Final Exam

CS 4390/5390

10 December 2019

Please put your name on each page of this test. Please keep answers on their own pages. While hand written and scanned responses are acceptable, typed responses are preferred as they are more clear and have more opportunities for partial credit. Remember: read all questions first, you don't need to complete them in the order presented.

The exam will be released on Monday 10 December 2019 at 1:00pm MST and will be due at 3:45pm MST. Questions during the exam can be addressed either via email or Skype (`danfdeblasio`).

This assessment is open book and open computer, but you are not allowed to communicate with other people either inside or outside of this class (other than the instructor).

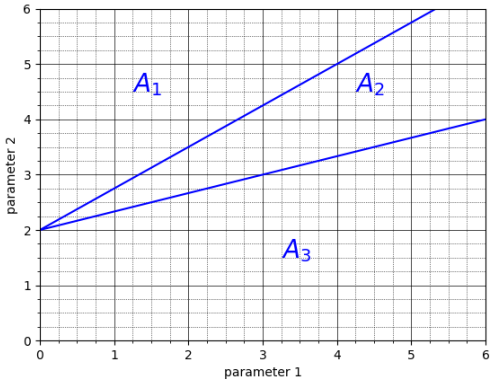
By tuning in this exam you are certifying that you did not have assistance from anyone else on this exam, that you did not help anyone else, and that if you know of or come to learn of someone who did gain assistance from another person you will inform the instructor immediately.

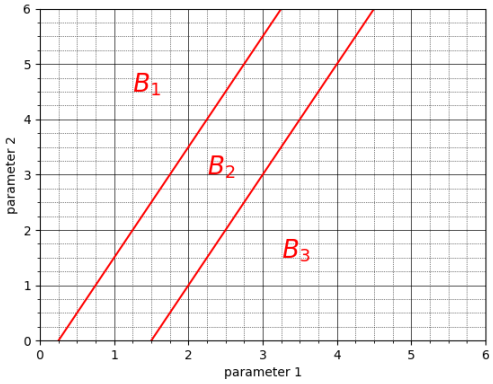
Problem	Score
1	/5
2	/6
3	/5
4	/3
5	/2
6	/3 (+1ec)
7	/1
Total	/25

1. (5 points) (a) Calculate the accuracies for the following groups of alignments, the reference alignment is provided at the top of each column. (b) Given the accuracies and the parameter decompositions shown in the figures, what is the region of the parameter space (identify the corners of the polygon) that provides the best alignments on average across these two pairs of sequences.

ATG-CTGGAT -TGA-TCGAT	TTGTGTCC-- TT-T-TCCAA
ATG-CT-GGAT -TGA-TC-GAT	TTGTGTCC TTTTCCAA
ATGCTGGAT -TGATCGAT	TTGTGTCC-- TTTT--CCAA
ATG--CTGGAT -TGATC--GAT	TTGTGTCC-- --TTTTCCAA

	A_1	A_2	A_3	B_1	B_2	B_3
Accuracies:	6/7	7/7	5/7	2/6	4/6	3/6





Optimal Polygon:
$(1.5, 2.5)$
$(2, 3.5)$
$(4, 5)$
$(3, 3)$

2. (6 points) (a) Construct the Burrows-Wheeler Indecies (BWT and C arrays) for the strings $S_1 =$ `ATGAT` and $S_2 =$ `TCTGAA`. (b) Using the provided pseudo-code outline, design an algorithm that when given two BWT Indices finds the length of the longest common substring. (c) Provide the output when applying your algorithm to the BTWs created in part (a), (notice the 3 print statements).

Algorithm 1: given two BWT indexes (L_1, C_1) and (L_2, C_2), defined as global variables.

Function `FindLongRep`(sp_1, ep_1, sp_2, ep_2)**:**
  print "(" $\cdot$ sp_1 $\cdot$ "," $\cdot$ ep_1 $\cdot$ "," $\cdot$ sp_2 + "," $\cdot$ ep_2 $\cdot$ ")\n"
  $itMax =$""
  for $c \in \Sigma$ **do**
    $spc_1 = C_1[c] + \text{rank}_c(L_1, sp_1 - 1) + 1$
    $epc_1 = C_1[c] + \text{rank}_c(L_1, ep_1)$
    $spc_2 = C_2[c] + \text{rank}_c(L_2, sp_2 - 1) + 1$
    $epc_2 = C_2[c] + \text{rank}_c(L_2, ep_2)$
    if $epc_1 \geq spc_1$ ***and*** $epc_2 \geq spc_2$ **then**
      $val =$ `FindLongRep`$(spc_1, epc_1, spc_2, epc_2) \cdot c$
      print "found a match of " $\cdot$ val
      $itMax = \underset{x \in \{itMax, val\}}{\text{argmax}} \{|x|\}$
    end
  end
  return $itMax$
print "The longest common sequence is " $\cdot$ `FindLongRep`$(1,|S_1|,1,|S_2|) \cdot$ " \n"

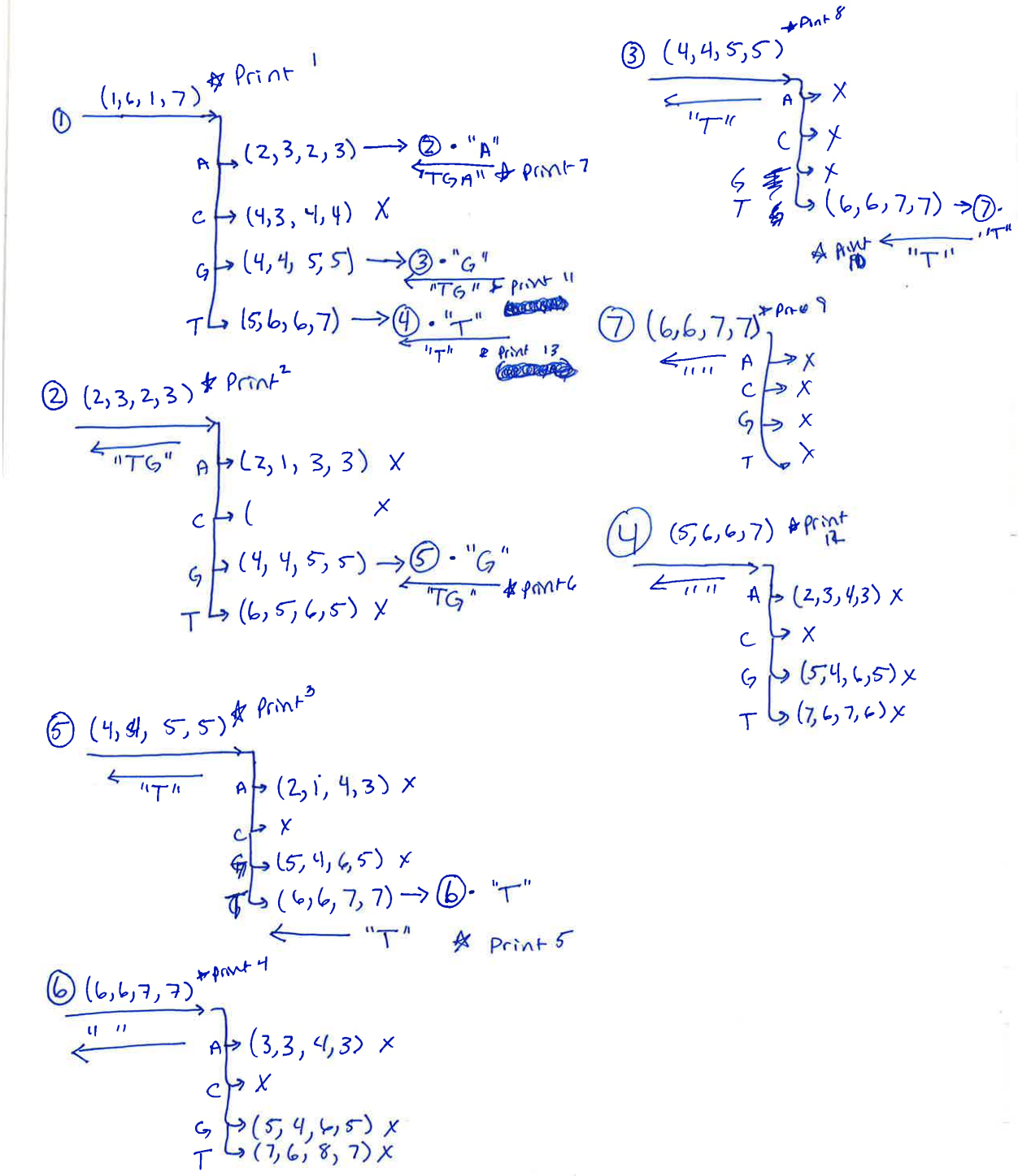
3	ATGAT$
6	TGAT$A
4	GAT$AT
2	AT$ATG
5	T$ATGA
1	$ATGAT

	L_1	L_2
1	T	A
2	G	A
3	$	G
4	T	T
5	A	T
6	A	$
7		C

6	TCTGAA$
4	CTGAA$T
7	TGAA$TC
5	GAA$TCT
3	AA$TCTG
2	A$TCTGA
1	$TCTGAA

	C_1	C_2
$	0	0
A	1	1
C	3	3
G	3	4
T	4	5

Program Output:
(1,6,1,7)
(2,3,2,3)
(4,4,5,5)
(6,6,7,7)
found match of T
found match of TG
found match of TGA
(4,4,5,5)
(6,6,7,7)
found match of T
found match of TG
(5,6,6,7)
found match of T
The longest common sequence is TGA


① (1,6,1,7) ★ Print 1
A → (2,3,2,3) → ② • "A"
"TGA" ★ Print 7
C → (4,3,4,4) X
G → (4,4,5,5) → ③ • "G"
"TG" ★ Print 11
T → (5,6,6,7) → ④ • "T"
"T" ★ Print 13
② (2,3,2,3) ★ Print 2
"TG"
A → (2,1,3,3) X
C → (X
G → (4,4,5,5) → ⑤ • "G"
"TG" ★ Print 6
T → (6,5,6,5) X
③ (4,4,5,5) ★ Print 8
"T"
A → X
C → X
G → X
T → (6,6,7,7) → ⑦ • "T"
★ Print 10 "T"
⑦ (6,6,7,7) ★ Print 9
" "
A → X
C → X
G → X
T → X
④ (5,6,6,7) ★ Print 12
" "
A → (2,3,4,3) X
C → X
G → (5,4,6,5) X
T → (7,6,7,6) X
⑤ (4,4,5,5) ★ Print 3
"T"
A → (2,1,4,3) X
C → X
G → (5,4,6,5) X
T → (6,6,7,7) → ⑥ • "T"
"T" ★ Print 5
⑥ (6,6,7,7) ★ Print 4
" "
A → (3,3,4,3) X
C → X
G → (5,4,6,5) X
T → (7,6,8,7) X

3. (5 point) Consider the Neighbor Joining algorithm we discussed in class.

   - How would you find a triple, $\{A, B, C\}$, of clusters, rather than a pair, that minimizes the sum of the distances between the three elements and maximizes the average distance to all of the other clusters?

$$\underset{(A,B,C)\in Z^3, A\neq B\neq C}{\operatorname{argmin}} \{D(A,B) + D(B,C) + D(A,C) - u_A - u_B - u_C\}$$

   - When a new cluster D is created, a new node r is created with edges from r to the root of A (r_A), the root of B (r_B), and the root of C (r_C). What values would you assign the weights of these edges so that the sum of the edges from r_A to r and r to r_B is equal to $D(A, B)$ (and the same for $D(A, C)$ and $D(B, C)$)?

$$e_A = 1/2D(A,B) - 1/2D(B,C) + 1/2D(A,C)$$
$$e_C = 1/2D(A,C) - 1/2D(A,B) + 1/2D(B,C)$$
$$e_B = 1/2D(B,C) - 1/2D(A,C) + 1/2D(A,B)$$

   - For each other cluster F we want the distance between D and F to be the average distance between F and A, B, and C minus the average of the new edge weights created in the previous step. Write this formally as an equation.

$$\forall F \in Z \setminus \{A, B, C\} : D(D,F) = \frac{1}{3}\Big(D(A,F) + D(B,F) + D(C,F) - e_A - e_B - e_C\Big)$$

4. (3 point) We know that Jaccard is only an approximation of the alignment score, if the number of aligned bases is high the Jaccard is guaranteed to be high. Give an example of why the converse is is not true (i.e. where Jaccard is high, and the number of aligned characters is small). This can be done with an alphabet as small as 2 characters.

assume $k = 3$

11000111

00111000

5. (2 point) Betweenness calculates the number of shortest paths run though a node in a network. Closeness measures the distance to all other nodes. Create a new measure that is a parameterized combination of these two measures. Use $\alpha \in [0, 1]$ as the parameter such that $\alpha = 1$ will find the node in the graph that optimizes closeness and $\alpha = 0$ will find the node that optimizes betweenness.

$$\alpha \frac{N-1}{\sum_j d(i,j)} + (1-\alpha) \sum_{j<k} \frac{g_{jk}(n_i)}{g_{jk}}$$

6. (3 point) For two sequences A and B, let F_A and F_B be σ^k-sized arrays containing the associated k-mer frequencies. Define the frequency distance between two sequences as

$$D_F(A,B) =: \frac{2 * \sum_{i \in \Sigma^k} |F_A[i] - F_B[i]|}{|A| + |B| - 2k + 2}.$$

Design heuristic greedy algorithm that that given two collections of m sequences $X = \{x_1, x_2, ..., x_m\}$ and $Y = \{y_1, y_2, ..., y_m\}$ will find the mapping assigns a mapping from each $x_i \in X$ to an element $y_j \in Y$ and tries to maximize $D_F(x_i, y_j)$.

(1 point extra credit) D_F as stated is exactly equivalent to one of: Jaccard, Weighted Jaccard, or Edit Distance. Which?

Algorithm 2: Return a mapping M indexed by elements in X and containing elements of Y

```
M(ℓ) = null ∀ℓ ∈ X
M'(ℓ') = null ∀ℓ' ∈ Y
for x ∈ X do
    for y ∈ Y do
        if M'[y] == null  and  (M[x] == null|D_F(x, M[x]) > D_F(x, y)) then
            M[x] = y
        end
    end
end
return M
```

(extra credit) Weighted Jaccard

7. (1 point) How many sequences are provided as input to **pairwise** alignment?

two (2)