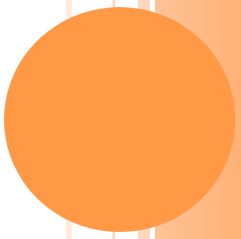
Genomic Bioinformatics

Lab Exam Programming Task Documentation

Fourth Year – 2021

TA. Esraa Hamdi

TA Nourhan Mohammed



Genomic Bioinformatics

Lab Exam Programming Task Documentation

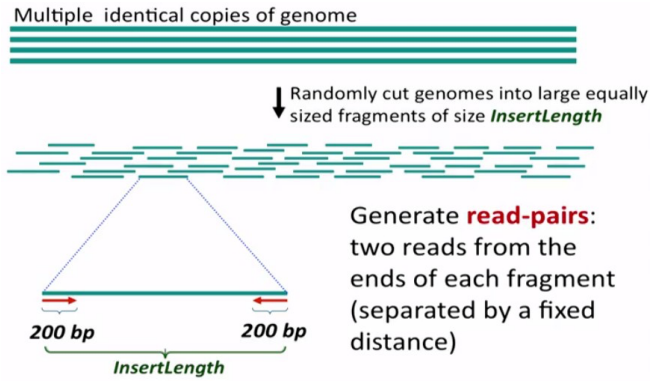
Task Overview

When the genome of a species is to be sequenced, the chromosomes from many cells are broken at random positions into small fragments, which are sequenced, and reassembled into long sequences (contigs). Contigs may be assembled into longer sequences. The resulting collection of sequences after assembly is called a genome assembly. De Bruijn graph-based assembly approach was originally proposed to handle the assembly of repetitive regions better.

In this task you should assemble two types of reads using the De Bruijn graph representation and the Eulerian path to obtain the original sequence from that representation.

First type of reads : Short reads with the same size

Second type of reads : Pair-reads with a known distance between them



Task Details

Language : Python
Number of Members in team: 6 members
Delivery Data : Dec 10th 2021

Task Requirements

Input :

Your program should be able to read **a Fastq format file (for the single read) and a regular text file (for the single and paired reads).**

The first line would be:

- the length of the sequences (for single reads).
- the length of the sequences in each side and the length of the gap (for the paired reads).

Each read will take one line in the regular text file format.
The pair reads will be separated by "|" .. Example : AGCC|TTAA

The fastq format will be as follows:

- a line for the length of the sequences
- ID
- Sequence
- +
- Quality

```
25
1
CGCGTACTTATCACATGTTTCATCT
+
SwVhMEFfBOHvNlRnppGyeCAZVp
2
TCCCACTACCCCTGGACTGTCGACT
+
PgqxjiCuwxkLIDvwZvFWMtBvuG
3
ACACTCGCTGGAGCCCTCTCACGGT
+
ZIrwGRnPblftHPlFjeGrfyWYwk
4
TTCGGAGGCCTTCAAGGCGGCAATG
+
yCnKXvEIPgCPxpMsBKrVXYjrMh
5
TCCCTTCCAAGTAGGCCTAATTCAC
+
rrPrRaTYlRqxBiKTdumAOsqMzv
```

Output : The program then outputs **the assembled sequence** to the screen.

Algorithms

De Bruijn Graph for single reads:

Constructing a de bruijn graph by dividing each k-mer into k-1-mer , the number of edges corresponds to the number of k-mers and the number of nodes corresponds to the number of distinct k-1-mers. You may represent the de bruijn graph as follows:

Sample Input:

GAGG

GGGG

GGGA

CAGG

AGGG

GGAG

Sample Output:

AGG -> GGG

CAG -> AGG

GAG -> AGG

GGA -> GAG

GGG -> GGA,GGG

De Bruijn Graph for paired reads:

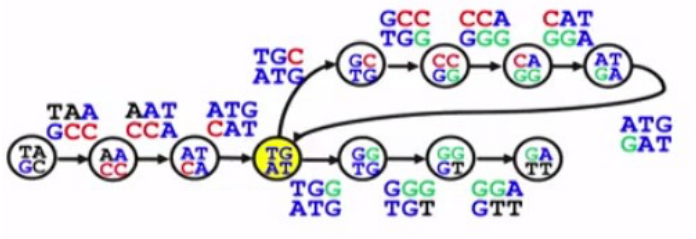
Paired reads are long "gapped" reads of length $k + d + k$ whose first and last k-mers are sequenced but whose middle segment of length d is unknown

It's an indirect way of increasing the Read length.

PairedComposition(TAATGCCATGGGATGTT)

TAA GCC
AAT CCA
ATG CAT
TGC ATG
GCC TGG
CCA GGG
CAT GGA
ATG GAT
TGG ATG
GGG TGT
GGA GTT

(AAT|CCA), (ATG|CAT), (ATG|GAT), (CAT|GGA),(CCA|GGG), (GCC|TGG), (GGA|GTT), (GGG|TGT), (TAA|GCC), (TGC|ATG), (TGG|ATG)

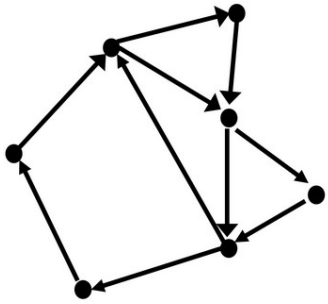


Eulerian Path Discovery :

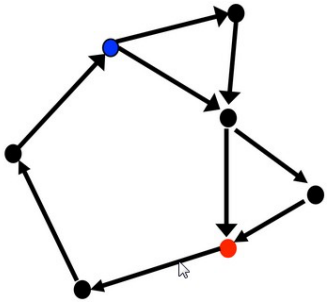
To find out the assembled sequence, we will traverse the de bruijn graph using the Eulerian path.

For a graph to be Eulerian, the number of incoming edges at any node must be equal to the number of outgoing edges at that node.

- A node v is **balanced** if *ingoing edges* (v) = *outgoing edges* (v).
- A graph is **balanced** if all its nodes are balanced.



Yes



No

Finding the Eulerian ***Cycle:***

- To solve the Eulerian Path problem, there will be a start node (without input edges) and an end node (without output edges).
- This will make the graph unclosed/unbalanced.
- So, we will connect the end node to the start node, then solve the Eulerian Cycle problem, then remove that edge from the output cycle.

Steps :

- ✓ Starting by any start node
- ✓ Ending at the same start node
- ✓ But, visiting every edge exactly once!
- ✓ If it stuck at some node:
- ✓ Search for any node at the same cycle that still has unused edges!

Example :

Sample Input:

```
0 -> 3
1 -> 0
2 -> 1,6
3 -> 2
4 -> 2
5 -> 4
6 -> 5,8
7 -> 9
8 -> 7
9 -> 6
```

Sample Output:

```
6->8->7->9->6->5->4->2->1->0->3->2->6
```

Keep in mind that the de bruijn graph in the actual input in your program will consist of k-mers not numbers, and you'll omit the last cycle "->6" to construct a path.

Sample Input:

```
GAGG
GGGG
GGGA
CAGG
AGGG
GGAG
```

Sample Output:

```
AGG -> GGG
CAG -> AGG
GAG -> AGG
GGA -> GAG
GGG -> GGA,GGG
```

The Eulerian walk for the above case is

1->2
3->1
4->1
5->4
2->5,2

3->1->2->2->5->4->1

CAGGGGAGG

Sample Run of the program using Paired reads:

Sample Input:

```
2
GAGA|TTGA
TCGT|GATG
CGTG|ATGT
TGGT|TGAG
GTGA|TGTT
GTGG|GTGA
TGAG|GTTG
GGTC|GAGA
GTCG|AGAT
```

Sample Output:

```
GTGGTCGTGAGATGTTGA
```

If there exists more than one Eulerian path, **one solution is enough**.

Notes

- Document your code whenever is possible with appropriate comments.
- Use meaningful names for variables and functions.

Lab Exam Structure :

- Each student should have a version of the code that was written by their team.
- Each student should be able **answer questions** about the code and the biological meaning behind it and **run sample data** on it.
- **The codes will be run through a plagiarism check. Any online/previous year/current year plagiarism cases will be considered zero with no exceptions. Please do your own work.**

Good Luck :)