kluesuite instructions

This is the precompiled jar package, not github install

Build KLUE database (k-mer disk based database)

You need to determine a database name, which will be the prefix to all database files. The database is called by its prefix name for simplicity, represented as {prefix name} below

Reasonable speed for database construction is obtained using platter drives (HD or SSHD). Database builders do not use random seek access. This allows using a slower, bigger drive for construction. Moreover, construction of a cluster using NAS is feasible.

1) run multiple instances of buildDatabasePiece.sh in parallel

Program will define parameters for you when you run without parameters. Parallelization is > 100% efficient, but you should not run more pieces than you can run concurrently. (Data is sorted during import, so a major processing step is reduced by log n, n is number of parallel pieces.)

```
./buildDatabasePiece.sh

#TODO example
```

2) run combineDatabasePieces.sh

Programs will define parameters for you when you run without parameters. This cannot be parallelized.

Making single database

```
./combineDatabasePieces.sh

#TODO example
```

Making 16-part database

You may wish to make 16-part database to distribute across systems or hard disks for parallelization.

```
./combineDatabasePieces16.sh

#TODO example
```

3) delete temporary files

```
rm -r *deleteme*
```

4) (optional) I recommend keeping a database backup in case of corruption, although not required. (It does take space and time to create)

As gzipped tar

```
tar -zcf {prefix name}.tar.gzip {prefix name}*
```

Or a backup copy

The temporary files take slightly more space than the final database, so you definitely have the disk space to make a simple copy. This is faster than gzipping. If you can afford to use the space, just make a copy:

```
mkdir backup
cp {prefix name}* backup/
```

To restore

To restore from gzipped tar

```
p={prefix name}
rm -r $p.kidDB.diskall $p.kidDB.kmer $p.kidDB.kmer.startEnd
tar -zxf {prefix name}.tar.gzip
```

To restore from backup copy

```
p={prefix name}
rm -r $p.kidDB.diskall $p.kidDB.kmer $p.kidDB.kmer.startEnd
cp -r backup/* ./
```

Applications to utilize KLUE database

running KLAT (alignment tool)

```
./klat.sh

#TODO example
```

Program will define parameters for you when you run without parameters.

Execution speed is largely limited by your disk random access speed (or IOPS). SSD drives are superior in this regard, but all are not created equal. Check specifications. Some Nvme M.2 drives can access at even higher rates. You can parallelize your database using the 16-piece database.

running WILDklat (short sequence alignment/search using wild cards)

```
./wildklat.sh

#TODO example
```

(Program will define parameters for you when you run without parameters) (not implemented yet)

wild cards

FastA format defines wild cards that can be used for multiple DNA sequence values. Submitting a fastA file using these letters defines the query.

- This is the FastA standardized alphabet
- R : A, G
- Y : C, T, (U)
- K : G, T, (U)
- M : C, A
- S : C, G
- W : A, T, (U)
- B : C, G, T, (U)
- D : A, G, T, (U)
- H : A, C, T, (U)
- V : A, C, G
- N : A, G, G, T (U)
- ? : A, G, G, T (U)

Gaps are not currently supported.

running count database frequency (counts the number of k-mers in database, number of positions stored, and frequency of positions at a k-mer)

Program will define parameters for you when you run without parameters.

```
./countDatabaseFrequency.sh

#TODO example
```