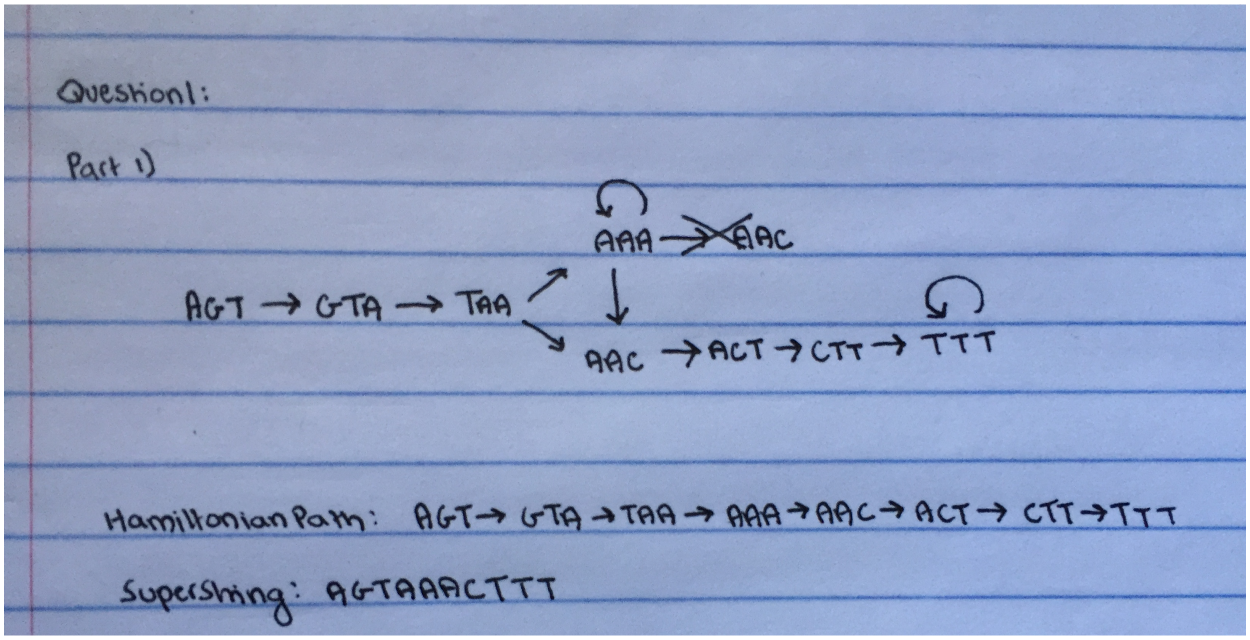
Ayaana Patel Sikora

Bi 181: Problem Set 2

1.



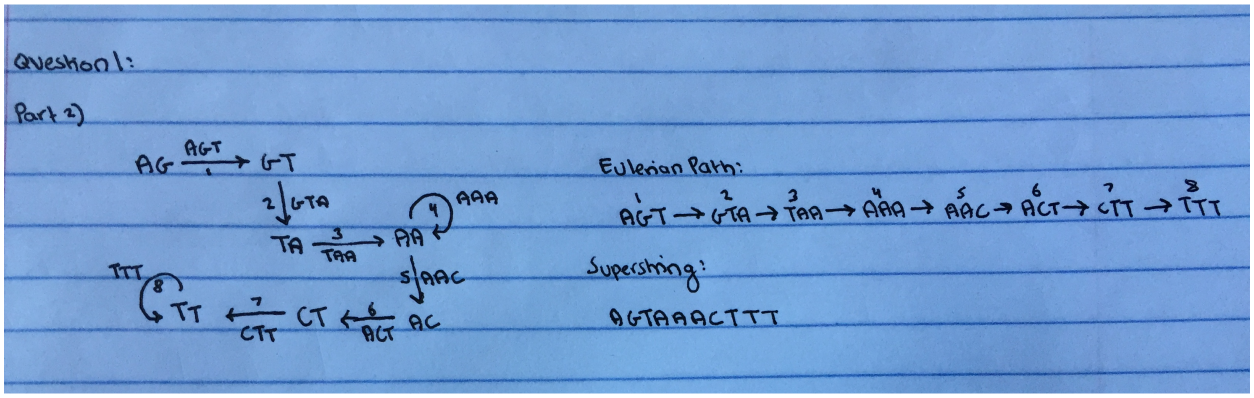
Question1:

Part 1)

$$\begin{tikzcd}
 & & & AAA \arrow[loop above] \arrow[r] \arrow[dd] & \cancel{AAC} & & \\
AGT \arrow[r] & GTA \arrow[r] & TAA \arrow[ur] \arrow[dr] & & & & \\
 & & & AAC \arrow[r] & ACT \arrow[r] & CTT \arrow[r] & TTT \arrow[loop above]
\end{tikzcd}$$

Hamiltonian Path: AGT → GTA → TAA → AAA → AAC → ACT → CTT → TTT

Superstring: AGTAAACTTT



Question1:

Part 2)

$$\begin{tikzcd}
AG \arrow[r, "AGT", "1"'] & GT \arrow[d, "2"', "GTA"] & \\
 & TA \arrow[r, "3", "TAA"'] & AA \arrow[loop right, "4", "AAA"'] \arrow[d, "5"', "AAC"] \\
TTT\ TT \arrow[loop left, "8"] & CT \arrow[l, "7", "CTT"'] & AC \arrow[l, "6", "ACT"']
\end{tikzcd}$$

Eulerian Path:

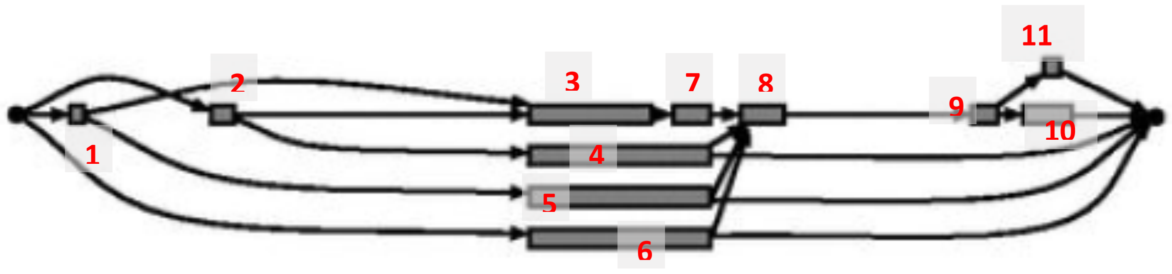
$\overset{1}{AGT} \to \overset{2}{GTA} \to \overset{3}{TAA} \to \overset{4}{AAA} \to \overset{5}{AAC} \to \overset{6}{ACT} \to \overset{7}{CTT} \to \overset{8}{TTT}$

Superstring:

AGTAAACTTT

2. See attached code entitled 'EulerianPath.py'
   Superstring:
   cgacgtggatatcccgggaggtcactctccccgggctctgtcctagtggcgagcgggagcttagggcattgcccggtgatgta
   cagtccctttccacaacgttggagataaagctgggctttcgagtctgcgcctgcatattcctacgacttctcagagtcctgtgga
   ccatgactgaggagacaaaccatgcaggaaacagtt

3.



There are a total of 13 isoforms in this picture. Each splicing area has been labeled with a number. Each isoform is depicted below using these numbers:

begin -> 6 -> end
begin -> 6 -> 8 -> 9 -> 10 -> end
begin -> 6 -> 9 -> 9 -> 11 -> end
begin -> 1 -> 5 -> end
begin -> 1 -> 5 -> 8 -> 9 -> 10 -> end
begin -> 1 -> 5 -> 8 -> 9 -> 11 -> end
begin -> 2 -> 4 -> end
begin -> 2 -> 4 -> 8 -> 9 -> 10 -> end
begin -> 2 -> 4 -> 8 -> 9 -> 11 -> end
begin -> 2 -> 3 -> 7 -> 8 -> 9 -> 10 -> end
begin -> 2 -> 3 -> 7 -> 8 -> 9 -> 11 -> end
begin -> 1 -> 3 -> 7 -> 8 -> 9 -> 10 -> end
begin -> 1 -> 3 -> 7 -> 8 -> 9 -> 11 -> end

4.

1.

Global	**-**	**1.G**	**2.A**	**3.C**	**4.G**	**5.T**	**6.A**	**7.C**	**8.G**
-	0	-1	-2	-3	-4	-5	-6	-7	-8
1.T	-1	-1	-2	-3	-4	-3	-4	-5	-6
2.A	-2	-2	0	-1	-2	-3	-2	-3	-4
3.C	-3	-3	-1	1	0	-1	-2	-1	-2
4.G	-4	-2	-2	0	2	1	0	-1	0
5.G	-5	-3	-3	-1	1	1	0	-1	0
6.G	-6	-4	-4	-2	0	0	0	-1	0
7.T	-7	-5	-5	-3	-1	1	0	-1	-1
8.A	-8	-6	-4	-4	-2	0	2	1	0
9.T	-9	-7	-5	-5	-3	-1	1	1	0

Score = (1 * numMatches) + (-1 * numMismatches) + (-1 * numGaps)

G A C - - G T A C G
T A C G G G T A - T
Score = (1 * 5) + (-1 * 2) + (-1 * 3) = 0

G A C - - G T A C G
T A C G G G T A T T
Score = (1 * 5) + (-1 * 3) + (-1 * 2) = 0

G A C - G - T A C G
T A C G G G T A - T
Score = (1 * 5) + (-1 * 2) + (-1 * 3) = 0

G A C - G - T A C G
T A C G G G T A T T
Score = (1 * 5) + (-1 * 3) * (-1 * 2) = 0

G A C G - - T A C G
T A C G G G T A - T
Score = (1 * 5) + (-1 * 2) + (-1 * 3) = 0

G A C G - - T A C G
T A C G G G T A T T
Score = (1 * 5) + (-1 * 3) + (-1 * 2) = 0

2.

Local	**-**	**G**	**A**	**C**	**G**	**T**	**A**	**C**	**G**
-	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	1	0	0	0
A	0	0	1	0	0	0	2	1	0
C	0	0	0	2	1	0	1	3	2
G	0	1	0	1	3	2	1	2	4
G	0	1	0	0	2	2	1	1	3
G	0	1	0	0	1	1	1	0	2
T	0	0	0	0	0	2	1	0	1
A	0	0	1	0	0	1	3	2	1
T	0	0	0	0	0	1	2	2	1

T A C G
T A C G
Score = (1 * 4) + (-1 * 0) + (-1 * 0) = 4

5. See the attached code entitled 'AlignmentGraph.py'
This code uses the local alignment algorithm to construct a table. It returned the

following result:
ATCTCAAACACTTGCGTGACCTCAGATA found at index: 9393 in CASP1 gene.
ATCTCAAACACATGCGGGACCCCAGATA found at index: 0 in sequence.

6.
   a. Using the substitution matrix for the scoring:
      Alignment Score = (-1) + (4) + (0) + (4) + (1) + (-4) + (2) + (5) + (-1) + (2) + (-4) + (-1) + (-1) + (-1) + (-2) + (4) + (-2) + (-1) + 8 = 12
   b. See the attached code entitled ‘originalBLAST.py’
      i. Results for part 1:
         1. TCAGGTCACTCCATGCACAT: [(1990, 2009)]
         2. CAGTTCTGATTCTTTAATGG: [(1401, 1420)]
         3. AACTCAAG: [(7621, 7628), (7632, 7639), (10237, 10244)]
         4. CATTAATT: [(3649, 3656), (4619, 4626)]
      ii. Results for part 2:
         1. TTTATCCAATAATGGACACGTT
            a. TTTATCCAATAATGGACAAGTC: 6160 – 6181
               Score: 0.0833333333333
            b. ATCTTCCAATAATGCTCTTCTT: 9912 – 9933
               Score: 0.333333333333
            c. TTTTTGAAATAATAAATCTGGT: 11120 – 11141
               Score: 0.375
            d. TTGGTCATATAATTTTCTTTTT: 3704 – 3725
               Score: 0.416666666667
            e. TTGACTCAGTAATGCCACTGTC: 6542 – 6563
               Score: 0.416666666667

            The best alignment is probably the one with the lowest score:
            TTTATCCAATAATGGACAAGTC

         2. CATAAATTTCACAAAACATATG
            a. CCTTAATTTCACAAAACATTTG: 8852 – 8873
               Score: 0.125
            b. ATTTTATCTCACACATTTTTTG: 5052 – 5073
               Score: 0.416666666667
            c. CGCCCAGAGCACAAGACCTCTG: 6796 – 6817
               Score: 0.416666666667
            d. AATAAAGCACACATTAAAAAAC: 8195 – 8216
               Score: 0.416666666667
            e. CAGAATTGGCACAGGAGGAGTA: 1833 – 1854
               Score: 0.458333333333

            The best alignment is probably the one with the lowest score:
            CCTTAATTTCACAAAACATTTG

7. ALLPATHS paper questions:
    a. Microread assembly is challenging because there are far too many overlaps between reads to compute and most of these overlaps are wrong. Furthermore, finding all overlaps between microreads is computationally very expensive because there are so many overlaps.
    b. During the build process, repeats are stacked on top of each other (this is in the case of long perfect repeats). Later, in the editing process, these repeat regions are pulled apart so that repeats are eliminated in the microread.

‘