Genome 540 Discussion

January 11th, 2024
Clifford Rostomily
Logistical stuff

- Come talk to me after class for a slack invite if you are not registered but plan on registering
- When complete, email your homework to Phil and me
Agenda

- Assignment 1
- Burrows Wheeler
- Assignment 2
- C++ tips
Assignment #1
How do I look up the longest match?

- **.gff file**
  - General Feature Format
  - Each row is a feature, each column is info about the feature (e.g. position, name, ...)
  - Similar to a .bed file but specified for sequences
  - Either intersect it with the position of your sequence using bed tools, or scroll to the correct position

- **Nucleotide BLAST**
  - [Link](#)
  - Enter sequence, select organism, and hit BLAST
How do I define match length?

- Three pointers to suffixes
  - p14: AAAGGGGGG (from seq2)
  - p10: ATAGGGGGG (from seq1)
  - p11: ATCCCCCC (from seq 2)

- What is the match length, and to what sequence (above or below)?
# Small Example

**Seq. 1:** TCAA  
**Seq. 2:** ACTG  
**Seq. 2:** CAGT (reverse complement)

### Steps:
- **Pointer array to suffixes**
  - p1: TCAA
  - p2: CAA
  - p3: AA
  - p4: A
  - p5: ACTG
  - p6: CTG
  - p7: TG
  - p8: G
  - p9: CAGT
  - p10: AGT
  - p11: GT
  - p12: T

### Sorted
- A
- AA
- AGT
- AATG
- CAA
- CAGT
- CTG
- G
- GT
- T
- TCAA
- TG

### Iterate through sorted list
- Find the longest match to seq1
- Make a histogram of seq1 longest matches

### Things to consider:
- Multiple seq1 suffixes in a row
- Need to look above and below to find longest match

### Match Lengths:
- 1: 3
- 2: 1

### The longest match length: 2
### Number of match strings: 1
Other questions on Assignment 1?

- Read in two fasta files
  - Track # non-alpha characters
  - Also track base counts

- Combine the 3 sequences and store suffixes as an array/vector
  - Forward of seq1, forward and reverse complement of seq2 = 3

- Implement and run the suffix array algorithm
  - Returns a sorted list of pointers

- Iterate through results
  - Track longest match length of each seq1 suffix to either the forward or reverse strand of seq2
  - Track the longest overall match
Burrows Wheeler
The Burrows Wheeler Transform

- Used originally for compression
- The Bowtie aligner uses it for compression and indexing

<table>
<thead>
<tr>
<th>Transformation</th>
<th>1. Input</th>
<th>2. All rotations</th>
<th>3. Sort into lexical order</th>
<th>4. Take the last column</th>
<th>5. Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>^BANANA$</td>
<td>$^BANANA A$^BAN NA$^BANA ANA$^BAN NANA$^BA ANANA$^B BANANA$^</td>
<td>ANANA$^B ANA$^BAN A$^BANAN BANANAS^ NANA$^BA NANA$^BA BANANA$^</td>
<td>ANANA$^B ANA$^BAN A$^BANAN BANANAS^ NANA$^BA NANA$^BA BANANA$^</td>
<td>BNN$AA$A</td>
</tr>
</tbody>
</table>
Bowtie

Langmead, Trapnell, Pop, Salzberg (2009) Genome Biology
Assignment #2
Part 1 - Write a program

- The program should:
  - Read in a fasta file
  - Determine the frequencies of the nucleotides and dinucleotides (based on the forward strand) and the length of the sequence
  - Generate 3 sequences of the same length as the input file using:
    - the length (equal frequency assumption)
    - nucleotide frequency (order 0-Markov)
    - dinucleotide frequency (order 1-Markov)
  - Save these sequences as fasta files
Part 2 - Simulate Sequences

- Using your program simulate 3 sequences from the mouse genomic region in HW1 using:
  - An equal frequency assumption
  - An order-0 Markov model
  - An order-1 Markov model

- Output sequences should be the same length as the input

- Store the sequences as fasta files
Part 3 - Run your HW1 on those seqs.

- Run your program from HW1 on each of those sequences
  - Sequence 1 should always be the 10Mb mouse region from HW1,
  - Sequence 2 should be your simulated sequence
C++ tips
Random numbers using rand()

- `rand()` is the base random number generator.
- It generates numbers from 0 to `RAND_MAX`.
- It can be bounded using the modulus operation (%).
- The standard `<random>` library was added in C++11 to offer other generators with better randomness.
- A seed can be set with `srand`.

```cpp
// generate random numbers using rand()
// rand() is the old C++ random number generator
// it works well if you aren't too picky about the distribution
// it generates numbers from 0 to RAND_MAX

// generate numbers from 0 to RAND_MAX
std::cout << "RAND_MAX = " << RAND_MAX << "\n";
std::cout << "using rand(): ";
for (int i = 0; i < 5; i++){
    std::cout << rand() << " ";
}
std::cout << "\n";

// To bound rand() to a specific range, use the modulus operator (%)
// This will give you a random number from 0 to 3
std::cout << "using rand() % 4: ";
for (int i = 0; i < 5; i++){
    std::cout << rand() % 4 << " ";
}
std::cout << "\n";
```
Random numbers using `<random>`

- If using `random_device` be sure to `#include` `<random>`
- `<random>` offers a number of random number generator functions with different speed/randomness tradeoffs

```cpp
// generate random numbers using random_device
// random_device is a C++11 random number generator
// it offers a number of different generators and distributions
// here we use the mt19937 generator and the uniform_int_distribution

// generate a random integer from 0-3
int seed = 1;
std::mt19937 gen(seed);
std::random_device rd;

int min = 0;
int max = 3;
std::uniform_int_distribution<> dist(min, max);

std::cout << "using uniform_int_distribution: ";
for (int i = 0; i < 5; i++){
    std::cout << dist(gen) << " ";
}
std::cout << "\n";
```
See you next week!

- HW1 due this Sunday, 11:59pm
- Please have your name in the filename of your homework assignment