`frag_asm`: User's guide

Martin Kochan (`mkochan@ksu.edu`)

25 November, 2005

1 Description

This program performs simple DNA fragment assembly. It was written as part of the "Implementation Project" course taken by the author at Kansas State University, Manhattan, Kansas, in Spring 2005. The current version is 1.00.

2 Motivation

DNA that may be several thousand bases (characters) long. It is impossible to read the whole string at one time with current technology. However, a method called *shotgun sequencing* is capable of reading strings of limited length (up to several hundred bases) that are taken at random places in the original string. These *reads* often contain reading errors, such as point mutations (changes of characters), additions (when a meaningless character is added), and deletions (when a meaningful character is deleted).

From these reads, we want to assemble longest possible contiguous segments of the original long string, called *contigs*. We must account for reading errors!

3 Synopsis

The command line looks like:

```
./frag_asm seqs.dat seqs.out seqs.progress
```

The contents of file `seqs.dat` (input sequences, one per line, separated by newlines or whitespaces) are:

```
cagctagctactgcatcgatgctaccgatcgtaagcccacaccacac
acgtaggcgctagggctatgctaggctgcggtacgatgccctcgatcgtaagcccacacc
ggtacgatgctcgatcgtaagcccggtgctagctagcatcgatgctagctagct
ccccccctattcgatttttgggggggacatt
gctacgatcgatgctagtgctgtacccccccccctattcgatttttggggacaatttttggcccaa
gctagc
```

The output file produced `seqs.dat` then is:

```
THERE ARE 6 READS.

Contig #0:
acgtaggcgctagggctatgctaggctgcggta-cgatgccctcgatcgtaagcccacacc (1)
                 cagcta-gctactgcatcgatgctaccgatcgtaagcccacaccacac (0)
                             ggta-cgatg--ctcgatcgtaagcccggtgctagctagcatcgatgctagctagct (2)

Contig #3:
                           ccccccctattcgatttttgggggggacatt (3)
gctacgatcgatgctagtgctgtacccccccccctattcgatttttggggacaatttttggcccaa (4)

Contig #5:
gctagc (5)
```

The progress of the assembly is recorded along with the output, and is found in the file `seqs.progress`. Note that the `+++++` sign means merging of contigs and `====>` means "the resulting contig follows." Note that resulting contigs always take name of the first "operand" in the merging operation. At the beginning, all reads had their own contigs — or *singlets*.

```
Contig #3:
ccccccctattcgatttttgggggggacatt (3)
+++++
Contig #4:
gctacgatcgatgctagtgctgtacccccccccctattcgatttttggggacaatttttggcccaa (4)
====>
Contig #3:
                           ccccccctattcgatttttgggggggacatt (3)
gctacgatcgatgctagtgctgtacccccccccctattcgatttttggggacaatttttggcccaa (4)
----------------

Contig #0:
cagctagctactgcatcgatgctaccgatcgtaagcccacaccacac (0)
+++++
Contig #1:
acgtaggcgctagggctatgctaggctgcggtacgatgccctcgatcgtaagcccacacc (1)
====>
Contig #0:
                 cagcta-gctactgcatcgatgctaccgatcgtaagcccacaccacac (0)
acgtaggcgctagggctatgctaggctgcggta-cgatgccctcgatcgtaagcccacacc (1)
----------------

Contig #0:
acgtaggcgctagggctatgctaggctgcggta-cgatgccctcgatcgtaagcccacacc (1)
```

```
                cagcta-gctactgcatcgatgctaccgatcgtaagcccacaccacac (0)
+++++
Contig #2:
ggtacgatgctcgatcgtaagcccggtgctagctagcatcgatgctagctagct (2)
====>
Contig #0:
acgtaggcgctagggctatgctaggctgcggta-cgatgccctcgatcgtaagcccacacc (1)
                cagcta-gctactgcatcgatgctaccgatcgtaagcccacaccacac (0)
                            ggta-cgatg--ctcgatcgtaagcccggtgctagctagcatcgatgctagctagct (2)
---------------
```

4 Notes

The program solves the above-mentioned problem only partly (with many simplifications adopted) and certainly not optimally. To make account of the most significant facts:

- Heuristical preprocessing of reads is performed, based on *14-mers*. It makes linear-time guess about which reads to put into common contig.
- Alignment of reads is done using gapped Smith–Waterman. Currently there is no support for banded Smith–Waterman, although generation of bands' positions is already implemented internally. To continue along these lines would make for a good optimization.
- Input must be simple DNA strings delimited by newlines or spaces. Inputs shorter than 14 bases are guaranteed to remain singlets. No other chars besides `AGCT` or `agct` are allowed.
- Output is given in form of layouts rather than consensus sequences, for didactic reasons.
- Currently, complementary strands are not supported.