Authors: Daniel Oh (20063998), Evelyn Yach (20071956)

# **Programming**

Eulerian Cycle

## Q1: Why you think your algorithm is correct (whether you program worked on the sample data or not).

Our algorithm uses builds a stack to keep track of which vertices come after the others, while also generating a list of edges that have already been visited. If an edge has already been visited, it will not be ever represented on the stack. Once the last vertex is determined to have no more traversable edges, the stack then starts popping the vertices and adds them to the path until either the stack is empty or there is a vertex with an untraversed path.

## Q2: Provide an estimate of the time and space complexity of your algorithm.

Where n is the number of vertices:

Time Complexity: O(n2)

Space Complexity: O(n2), due to storing the visited edges

## Q3: Add three-unit tests using the Rosalind sample data, and some of your own. There must be at least one positive and one negative unit test.

See main.py

Contigs

## Q1: Why you think your algorithm is correct (whether you program worked on the sample data or not).

[put why here]

## Q2: Provide an estimate of the time and space complexity of your algorithm.

[put estimate here]

## Q3: Add three-unit tests using the Rosalind sample data, and some of your own. There must be at least one positive and one negative unit test.

[put output here]

# **Theory**

## 2.1 Lesson 3.3

Fuck

## 2.2 Peaceful Placement of Queens

1. What is the smallest n such that n be peacefully placed?

n = 0

n = 1, if there must be at least one Queen

n = 4, if there must be greater than one Queen

2. Write a recursive algorithm that either places the n Queen’s or determines that no such placement is possible.

See nQueens.py

3. Modify the algorithm so that it counts all peaceful placements.

See nQueens.py