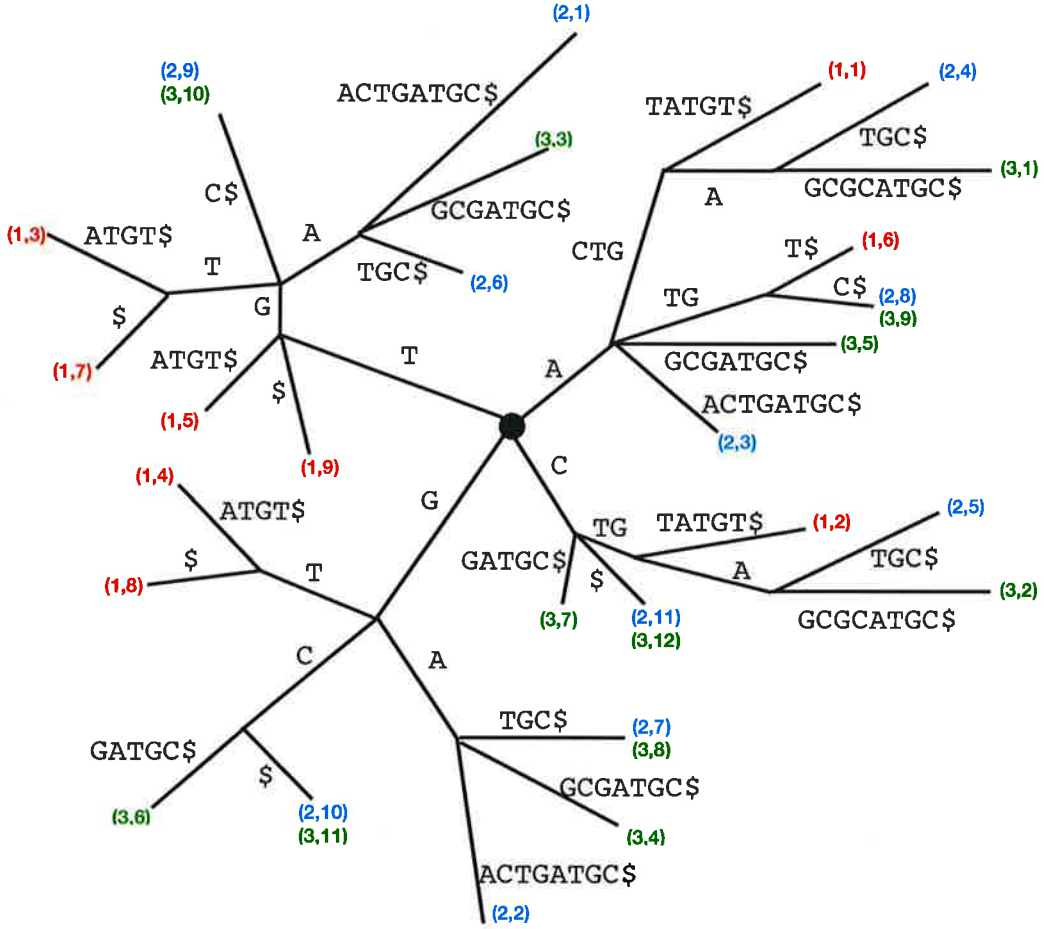
compare $S[i4] = S[1]$

compare ~~[illegible]~~ $S[13] = S[1]$

$Z_4 > 5-3$, so check $S[13] = S[3]$.

it does not so true value is 2

4. (3 points) From the suffix tree below: (a) determine if the string ACTG is in the input set of sequences, and explain your reasoning; (b) find the longest substring that occurs in all of the sequences ***twice***, and explain your reasoning; (c) list the missing suffix links.



(a) yes, a path from the root labeled "A", "CTG" exists

(b) "TG", deepest of "G", "TG", "T", "A"

(c) ACTG → CTG
ACTGA → CTGA
ATG → TG
CTG → TG
CTGA → TGA
GA → A
GC → C
GT → T
TG → G
TGA → GA

10. (2 points) How would we modify the Needleman-Wunsch algorithm if we wanted to allow for any character in S to be repeated aligned as many times as we want in place.

    For example when aligning S =AGA with T =GGGGGA, an optimal alignment would repeat the G in S 5 times to give the alignment:

    ```
    AGGGGGA
    -GGGGGA
    ```

    In reality, the middle G is is being aligned with all of the Gs in T.

modify the recurrance by adding an extra term

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + \delta(S[i],T[j]) \\ V(i-1,j) + \delta(S[i],\text{'-'}) \\ V(i,j-1) + \delta(\text{'-'},T[j]) \\ V(i,j-1) + \delta(S[i],T[j]) \end{cases}$$

allows for the best match/mismatch from last char of S.

On backtrack, follow links that may move non-diagonally but still output a column w/ two characters.