CSCI 3104: Algorithms
Spring 2019

Grand Challenge

Option 1: Genome Assembly

The modern DNA revolution was brought about in 2001 by the completion of the human reference genome, which consisted of piecing together small fragments of DNA sequence into 24 full chromosomes and cost about $1 billion. Since 2001, computer scientists have made considerable headway on new algorithms for assembling genomes, and we will need to continue to improve our methods as we strive to complete reference genomes for all extant and extinct (i.e., Neanderthals) organisms. Your project is to develop an assembly algorithm.

Background

A **genome** is the complete set of an organism's DNA and encodes instruction for all development and life in a sequence of nucleotides (A, C, T, or G). Genomes are large. The smallest (non-viral) observed genome is 159,662 nucleotides long and belongs to the bacterial endosymbiont *Carsonella*[1], and the largest genome is 149 billion nucleotides and belongs to a rare Japanese flower named *Paris japonica*[2]. The human genome is 3 billion nucleotides.

Genome sequencing is the process of decoding the sequence of nucleotides from some sample's genome. Genome sequencing produces **sequences** or **reads**. Due to technical limitations, these reads are short. The most widely used sequencing platform produces reads that are 200 nucleotides long. Newer technologies are producing longer reads, but they are currently expensive and error-prone.

Genome assembly is the process of taking a large number of sequences and putting them back together to create a representation of the original genome.

Example

Consider the following example tiny genome (red) and the reads from a sequencing run (blue):

```
AAAATCTCCAGTGCCCAAGACCACGGGCGCTCGGCGCCTTGGCTAATCCCCGTACATGTTGTTATAAATAATCAGTAGAAA
AAAATCTCCAGTGCCCAAGA             GCGCCTTGGCTAATCCCCGT        TTGTTATAAATAATCAGTAA
           TGCCCAAGACCACGGGCGCT             AATCCCCGTACATGTTGTTA
                     CACGGGCGCTCGGCGCCTTG
```

The first thing to notice is that when we sequence a genome, we compensate for small read lengths by oversampling the genome. The extent to which a genome is sampled is called coverage, which means either position-level coverage (e.g., some regions above have been sampled by 2 reads and those have a coverage of 2, most have coverage of1) or a genome-wide average.

From this view, it may seem straightforward to take the reads and produce the genome. But in reality, the reads arrive in random order and the genome unknown. In many cases, we don't even know the size of the genome.

The challenge is to start from reads in random order:

```
AATCCCCGTACATGTTGTTA
GCGCCTTGGCTAATCCCCGT
AAAATCTCCAGTGCCCAAGA
TGCCCAAGACCACGGGCGCT
CACGGGCGCTCGGCGCCTTG
TTGTTATAAATAATCAGTAA
```

and produce an assembly that is as close as possible to the target genome.

The basic observation is that if we can identify pairs of reads with a sufficiently large **overlap** between the suffix (the end) of one read and the prefix (the beginning) of another. For example:

```
GCGCCTTGGCTAATCCCCGT
           AATCCCCGTACATGTTGTTA
```

The assumption here is that if the **overlap** is long enough then that sequence in the two reads originated from the same region of the genome and you can safely collapse the two reads into the single *contig*.

```
GCGCCTTGGCTAATCCCCGTACATGTTGTTA
```

As you create and combine contigs, you can expand your assembly with the ultimate goal of producing a single contig that represents the full genome of the organism.

```
   reads GCGCCTTGGCTAATCCCCGT                  TGCCCAAGACCACGGGCGCT
                     AATCCCCGTACATGTTGTTA                CACGGGCGCTCGGCGCCTTG
 contigs GCGCCTTGGCTAATCCCCGTACATGTTGTTA     TGCCCAAGACCACGGGCGCTCGGCGCCTTG
                     TGCCCAAGACCACGGGCGCTCGGCGCCTTG
                                        GCGCCTTGGCTAATCCCCGTACATGTTGTTA
                     TGCCCAAGACCACGGGCGCTCGGCGCCTTGGCTAATCCCCGTACATGTTGTTA
genome AAAATCTCCAGTGCCCAAGACCACGGGCGCTCGGCGCCTTGGCTAATCCCCGTACATGTTGTTATAAATAATCA
```

NOTE: While a single contig spanning a chromosome is the goal, in many cases the current generation of sequencing technologies and assembly algorithms do not single contig that spans a full chromosome.

You must exercise caution because there can be spurious **overlaps** that produce bad contigs. For example, while there is some **overlap** among the following three reads, the resulting contig does not resemble the target genome.

```
TTGTTATAAATAATCAGTAA
                    AATCCCCGTACATGTTGTTA
                                        AAAATCTCCAGTGCCCAAGA
```

Evaluating your assembly

When your assembly algorithm is complete and returns a set of contigs, you can use the "N50" to assess quality. The N50 is defined as the minimum contig length needed to cover 50% of the genome. To find the N50, sort your contigs by length, then scan until you reach the point in that accounts for 50% of your assembly's total length. The size of the contig that you are in is the N50.

Suppose your sorted contig lengths are 1000, 2000, 4000, and 5000 nucleotides long and the total length is 12,000. Half of 12,000 is 6,000, which is contained by the contig that is 4,000 nucleotides, giving you an N50 of 4,000.

Data

We will provide 3 sets of reads of increasing difficulty. Each data set will include short reads from the genome. Your team may attempt to assemble any or all of these genomes. Shortly before the project is due, we will release a new dataset, and we ask that you only report your N50 for that dataset.

Rules

- No post-processing. You must calculate the N50 on the contigs that are emitted by your assembly algorithm.
- No extra parameters. You cannot pass in special parameters for different inputs.
- No libraries. You must implement all data structures and algorithms from scratch.
- Cite all papers, GitHub repos, websites, books, or other sources that made meaningful contributions to your solution.

What to submit.

1. Well-formatted and well-documented **source code**.
2. Final **assemblies** in FASTA[3] format.
3. **An annotated bibliography**. List all references for your project with a short paragraph for each (2-3 sentences) that describes how the work contributed to your project.
4. A **video presentation** of your project
    - mp4 or QuickTime format
    - 7 minutes or less.
    - This is a technical presentation to your professors in this class
    - You can assume a vague familiarity with genomes, algorithms, and asymptotic analysis, but you cannot assume knowledge of specific algorithms and you should refrain from using jargon.
    - Clearly and concisely explain your solution, your results, and future directions.
    - You can assume your audience has your source code and please refer to specific sections as you describe your solution.
    - Do not treat this presentation as a code review. Do not step through every line of your program.

Grading

- Assemblies: **max 40 points**
    - Up to **10 points** for each assembly depending on the N50.
    - Assemblies with an N50 that is not appreciably larger than the size of the input reads will receive zero credit. That is, do not give us an assembly that matches the input.
    - Contigs that do not match the source genome in at least 90% of the positions (called 90% sequence identity) will not be considered. That is, do not give us randomly generated contigs.

12/20 match = 60% sequence identify

AGTGCCCAAGACCACGGGCG

AAAATCTCCAGTGCCCAAGACCACGGGCGCTCGGCGCCT

18/20 match = 90% sequence identify

AGTGCCCAAGACCACGGGCG

AAAATCTCCAGTGCCCAAGACCACGGGCGCTCGGCGCCT

- Annotated bibliography: **max 10 points**
- Video presentation: **max 50 points**
    - Half of the points will depend on clarity and half on technical correctness
    - Solution, algorithm and code (suggest ~4-5 min): **max 40 point**s
        - a clear explanation of your solution: **max 20 points**
        - implement an algorithm from the literature or a non-trivial algorithm of your own design (i.e., more complex than an array): **max 20 points**
    - Results (suggest ~1-2 min): **max 5 points**
    - Future directions (i.e., what would you do with more time?) (about 1 min): **max 5 points**
- Projects will not be graded unless all 4 components (source code, assemblies, annotated bibliography, video) are received.

References

1. Nakabachi A., et al. Science, 314 . 267 (2006).
2. Pellicer J,. et al. Bot J Linean Soc, 164: 10-15.
3. Wikipedia.org, FASTA format. (2019). at <https://en.wikipedia.org/wiki/FASTA_format>