GENOME ASSEMBLY PROGRAMMING CHALLENGE

Supervised by :-
Dr. Pritish Varadhwaj

Prepared by :-

Vidhu Agrawal	RSS2018003
Mrityunjay Choudhary	BIM2015004
Rameshwar Sankhala	BIM2015001
Anand Khandelwal	MBI2018001

CONTENT

- What is the need of DNA sequencing?
- Why DNA sequencing is computationally tough?
- Use of Graph Theory for DNA sequencing.
- Problems along with Algorithms.

What is the need of DNA sequencing?

- In April 2011, a mysterious bacterial strain ***E. coli X*** had infected thousands and killed 53 people in Germany.
- **Question was** " How to prevent the spread of the outbreak?"
- The **possible solution** was expected from computational biologists all over the world **"What is the genome sequence of E. coli X?"**
- To know the **new genes it acquired to become pathogenic.**

Why DNA sequencing is computationally tough?

- The major problem is that biologists still **lack the technology to read the nucleotides of a genome from beginning to end like a book.**

- **Our aim** is to assemble small fragments and the difficulty is that researchers do not know **where in the genome these reads came from**, so they must use overlapping reads to reconstruct the genome and **there are multiple overlapping reads.**

Assembling Puzzles from Repetitive Pieces

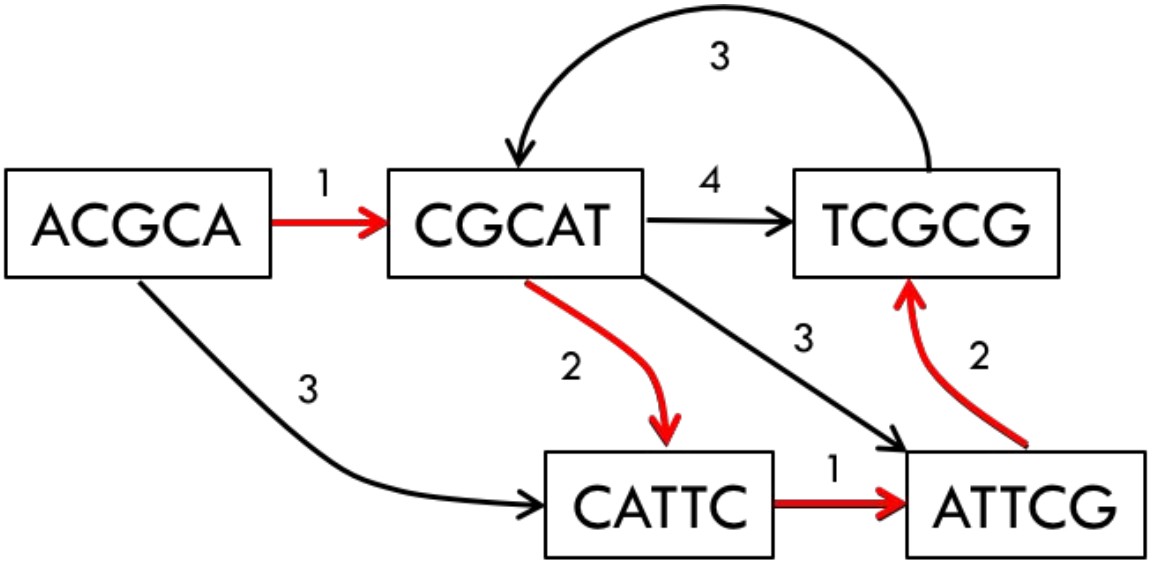
- An **analogy illustrating the difficulty of assembling a genome** with many repeats is the Triazzle jigsaw puzzle.
- People usually put together jigsaw puzzles by **connecting matching pieces.**
- However, every piece in the Triazzle **matches more than one** other piece, e.g., each frog appears several times.
- **If you proceed carelessly, then you will likely match most of the pieces but fail to fit the remaining ones.**

Triazzle jigsaw puzzle

Graph Theory as Tool for Genome Assembly

OVERLAP GRAPH

- Define SUFFIX(Pattern) and PREFIX(Pattern) as the last and first (k-1)-mers in a k-mer Pattern respectively.
- Form a vertex for each k-mer in Patterns and connect k-mers Pattern and Pattern' by a directed edge from Pattern to Pattern' if SUFFIX(Pattern)=PREFIX(Pattern').



ACGCATTCGCG

Hamiltonian path

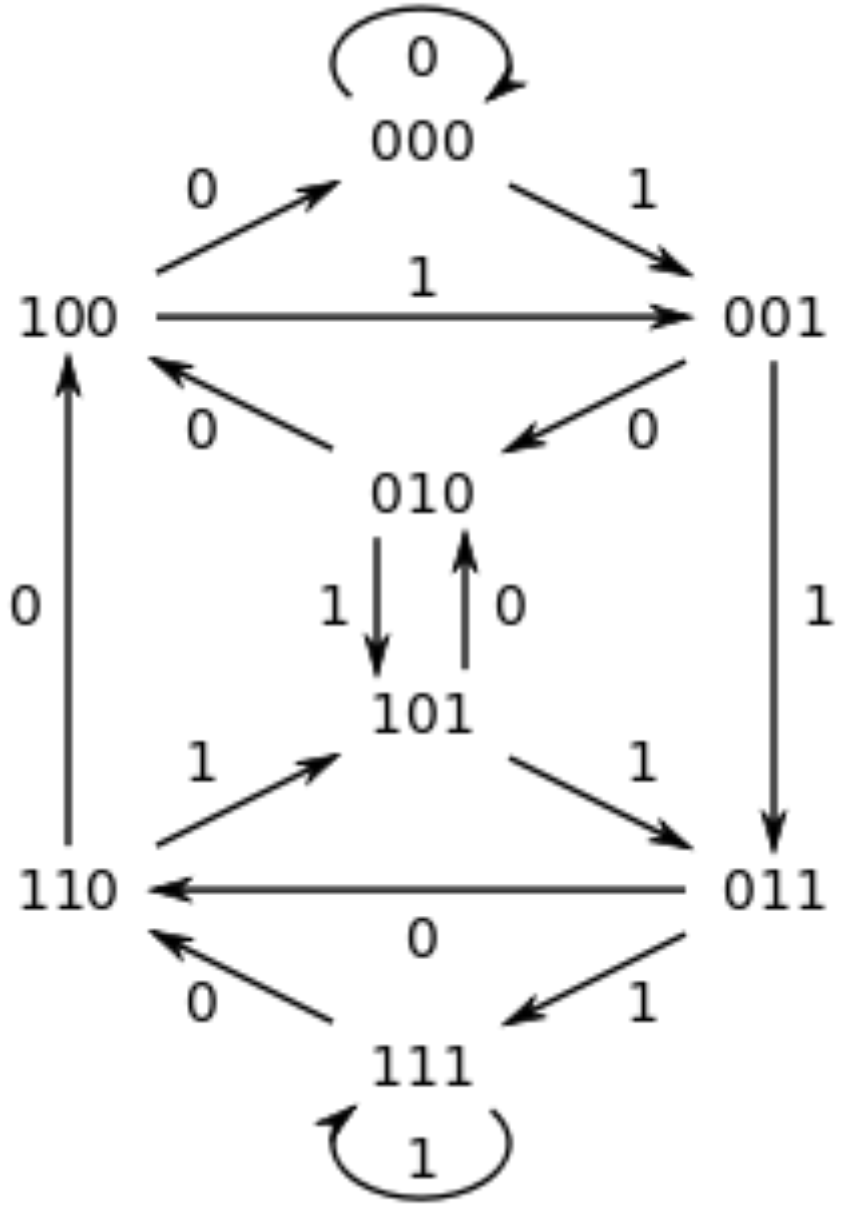
- Path in a graph that visits each vertex exactly once.
- Graph should be cyclic.
- Its a NP Hard problem.
- Reducing the String Reconstruction Problem to the Hamiltonian Path Problem is still unable to solve this problem.

DE BRUIJN SEQUENCE

- **A De Bruijn Sequence** of order *n* on a size-*k* on set A is a cyclic sequence in which every possible length-*n* string on *A* occurs exactly once as a substring.
- Denoted by *B*(*k*, *n*) and has length k^n, also the number of distinct substrings of length *n* on *A*.
- *K factorial rest to k rest to n-1 no of distinct sequences.*
- *A k-universal string is superstring when the k-mer composition is the collection of all binary k-mers.*
- *Thus, finding a k-universal string is equivalent to finding a Hamiltonian path in the overlap graph formed on all binary k-mers.*
- *Construction of k universal string for arbitary values of k by Hamiltonian Path approach is not feasible computationally and hence it paved way for new approach.*

EULERAIN PATH AND CYCLE

- A trail a finite graph which visits every edge exactly once allowing for revisiting vertices is an Eulerian Path.
- An Eulerian trail which starts and ends on the same vertex is an Eulerian Cycle.
- EULER theorem implies a polynomial-time algorithm for constructing an Eulerian cycle for any Eulerian graph irrespective of no of vertices and edges.
- An Eulerian graph must be both balanced and strongly connected.



Every four-digit sequence occurs exactly once if one traverses every edge exactly once and returns to one's starting point
.
(an Eulerian cycle)

Every three-digit sequence occurs exactly once if one visits every vertex exactly once.
(Hamiltonian path)

Case of Multiple solutions

If we represent genome as a circular string $AR_1BR_2CR_1DR_2$ (repeats are R_1 and R_2),

other solutions are_

$AR_1BR_2CR_1DR_2$

$AR_1DR_2CR_1BR_2$

$CR_1DR_2AR_1BR_2$

Assignment : 1

Assembling phi X174 Using Overlap Graphs

Problem :- 1

Assembling the phi X174 Genome from Error-Free Reads Using Overlap Graphs

Problem Statement : Given a list of error-free reads, perform the task of Genome Assembly and return the circular genome from which they came.

Algorithm :

- Construct an overlap graph. Two reads are joined by a directed edge of weight equal to the length of the maximum overlap of these two strings.
- Then construct a Hamiltonian path in this graph in a greedy fashion.
- Then read a string spelled by this path. i.e combine to form a super string.
- Sometimes choosing the wrong first vertex may result in longer superstring. So you should generate random index probably 2-3 times and find minimum length super string.
- Now in the last step since genome can be circular also so remove the overlap length between last and first read.

Output :

Assembled Sequence is aligned using Pairwise Sequence Alignment tool EMBOSS Stretcher, Result can be found here

```
#=======================================
#
# Aligned_sequences: 2
# 1: test1
# 2: test2
# Matrix: EDNAFULL
# Gap_penalty: 16
# Extend_penalty: 4
#
# Length: 125927
# Identity:   98135/125927 (77.9%)
# Similarity: 98135/125927 (77.9%)
# Gaps:       27792/125927 (22.1%)
# Score: 379483
#
#
#=======================================
```

Assignment : 2

Assembling Genomes Using de Bruijn Graphs

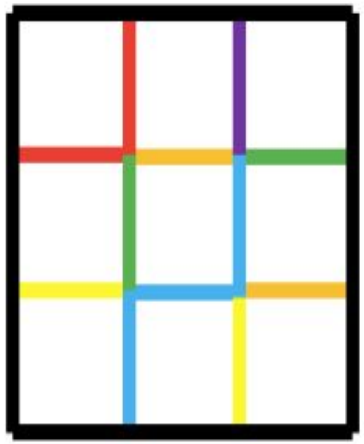
Problem : 1

Puzzle Assembly

Problem Statement : In this problem, we will consider a square puzzle consisting of n-by-n square pieces, where each square piece has a single color on each of its four edges. Given a set of n 2 square pieces, your task is to find a way to place them in an n-by-n grid such that all adjacent edges of square pieces are of the same color.

Task. Let each square piece be defined by the four colors of its four edges, in the format (up,left,down,right). Below is an example of a square piece:

Let a "valid placement" be defined as a placement of n^2 square pieces onto an n-by-n grid such that all "outer edges" (i.e., edges that border no other square pieces), and only these edges, are black, and for all edges that touch an edge in another square piece, the two touching edges are the same color. Below is an example of a "valid placement" on a 3-by-3 square grid:



You will be given 25 square pieces in the format described above, and you will need to return a "valid placement" of them onto a 5-by-5 grid. To simplify the problem, we guarantee that all of the square pieces are given to you in the correct orientation (i.e., you will not need to rotate any of the pieces to have them fit in a "valid placement"). For example, the square (green,black,red,blue) and the similar square (black,red,blue,green) are not equivalent in this problem.

- take input
- store in structure using boundary
- permutation for boundary
- permutation for non-boundary
- check the conditions with result
- print the result
- store output in a file

Puzzle Assembly.

Input :

```
jarvis@jarvis ~/Dropbox/VIII/SNGS/Final_NGS/Assignment_2/Q1 $ cat input.txt
(yellow,  black,black, blue)
(blue,blue,black,yellow)
(orange,yellow,black,black)
(red,black,yellow,green)
(orange,green,blue,blue)
(green,blue,orange,black)
(black,black,red,red)
(black,red,orange,purple)
(black,purple,green,black)jarvis@jarvis ~/Dropbox/VIII/SNGS/Final_NGS/Assignment_2/Q1 $
```

Output

```
rsbi@rsbi ~/Desktop/Final_NGS/Assignment_2/Q1 $ cat output.txt
(black,black,red,red);(black,red,orange,purple);(black,purple,green,black);
(red,black,yellow,green);(orange,green,blue,blue);(green,blue,orange,black);
(yellow,black,black,blue);(blue,blue,black,yellow);(orange,yellow,black,black);
```

Problem : 2

Finding an Eulerian Cycle in Directed Graph

Problem Statement : Given a directed graph, find an Eulerian cycle in the graph or report that none exists.

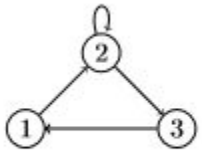
Result :

Input :

```
jarvis@jarvis ~ $ cat input
3 4
2 3
2 2
1 2
3 1
```

Output :

```
jarvis@jarvis ~ $ python3 eulerianCycle.py < input
1
1 2 2 3
```



There is an Eulerian cycle in this graph: $1 \to 2 \to 2 \to 3 \to 1$.

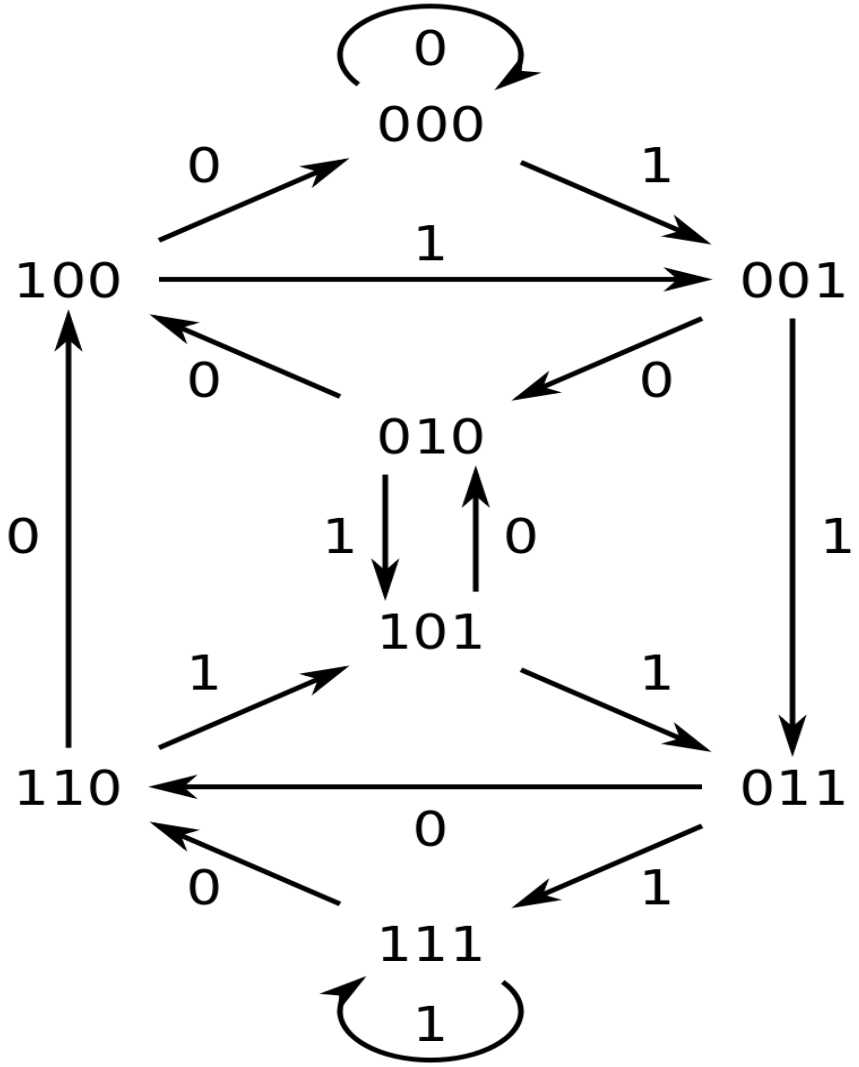
Problem : 3

Finding a k-Universal Circular String

Problem Statement : A k-universal circular string is a circular string that contains every possible k-mer constructed over a given alphabet.

Goal: to construct a $B(2, 4)$ de Bruijn sequence of length $2^4 = 16$ using Eulerian ($n - 1 = 4 - 1 = 3$)

3-D de Bruijn graph cycle.:



This corresponds to the following de Bruijn sequence:

0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 1

The eight vertices appear in the sequence in the following way:

```
    {0  0  0  0} 1  1  1  1  0  1  1  0  0  1  0  1
     0 {0  0  0  1} 1  1  1  0  1  1  0  0  1  0  1
     0  0 {0  0  1  1} 1  1  0  1  1  0  0  1  0  1
     0  0  0 {0  1  1  1} 1  0  1  1  0  0  1  0  1
     0  0  0  0 {1  1  1  1} 0  1  1  0  0  1  0  1
     0  0  0  0  1 {1  1  1  0} 1  1  0  0  1  0  1
     0  0  0  0  1  1 {1  1  0  1} 1  0  0  1  0  1
     0  0  0  0  1  1  1 {1  0  1  1} 0  0  1  0  1
     0  0  0  0  1  1  1  1 {0  1  1  0} 0  1  0  1
     0  0  0  0  1  1  1  1  0 {1  1  0  0} 1  0  1
     0  0  0  0  1  1  1  1  0  1 {1  0  0  1} 0  1
     0  0  0  0  1  1  1  1  0  1  1 {0  0  1  0} 1
     0  0  0  0  1  1  1  1  0  1  1  0 {0  1  0  1}
     0} 0  0  0  1  1  1  1  0  1  1  0  0 {1  0  1 ...
... 0  0} 0  0  1  1  1  1  0  1  1  0  0  1 {0  1 ...
... 0  0  0} 0  1  1  1  1  0  1  1  0  0  1  0 {1 ...
```

Problem : 4

Assembling the phi X174 Genome from its k-mer Composition

Problem Statement : Given the k-mer composition of some unknown string, perform the task of Genome Assembly and return the circular genome from which the k-mers came. In other words, return a string whose k-mer composition is equal to the given list of k-mers.

Algorithm :

- Read the input one by one and store in a array.
- Store array in a dictionary with id.
- keep Track on Visited and unvisited Vertices and Initialize "Visited" array
- Check if all the Vertex are visited or not if yes then break
- Goto Next if already not visited then append in a new array repeat previous 1 steps until already_visited < reads.

Result :

Input :

```
jarvis@jarvis ~ $ cat in2
AAC
ACG
CGT
GTA
TAAjarvis@jarvis ~ $
```

Output :

```
jarvis@jarvis ~ $ python3 q4.py < in2
AACGT
```

Assignment : 3

Genome Assembly Faces Real Sequencing Data

Problem : 1

Finding a Circulation in a Network

Problem Statement : Given a network with lower bounds and capacities on edges, find a circulation if it exists.

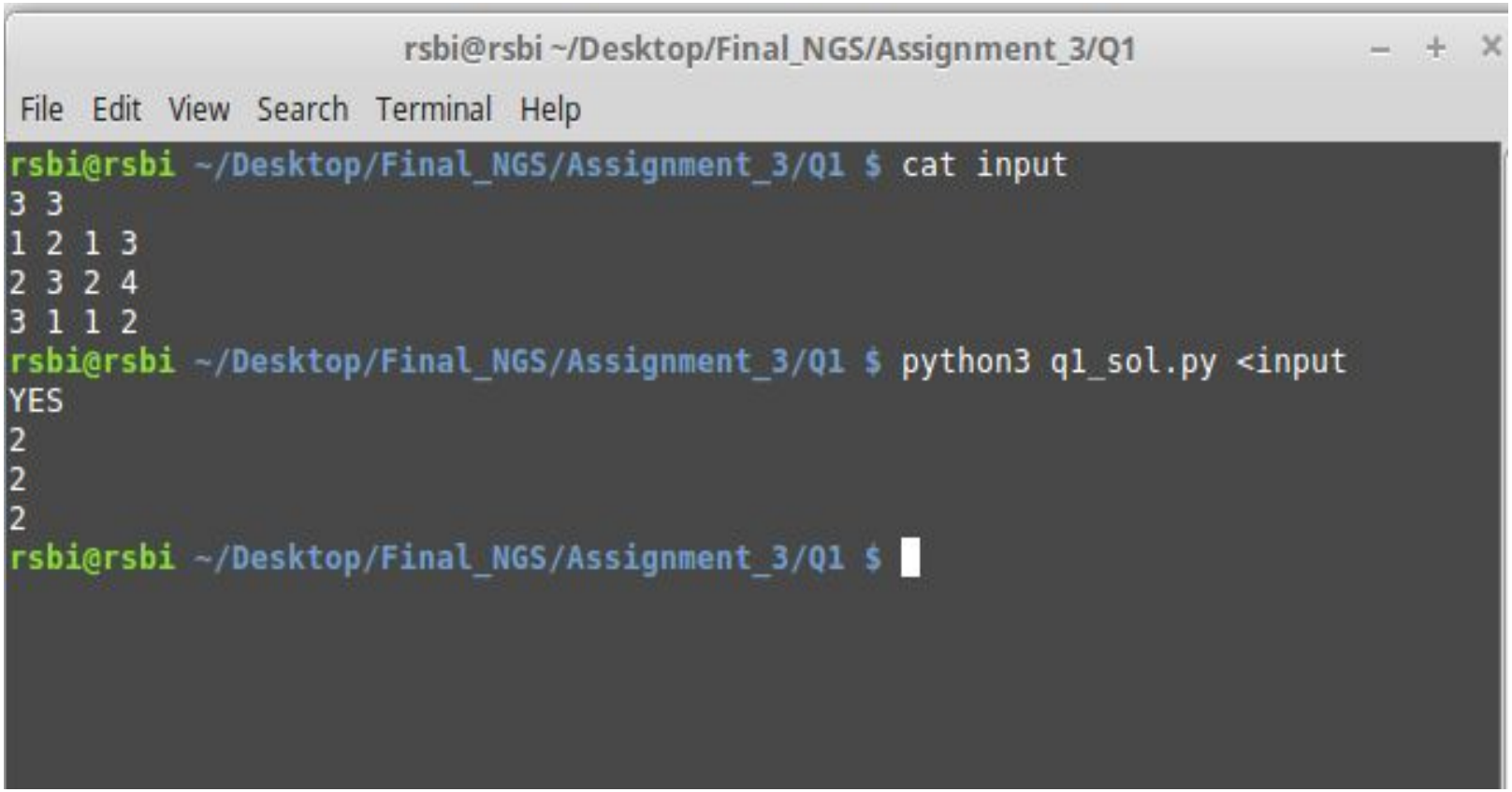
In a circulation problem, one is given a directed graph $G(V, E)$ where each edge $e \in E$ is assigned a lower bound l_e and a capacity c_e such that $0 \leq l_e \leq c_e$. The goal is to check whether it is possible to assign a flow f_e to each edge so as to satisfy the following two conditions:

- capacity conditions: for each edge $e \in E$,

$$l_e \leq f_e \leq c_e ;$$

- conservation of flow: for each vertex $v \in V$,

$$\sum_{(u,v) \in E} f_{(u,v)} = \sum_{(v,w) \in E} f_{(v,w)} .$$


rsbi@rsbi ~/Desktop/Final_NGS/Assignment_3/Q1
File Edit View Search Terminal Help
rsbi@rsbi ~/Desktop/Final_NGS/Assignment_3/Q1 $ cat input
3 3
1 2 1 3
2 3 2 4
3 1 1 2
rsbi@rsbi ~/Desktop/Final_NGS/Assignment_3/Q1 $ python3 q1_sol.py <input
YES
2
2
2
rsbi@rsbi ~/Desktop/Final_NGS/Assignment_3/Q1 $

Problem : 2

Selecting the Optimal k-mer Size

Problem Statement : Given a list of error-free reads, return an integer k such that, when a de Bruijn graph is created from the k-length fragments of the reads, the de Bruijn graph has a single possible Eulerian Cycle.

Input : Given dataset of Reads (size 33609 with 100-mers)

```python
def isOptimized(k, reads):
    kmers = set()
    for read in reads:
        for i in range(0, len(read)-k+1):
            kmers.add(read[i:i+k])
    prefixes = set()
    suffixes = set()
    for kmer in kmers:
        prefixes.add(kmer[:-1])
        suffixes.add(kmer[1:])
    return prefixes == suffixes
```

Output : 67

```
jarvis@jarvis ~/Dropbox/VIII/SNGS/Final_NGS/Assignment_3/Q2 $ python q2_sol.py < input.txt
67
jarvis@jarvis ~/Dropbox/VIII/SNGS/Final_NGS/Assignment_3/Q2 $
```

Problem : 3

Bubble Detection

Problem Statement : Given a list of error-prone reads and two integers, k and t, construct a de Bruijn graph from the k-mers created from the reads and perform the task of bubble detection on this de Bruijn graph with a path length threshold of t.

Solution :

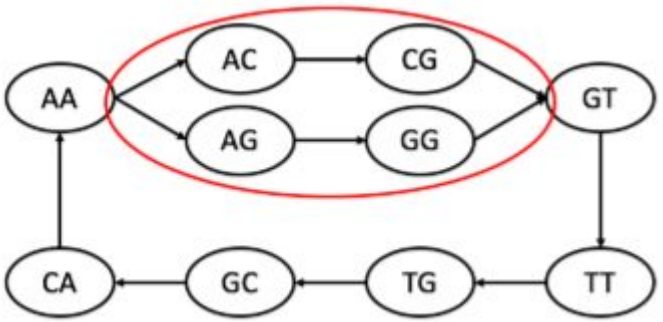
Bubble (in a de Bruijn graph) is defined as a pair of short

non-overlapping disjoint paths between some vertices V and W.

Input :

```
jarvis@jarvis ~/Dropbox/VIII/SNGS/Final_NGS/Assignment_3/Q3 $ cat input
3 3
AACG
AAGG
ACGT
AGGT
CGTT
GCAA
GGTT
GTTG
TGCA
TTGC
```

Output : 1



Problem : 4

Tip Removal

Problem Statement : Given a list of error-prone reads, construct a de Bruijn graph from the 15-mers created from the reads and perform the task of tip removal on this de Bruijn graph.

Tips : Tips are error-prone ends of the reads that do not form a bubble but instead form a path starting in a vertex without incoming edges or ending in a vertex without outgoing edges in the de Bruijn graph.

Input :

```
jarvis@jarvis ~/Dropbox/VIII/SNGS/Final_NGS/Assignment_3/Q4 $ cat input
AACG
AAGG
ACGT
CAAC
CGTT
GCAA
GTTG
TCCA
TGCA
TTGCjarvis@jarvis ~/Dropbox/VIII/SNGS/Final_NGS/Assignment_3/Q4 $
```

Output : 4