# 22W-COM SCI-CM121-LEC-1 / 22W-COM SCI-CM221-LEC-1 / 22W-CHEM-CM160A-LEC-1 / 22W-CHEM-CM260A-LEC-1 / 22W-BIOINFO-M221-LEC-1 / 22W-HUM GEN-M260A-LEC-1 Quiz 1 (Discussion 1A)

WILLIAM CULVER RANDALL

TOTAL POINTS

**25 / 25**

QUESTION 1

1 1a **2 / 2**

✓ **- 0 pts** Correct

**- 1 pts** conceptually correct, off by one

**- 2 pts** conceptually incorrect

QUESTION 2

2 1b **2 / 2**

✓ **- 0 pts** Correct

**- 1 pts** off by 1

**- 2 pts** does not include the correct length

QUESTION 3

3 1c **2 / 2**

✓ **- 0 pts** Correct

**- 1 pts** off by one

**- 1 pts** one set of indexes incorrect

**- 2 pts** missed comparison

QUESTION 4

4 1d **2 / 2**

✓ **- 0 pts** Correct

**- 1 pts** Part

**- 2 pts** Click here to replace this description.

QUESTION 5

5 1e **2 / 2**

✓ **- 0 pts** Correct

**- 1 pts** Click here to replace this description.

**- 2 pts** Click here to replace this description.

QUESTION 6

6 2a **3 / 3**

✓ **- 0 pts** Correct

**- 1 pts** Partially correct

**- 2 pts** Partially correct

**- 3 pts** Not correct

QUESTION 7

7 2b **6 / 6**

✓ **- 0 pts** Correct

**- 2 pts** Partially correct (missing one step)

**- 4 pts** Partially correct (not including every step)

**- 6 pts** Not correct

QUESTION 8

8 2c **3 / 3**

✓ **- 0 pts** Correct

**- 0 pts** correct, conditional on previous incorrect step

**- 3 pts** path does not reconstruct original sequence

QUESTION 9

9 2d **2 / 2**

✓ **- 0 pts** Correct

**- 2 pts** original path

QUESTION 10

10 2e **1 / 1**

✓ **- 0 pts** Correct

The rules: you can use whatever resources you want, just don't ask other people for answers. This includes online searches of "how do I solve X." Please leave your cameras on (audio off) unless you have a compelling reason to have the camera off. If you do have a compelling reason, please private message the TA (no need to provide a reason).

## Problem 1

(10 points total, 2 points each (i).)

Below is some code to generate a very simple (uncollapsed) de Bruijn graph. Anywhere there is red text and underscores, fill in the appropriate code. While the code is in Python, it should be readable as pseudocode. The code you write does not need to be syntactically correct, simply logically correct. That is, *your answer does not necessarily need to be in Python* (but it can be).

To keep things consistent, let's stick to indexing starting at 0, ranges can be provided with ":", and the final value in a sequence is not inclusive. For example, 0:4 would generate the set {0, 1, 2, 3}, as range(4) would in Python.

Some people have asked if they can run the code. Sure, go for it. Given the time constraints, I probably wouldn't.

```
# This is a function to find the kmer composition of a sequence
# seq: a sequence of {A, C, G, T}
# k: an integer
def kmer_composition(seq, k):
    all_kmers = list()
    # (a) what is the correct set of indices?
    for i in range(len(seq)+1-k):
        # (b) what are the indices for the current kmer?
        current_kmer = seq[i:i+k]
        all_kmers.append(current_kmer)
    return all_kmers
```

1 1a **2 / 2**

- ✓ **- 0 pts** Correct
- **- 1 pts** conceptually correct, off by one
- **- 2 pts** conceptually incorrect

The rules: you can use whatever resources you want, just don't ask other people for answers. This includes online searches of "how do I solve X." Please leave your cameras on (audio off) unless you have a compelling reason to have the camera off. If you do have a compelling reason, please private message the TA (no need to provide a reason).

## Problem 1

(10 points total, 2 points each (i).)

Below is some code to generate a very simple (uncollapsed) de Bruijn graph. Anywhere there is red text and underscores, fill in the appropriate code. While the code is in Python, it should be readable as pseudocode. The code you write does not need to be syntactically correct, simply logically correct. That is, *your answer does not necessarily need to be in Python* (but it can be).

To keep things consistent, let's stick to indexing starting at 0, ranges can be provided with ":", and the final value in a sequence is not inclusive. For example, 0:4 would generate the set {0, 1, 2, 3}, as range(4) would in Python.

Some people have asked if they can run the code. Sure, go for it. Given the time constraints, I probably wouldn't.

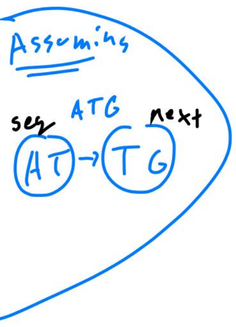
```
# This is a function to find the kmer composition of a sequence
# seq: a sequence of {A, C, G, T}
# k: an integer
def kmer_composition(seq, k):
    all_kmers = list()
    # (a) what is the correct set of indices?
    for i in range(len(seq)+1-k):
        # (b) what are the indices for the current kmer?
        current_kmer = seq[i:i+k]
        all_kmers.append(current_kmer)
    return all_kmers
```

## 2 1b 2 / 2

✓ - **0 pts** Correct

- **1 pts** off by 1

- **2 pts** does not include the correct length



```
# This is a class for a node in a graph.
# Each node can connect to the next one, thus,
# this is all you should need to make a minimalist graph.
class Node():
    # this is the constructor
    def __init__(self, seq):
        self.seq = seq
        self.edge_seq = None
        self.next = None

    # This is a member function which creates an edge to the next node.
    # For the purpose of this exercise,
    # let's assume there is only one next node.
    def add_edge(self, next_node):
        k = len(self.seq)
        # (c)
        # check if the left and right nodes have the correct matching substring
        if self.seq[        1 : k                              ] != \
                next_node.seq[          0 : k- 1              ]:
            raise Exception("substrings don't match. {%s} - {%s}" %
                            (self.seq, next_node.seq))
        self.edge_seq = self.seq + next_node.seq[k - 1]
        self.next = next_node
        return


def debruijn(seq, k):
    composition = kmer_composition(seq, k)
    graph = None
    position_pointer = None
    for kmer in composition:
        left_right_mers = kmer_composition(kmer, k - 1)
        if graph is None:
            graph = Node(left_right_mers[0])
            position_pointer = graph
        # (d) what is the correct index?
        position_pointer.add_edge(Node(left_right_mers[   1   ]))
        position_pointer = position_pointer.next
    return graph
```

3 1c **2 / 2**

✓ **- 0 pts** Correct

**- 1 pts** off by one

**- 1 pts** one set of indexes incorrect

**- 2 pts** missed comparison

```python
# This is a class for a node in a graph.
# Each node can connect to the next one, thus,
# this is all you should need to make a minimalist graph.
class Node():
    # this is the constructor
    def __init__(self, seq):
        self.seq = seq
        self.edge_seq = None
        self.next = None

    # This is a member function which creates an edge to the next node.
    # For the purpose of this exercise,
    # let's assume there is only one next node.
```

Assuming

$$\begin{tikzcd}[column sep=small] \text{seq} & \text{ATG} & \text{next} \\ \boxed{AT} \arrow[rr] & & \boxed{TG} \end{tikzcd}$$

```python
    def add_edge(self, next_node):
        k = len(self.seq)
        # (c)
        # check if the left and right nodes have the correct matching substring
        if self.seq[          1 : k                        ] != \
                next_node.seq[          0 : k- 1          ]:
            raise Exception("substrings don't match. {%s} - {%s}" %
                            (self.seq, next_node.seq))
        self.edge_seq = self.seq + next_node.seq[k - 1]
        self.next = next_node
        return


def debruijn(seq, k):
    composition = kmer_composition(seq, k)
    graph = None
    position_pointer = None
    for kmer in composition:
        left_right_mers = kmer_composition(kmer, k - 1)
        if graph is None:
            graph = Node(left_right_mers[0])
            position_pointer = graph
        # (d) what is the correct index?
        position_pointer.add_edge(Node(left_right_mers[   1   ]))
        position_pointer = position_pointer.next
    return graph
```

4 1d **2 / 2**

✓ **- 0 pts** Correct

**- 1 pts** Part

**- 2 pts** Click here to replace this description.

```
def print_seq(dbg):
    i = 0
    while dbg.next is not None:
        # (e) print the correct sequence
        if i == 0:
            print(dbg.seq)
        else:
            print(dbg.edge_seq[-1])
        dbg = dbg.next
        i += 1


# example of how you might run things
seq = 'ACGGCTAAT'
dbg = debruijn(seq, 3)
print_seq(dbg)
```

5 1e **2 / 2**

- ✓ **- 0 pts** Correct
- **- 1 pts** Click here to replace this description.
- **- 2 pts** Click here to replace this description.

## Problem 2

(15 points)

**Show work where possible.**

Assume the following DNA sequence:

$$A\overset{1}{T}\overset{2}{T}\overset{3}{T}\overset{4}{T}ATTGCGC$$

a) Draw the simple de Bruijn graph with $k = 3$, aka, $DeBruijn_3(.)$, that does not collapse any nodes (except the trivial left/right-mers). (3 points)

$$\begin{tikzcd}
\boxed{AT} \arrow[r, "ATT"] & \boxed{TT} \arrow[r, "TTT"] & \boxed{TT} \arrow[r, "TTT"] & \boxed{TT} \arrow[r, "TTA"] & \boxed{TA} \arrow[r, "TAT"] & \boxed{AT} \arrow[r, "ATT"] & \boxed{TT} \arrow[dllllll, "TTG"] \\
\boxed{TG} \arrow[r, "TGC"] & \boxed{GC} \arrow[r, "GCG"] & \boxed{CG} \arrow[r, "CGC"] & \boxed{GC} & & &
\end{tikzcd}$$

6 2a **3 / 3**

✓ **- 0 pts** Correct

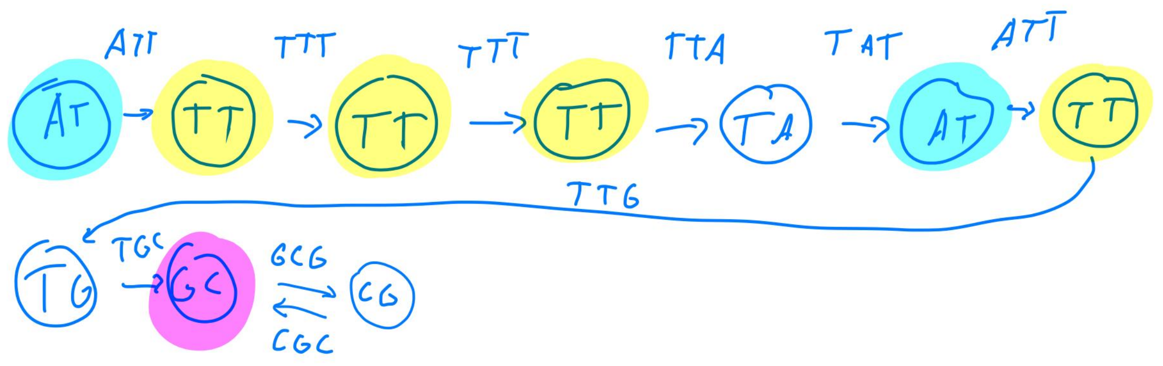
**- 1 pts** Partially correct

**- 2 pts** Partially correct

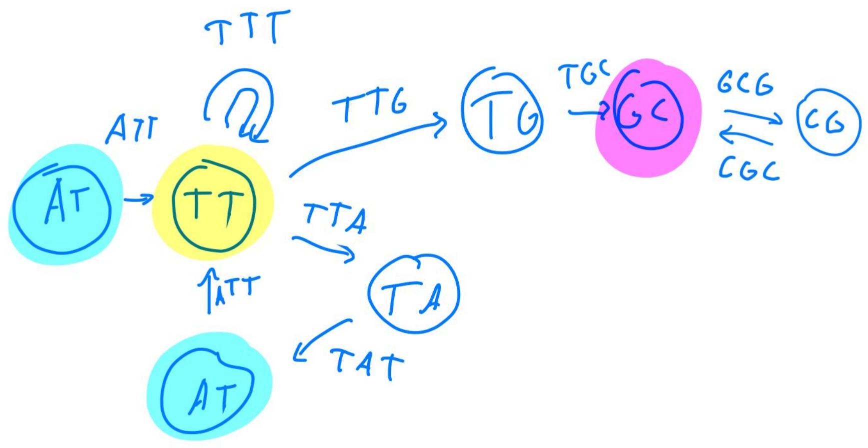
**- 3 pts** Not correct

b) Draw all intermediate collapsed graphs that collapse on common nodes. (6 points)

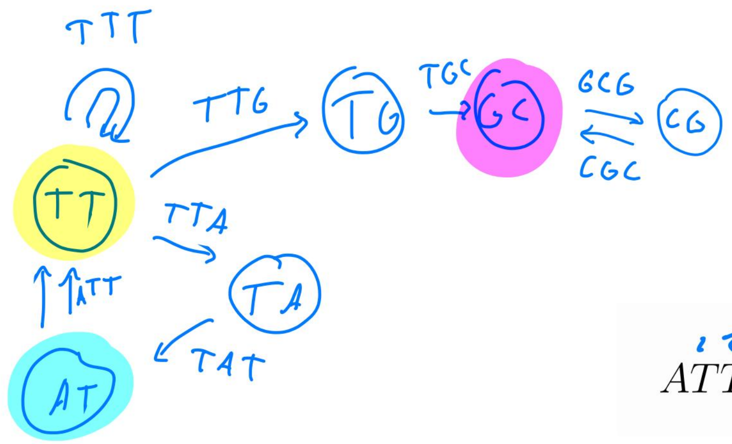
1



2



3

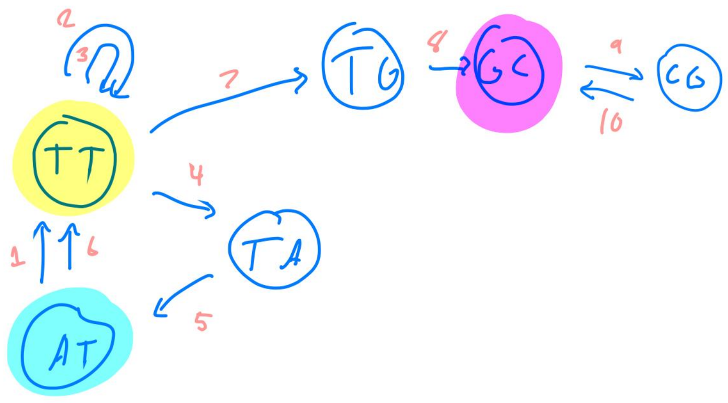


$\overset{1\,2\,3\,4}{ATTTT}ATTGCGC$ ✓

7 2b **6 / 6**

- ✓ **- 0 pts** Correct
- **- 2 pts** Partially correct (missing one step)
- **- 4 pts** Partially correct (not including every step)
- **- 6 pts** Not correct

c) Redraw your final graph from (b) and find the Eulerian path that corresponds to the original sequence but do not label the edges with their corresponding $k$-mer. Instead, label the edges on the Eulerian path edges with with an unique increasing integers starting with 1 (e.g. 1, 2, ...). (3 points)



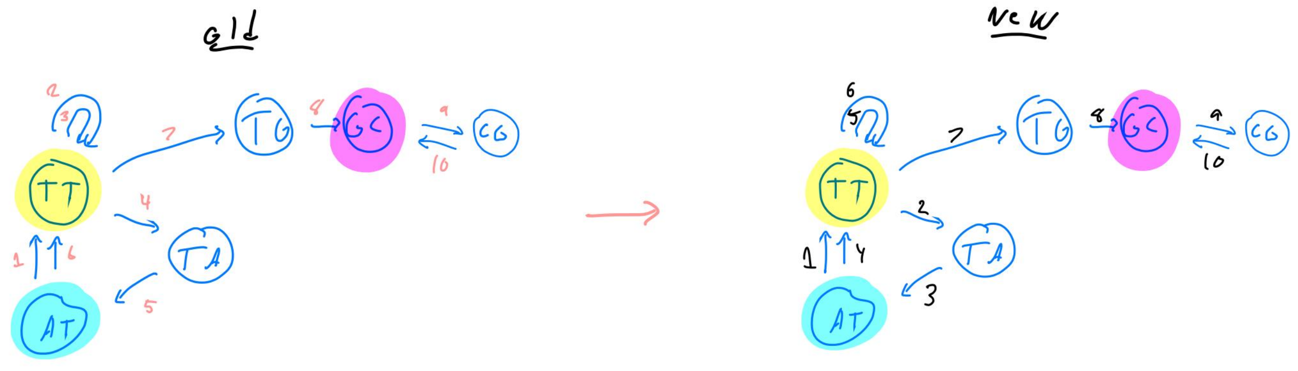
8 2c 3 / 3

✓ - **0 pts** Correct

- **0 pts** correct, conditional on previous incorrect step

- **3 pts** path does not reconstruct original sequence

d) Find an Eulerian path that generates a different sequence. You can simply write down the edge labels here and make sure to write down the corresponding sequence. (2 points)



1, 4, 5, 6, 2, 3, 7, 8, 9, 10

ATT ATTTT GCGC

## 9 2d 2 / 2

✓ **- 0 pts** Correct

**- 2 pts** original path

e) What's your favorite Power Ranger or Pokémon? (1 point)

I like the blue Power Ranger :)

10 2e **1 / 1**

✓ **- 0 pts** Correct