**Genome Assembly Using Overlap Graphs: A Greedy Approach**

**Abstract**

In this work I present an implementation of a greedy assembly algorithm that constructs an overlap graph from simulated reads generated from the PhiX genome. Our pipeline generates both error‑free and error‑prone reads, applies a greedy method to merge overlapping reads, and computes assembly performance metrics. Experiments show that while error‑free data produces a near‐perfect assembly (a single contig exactly reconstructing the genome), even modest sequencing errors (1% error probability) lead to fragmentation and redundancy, despite the complete genome being present in the output.

My results show that exact-match overlaps are very sensitive to sequencing errors, so maybe we need methods that can better handle errors during genome assembly.

**Introduction**

Genome assembly is a fundamental task in computational biology, involving the reconstruction of a complete genome from short sequencing reads. One common approach is to construct an overlap graph, where nodes represent reads and edges represent overlaps between reads. The problem of assembling the genome can be viewed as finding a **shortest common superstring** that best explains the data. Although greedy algorithms provide a straightforward heuristic solution to this problem, their reliance on exact overlaps makes them vulnerable to sequencing errors.

The PhiX bacteriophage genome, which is circular and well-characterized, is frequently used as a control in sequencing experiments. In this study, I use the PhiX genome to evaluate our greedy overlap-graph assembly method under both ideal (error-free) and realistic (error-prone) conditions.

**Methods**

Our implementation is structured into several key steps:

1. **Genome Reading:**  
   The PhiX genome is read from a FASTA file ("sequence.fasta"), ignoring header lines and concatenating the sequence. The genome is treated as circular to simulate wrap-around read generation. **(can be change with Circular=False parameter in the generate function)**

def generate\_error\_free\_reads(genome, num\_reads, read\_length, circular=True):

1. **Read Generation:**
   * **Error-Free Reads:** Randomly sampled reads of length *L* (read\_length parameter) are generated from the genome, **ensuring uniform coverage.**

def generate\_error\_free\_reads(genome, num\_reads, read\_length, circular=True):

* + **Error-Prone Reads:** Errors are introduced into error-free reads by substituting each base with a different nucleotide with probability *P*. (error\_prob parameter)  
    The parameters used include the number of reads (*N*) (num\_read parameter), read length (*L*) (read\_length parameter) , and error probability (*P*). (error\_prob parameter)

def generate\_error\_prone\_reads(genome, num\_reads, read\_length, error\_prob, circular=True):

1. **Overlap Graph Construction:**  
   I construct an overlap graph where nodes represent reads, and edges represent overlaps between reads by checking for exact matches between the suffix of one read and the prefix of another. Only overlaps of at least a specified minimum length (min\_overlap default 1) are retained.

def build\_overlap\_graph(reads, min\_overlap=1):

1. **Greedy Assembly:**  
   The algorithm iteratively identifies the pair of reads with the maximum overlap and merges them until no further overlaps above the threshold remain, yielding assembled contigs.

def greedy\_assembly\_overlap\_graph(reads, min\_overlap=1):

1. **Performance Metrics:**  
   We calculate several metrics:
   * **num\_contigs:** Number of contigs generated.
   * **total\_assembled\_length:** Sum of contig lengths.
   * **max\_contig\_length:** Length of the longest contig.
   * **N50:** The contig length such that 50% of the total assembled bases are in contigs of that size or larger.
   * **genome\_reconstructed:** A boolean indicating if the original genome is a substring of any contig.

The complete code is available on GitHub at:  
<https://github.com/almogkarif/Phix-assembly>

Top of Form

Bottom of Form