Extensions to CSE Discovery

Barna Zajzon

Practical Course in Bioinformatics RWTH Aachen June 29, 2015

Outline

- Goals
- 2 Implementation Details
- 3 Results
- 4 Summary

Goals

- Initial Improvements
- 2 Indel Detection
- 3 Higher Order Errors
- 4 Hashing Memory Usage

Bugs

Index error on forward strands → motif inconsistency:

```
qgram = genome [i : i+q]  # pos 10-19
... genome_annotate[0][i + q] # trying pos 20
```

Changed to

```
genome\_annotate[0][i + q - 1] # trying pos 19
```

Bugs

Index error on forward strands → motif inconsistency:

```
qgram = genome [i : i+q]  # pos 10-19
... genome_annotate[0][i + q] # trying pos 20
```

Changed to

```
genome\_annotate[0][i + q - 1] # trying pos 19
```

Last motif in genome was missed:

```
for i in range(len(genome) - q):
```

Changed to

```
for i in range(len(genome) - q + 1):
```

Bugs

Error while merging (F + R strands) annotated qgrams
 → motifs only on REV strand missed:

```
for qgram in qgram_last:
    qgram_rev = reverse(qgram)
    if qgram_rev in qgram_first:
    ...
```

Solution

- pack motifs on F-strand in result
- iterate over motifs on R-strand
- merge or add if not present

Speedup

- Use SciPy's chi squared test for large tables
- · Keep R's Fisher's test
- Intermediate counts for motifs instead of regex

Indel Detection

Create similar contingency tables during genome annotate

	8	9	10	-	11
	Α	G	С	Х	Т
F-strand	Α	G	-	С	Т
F-strand	Α	G	-	-	G
R-strand	Α	С	Α	С	Т
R-strand	Α	Α	-	-	Т
F-strand	Α	G	-	Α	Т

Deletion Table(10)	Match	Mismatch	Insertion Table(10)	Match	Mismatch
F-strand	3	0	F-strand	2	1
R-strand	1	1	R-strand	1	1

Indel Detection

Create similar contingency tables during genome annotate

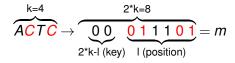
	8	9	10	-	11
	Α	G	С	Х	Т
F-strand	Α	G	-	С	Т
F-strand	Α	G	-	-	G
R-strand	Α	С	Α	С	Т
R-strand	Α	Α	-	-	Т
F-strand	Α	G	-	Α	T

Deletion Table(10)	Match	Mismatch	Insertion Table(10)	Match	Mismatch
F-strand	3	0	F-strand	2	1
R-strand	1	1	R-strand	1	1

- Parse cigars for I and D
- Tripled annotation dictionary (qgram) and array (genome)

Higher Order Errors

- Flexible offset during qgram annotation
- Arbitrary position after motif (before on R-strand)
- Drawback: one run = one position



- Size of hash table: $M = 2^{I}$
- Binary, $2k \times 2k$ invertible matrix A

$$\overbrace{ACTC}^{k=4} \rightarrow \underbrace{\begin{array}{c} 2^*k=8 \\ 0 \ 0 \\ 2^*k-l \ (key) \end{array}}_{l \ (position)} = m$$

- Size of hash table: $M = 2^{I}$
- Binary, $2k \times 2k$ invertible matrix A

$$f(m) = A * m = 10110010 \tag{1}$$

$$hash(m) = f(m) \% M = 110010$$
 (2)

$$pos(m,i) = (hash(m) + reprobe(i)) \% M$$
 (3)

$$\overbrace{ACTC}^{k=4} \rightarrow \underbrace{\begin{array}{c} 2^*k=8 \\ 0 \ 0 \\ 2^*k-l \text{ (key)} \end{array}}_{2^*k-l \text{ (position)}} = m$$

- Size of hash table: $M = 2^{I}$
- Binary, $2k \times 2k$ invertible matrix A

$$f(m) = A * m = 10110010 \tag{1}$$

$$hash(m) = f(m) \% M = 110010$$
 (2)

$$pos(m,i) = (hash(m) + reprobe(i)) \% M$$
 (3)

• i+1 is concatenated to the higher 2*k-l bits

$$f(m) = A * m = 10110010 \tag{4}$$

$$hash(m) = f(m) \% M = 110010$$
 (5)

$$pos(m, i) = (hash(m) + reprobe(i)) \% M$$
 (6)

$$f(m) = A * m = 10110010 \tag{4}$$

$$hash(m) = f(m) \% M = 110010$$
 (5)

$$pos(m, i) = (hash(m) + reprobe(i)) \% M$$
 (6)

hashtable pos	49	50 = 110010	51
key		110	
value		[fm, rm, fmm, rmm]	

Problems / Tricks

- With "N", 3 * k needed for encoding
- Numpy arrays: 1 byte * #bits
- Integers & operations on bits instead "bitvectors"
- One list as hash table entry
- Matrix A actually not needed?
- Memory improvement, but terrible speed

Results CSE Discovery

	Orig	ginal	All improvements		
bsubtilis	Time (s)	Time (s) Memory Max (MB)		Memory Max (MB)	
Q=4, N=2	421.289	223	235.481	231	
Q=8, N=2	4083.274	251	367.882	265	

	Orig	ginal	All improvements		
bordetella	Time (s)	Memory Max (MB)	Time (s)	Memory Max (MB)	
Q=4, N=2	132.779	217	58.059	225	
Q=8, N=2	870.690	242	357.841	254	

Results Hashing

	Dictionary		Key encoding		C++ with Python	
bsubtilis	Time	Memory	Time	Memory	Time	Memory
Q=10, N=0	4s	335	60s	309	16s	125
Q=10, N=1	14s	664	159s	535	Х	х
Q=10, N=2	32s	1.257	302s	914	Х	Х

Summary

- Major and initial goals achieved
- Hashing only proof-of-concept
- Better possible (merge data structures, parallelization)
- More testing required, especially for indels
- Is k-mer hashing worth it as pure Python?