Holtgrewe, Manuel
Department for Computer Science, FU Berlin

Freie Universität Berlin

Genomics Exercise 2

Read Mapping Considerations

Freie Universität Berlin

Hamming and Edit Distance

EXERCISE 2.1

Exercise 2.1 a)

Freie Universität Berlin

a) Write up the definition of the hamming distance and edit distance. For the edit distance, consider only **insert**, **delete**, and **replace** operations. Each operation has a cost of one.

Hamming Distance

Definition: An **alphabet** ∑ is a finite set of characters, e.g. ∑ = {C, G, A, T} for DNA.

Definition: A **sequence** S of length n $(0 \leq n)$ over an alphabet ∑ is an ordered list of characters from ∑. The i-th character from S is denoted as S[i] (i = 0…n-1).

Definition: Given two sequences A and B of the same length n, the **Hamming distance** H(A, B) of A and B is defined as $\sum_{i=0}^{n-1} I(A[i], B[i])$ where I is the indicator function (i.e. I(x, y) = 1 if x = y and I(x, y) = 0 otherwise).

Observation: $H(A, B) \leq n$

Exercise 2.1 a)

Freie Universität Berlin

a) Write up the definition of the hamming distance and edit distance. For the edit distance, consider only **insert**, **delete**, and **replace** operations. Each operation has a cost of one.

Edit Distance

Definition: Given a sequence S of length n, an integer i (0 ≤ i ≤ n), and a character x ∈ ∑, we define the insert operation ins(S, i, x) as a function $ins : \sum^n \rightarrow \sum^{n+1}$ that inserts x into S after S[i].

Definition: Given a sequence S of length n $(1 \leq n)$ and an integer i (0 ≤ i ≤ n − 1), we define the delete operation del(S, i, x) as a function $del: \sum^n \rightarrow \sum^{n-1}$ that removes S[i] from S.

Definition: Given a sequence S of length n, an integer i (0 ≤ i ≤ n − 1), and a character x ∈ ∑, we define the replace operation rep(S, i, x) as a function $rep: \sum^n \rightarrow \sum^n$ that replaces S[i] by x.

Definition: Given two sequences A and B of lengths n and m, we define the edit distance E(A, B) as the smallest number of insertion, deletion, and replacement operations such that after applying the operations, A is transformed into B.

Observation: $E(A, B) \leq \max(n, m)$

Exercise 2.1 b)

Freie Universität Berlin

b) Determine the Hamming distances and the edit distance between the following pairs of strings.

```
TGGTACTTCTC        TGGTGGTGG        TAGGTGGTG
TAGTTCTTCTT        TGGTGGTGG        TGGTGGTGG
```

Hamming Distance (counting mismatches)

```
TGGTACTTCTC        TGGTGGTGG        TAGGTGGTG
 X  X     X = 3                      = 0        X XX XX = 5
TAGTTCTTCTT        TGGTGGTGG        TGGTGGTGG
```

Edit Distance (alignments)

```
TGGTACTTCTC        TGGTGGTGG        TAGGTGGTG-
| || |||||  = 3    |||||||||  = 0   | |||||||  = 2
TAGTTCTTCTT        TGGTGGTGG        T-GGTGGTGG
```

```
TAGGTGGT-G
| |||||| |
T-GGTGGTGG
```

Exercise 2.1 c)

Freie Universität Berlin

c) Write a function, method, or program in a programming language of your choice to determine the edit distance between two given text strings. Test it with the examples of b)

```
13:27:48 ~ $ python
Python 2.7.5+ (default, Sep 19 2013, 13:48:49)
[GCC 4.8.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> # Adapted from http://en.wikibooks.org/wiki/Algorithm_Implementation
... def edit_distance(s1, s2):
...     # Handle corner cases, below we can assume len(s1) >= len(s2) > 0.
...     if len(s1) < len(s2): return edit_distance(s2, s1)
...     if len(s2) == 0: return len(s1)
...
...     # We fill the matrix column-wise.
...     previous_col = xrange(len(s2) + 1)  # == [0, 1, ..., len(s2)]
...     for i, c1 in enumerate(s1):
...         current_col = [i + 1]
...         for j, c2 in enumerate(s2):
...             hor = previous_col[j + 1] + 1        # horizontal in matrix
...             vert = current_col[j] + 1            # vertical in matrix
...             diag = previous_col[j] + (c1 != c2)  # diagonal in matrix
...             current_col.append(min(vert, hor, diag))
...         previous_col = current_col
...     return previous_col[-1]
```

Exercise 2.1 c)

Freie Universität Berlin

c) Write a function, method, or program in a programming language of your choice to determine the edit distance between two given text strings. Test it with the examples of b)

```
...     return previous_row[-1]
...
>>> edit_distance('TGGTACTTCTC', 'TAGTTCTTCTT')
3
>>> edit_distance('TGGTGGTGG', 'TGGTGGTGG')
0
>>> edit_distance('TAGGTGGTG', 'TGGTGGTGG')
2
...
```

Read Mapping

EXERCISE 2.2

Exercise 2.2 a)

Freie Universität Berlin

a) What is the purpose of read mapping in a next generation sequencing workflow? Which constraints make it special from more general approximate string matching problems?

Purpose (for whole genome/exome sequencing)

- Given a reference sequence **S** and a large set **R** of short reads **r** from a donor that has a genome **G** that is similar to **S**.
- The overall aim of WGS/WES is to measure features of the donor's genome (e.g. SNPs, small indels, structural variants, copy number variations, …).
- Ideally, we want to find the positions in **S** for each **r** that correspond to the sample positions in **G**.
- Many practitioners want to find a (the?) position in **S** that corresponds to the sample position in **G** in the "best" fashion.
- Another option would be to enumerate a set of positions in **S** that are likely to correspond to the sample position of **r** in **G**. After mapping all reads, a post-processing step could be used to select a "best" location for each read using the "global" view. However, this is rarely (if ever) done given the huge data sets generated by NGS.

Exercise 2.2 b)

Freie Universität Berlin

b) Give a formal definition of the read mapping problem.

For Hamming distance

Given a reference sequence S over an alphabet $\sum$, a set R of reads r, the Hamming distance function H, and a maximal distance k.

For each read r, we now want to find a set of matches (locations) in S.

A **feasible match** is a match with distance $\leq$ k. A **best match** for r is a feasible match that has the smallest distance of all feasible matches. There can be more than one best match. Matches can be ranked ascendingly by their distance.

We now can define multiple problems. For each read (1) find a best match, (2) find all best matches, (3) find up to c best matches (for a constant c), (4) find up to c best-ranking feasible matches, (5) find all matches, ...

The extension to forward and reverse strand of the reference is trivial.

For Edit distance

The definition of a match becomes more complicated (see lecture script and *Holtgrewe et al., 2011*) but the remaining definition remains the same.

Exercise 2.2 c)

Freie Universität Berlin

c) Solve the following read mapping problem instances. The distance function is the edit distance. All reads are of good quality. Write down all matches with a distance not greater than 2.

Reference: `TGGTACTTCTCCTACCCCCCA`

Read #1: `TACTT`

Read #1

```
Reference:    TGGTACTTCTCCTACCCCCCA
                 TACTT

Reference:    TGGTACTT-CTCCTACCCCCCA
                    TACTT

Reference:    TGGTACTTCTCCTACCCCCCA
                          TACTT
```

Exercise 2.2 c)

c) Solve the following read mapping problem instances. The distance function is the edit distance. All reads are of good quality. Write down all matches with a distance not greater than 2.

Reference: `TGGTACTTCTCCTACCCCCCA`

Read #2: `CTTTC`

Read #2

```
Reference:    TGGTACT-TCTCCTACCCCCCA
                   CTTTC

Reference:    TGGTACTTC--TCCTACCCCCCA
                      CTTTC

Reference:    TGGTACTTCTCCTACCCCCCA
                         CTTTC
```

Exercise 2.2 c)

Freie Universität Berlin

c) Solve the following read mapping problem instances. The distance function is the edit distance. All reads are of good quality. Write down all matches with a distance not greater than 2.

Reference: `TGGTACTTCTCCTACCCCCCA`

Read #3: `TCCTC`

Read #3

```
Reference:      TGGTACTTC-TCCTACCCCCCA
                       TCCTC

Reference:      TGGTACTTCTCCTACCCCCCA
                          TCCTC
```

Exercise 2.2 c)

Freie Universität Berlin

c) Solve the following read mapping problem instances. The distance function is the edit distance. All reads are of good quality. Write down all matches with a distance not greater than 2.

Reference: `TGGTACTTCTCCTACCCCCCA`

Read #4: `CCGCC`

Read #4

```
Reference:      TGGTACTTCTCCTACCCCCCA
                          CCGCC

Reference:      TGGTACTTCTCCTACCCCCCA
                            CCGCC

Reference:      TGGTACTTCTCCTACCCCCCA
                              CCGCC
```

Exercise 2.2 d)

Freie Universität Berlin

d) This is optional. Think about how a simple read mapper could be implemented, for instance, by reusing the result of 1b). The input to the function could be a reference sequence, a read, a constant k and a distance function. The output should contain locations of the reference sequence where the read matches with a distance not greater than k. There is no need to take efficiency considerations into account. Implement it as a testable function/method in one of your favourite programming languages. Show the correctness of your implementation by comparing it with the result of c).

Exercise 2.2 d)

Freie Universität Berlin

```
#!/usr/bin/env python
"""Primitive read mapper.

The program gets the reference and a read as the argument.  It will print a
result TSV table.

USAGE: read_mapper.py REF READ

For example:
"""

import sys
```

Exercise 2.2 d)

Freie Universität Berlin

```
def begin_search(ref, read, k):
    previous_col = range(len(read) + 1)  # no free begin gaps
    # Store best match position for reverse search.
    best = None
    for i, c1 in enumerate(ref):
        current_col = [i + 1]  # no free end gaps
        for j, c2 in enumerate(read):
            hor = previous_col[j + 1] + 1        # horizontal in matrix
            vert = current_col[j] + 1            # vertical in matrix
            diag = previous_col[j] + (c1 != c2)  # diagonal in matrix
            current_col.append(min(vert, hor, diag))
        if current_col[-1] <= k:
            if best is None or current_col[-1] < best[1]:
                best = (i + 1, current_col[-1])
        previous_col = current_col
    assert best is not None
    return best
```

Exercise 2.2 d)

Freie Universität Berlin

```
def edit_distance_search(ref, read, k):
    previous_col = range(len(read) + 1)  # no free begin gaps
    for i, c1 in enumerate(ref):
        current_col = [0]  # free begin gaps
        for j, c2 in enumerate(read):
            hor = previous_col[j + 1] + 1          # horizontal in matrix
            vert = current_col[j] + 1              # vertical in matrix
            diag = previous_col[j] + (c1 != c2)  # diagonal in matrix
            current_col.append(min(vert, hor, diag))
        if current_col[-1] <= k:
            ref_rev, read_rev = ref[:i + 1][::-1], read[::-1]
            ref_rev = ref_rev[:len(read) + k]  # no need to search more
            pos, score = begin_search(ref_rev, read_rev, k)
            yield {'begin': i + 1 - pos, 'end': i + 1,
                   'ref': ref[i + 1 - pos:i + 1], 'read': read,
                   'score': score}
        previous_col = current_col
```

Exercise 2.2 d)

Freie Universität Berlin

```
if __name__ == '__main__':
    # Program entry point.
    if len(sys.argv) != 3:
        print 'Invalid number of arguments'
        print ''
        print 'Usage: read_mapper.py REF READ'
        print ''
        print 'Example: read_mapper.py TGGTACTTCTCCTACCCCCCA TACTT'
    ref  = sys.argv[1]
    read = sys.argv[2]

    print 'BEGIN\tEND\tREF\tREAD\tSCORE'
    for match in edit_distance_search(ref, read, 2):
        print '%(begin)d\t%(end)d\t%(ref)s\t%(read)s\t%(score)d' % match
```

Exercise 2.2 d)

Freie Universität Berlin

The program finds surprisingly many matches for the first pair:

```
17:01:00 tmp $ python edit_distance2.py TGGTACTTCTCCTACCCCCCA TACTT
BEGIN    END      REF      READ     SCORE
3        6        TAC      TACTT    2
3        7        TACT     TACTT    1
3        8        TACTT    TACTT    0
3        9        TACTTC   TACTT    1
7        10       TCT      TACTT    2
7        11       TCTC     TACTT    2
9        13       TCCT     TACTT    2
9        14       TCCTA    TACTT    2
12       15       TAC      TACTT    2
12       16       TACC     TACTT    2
12       17       TACCC    TACTT    2
```

Exercise 2.2 e)

Freie Universität Berlin

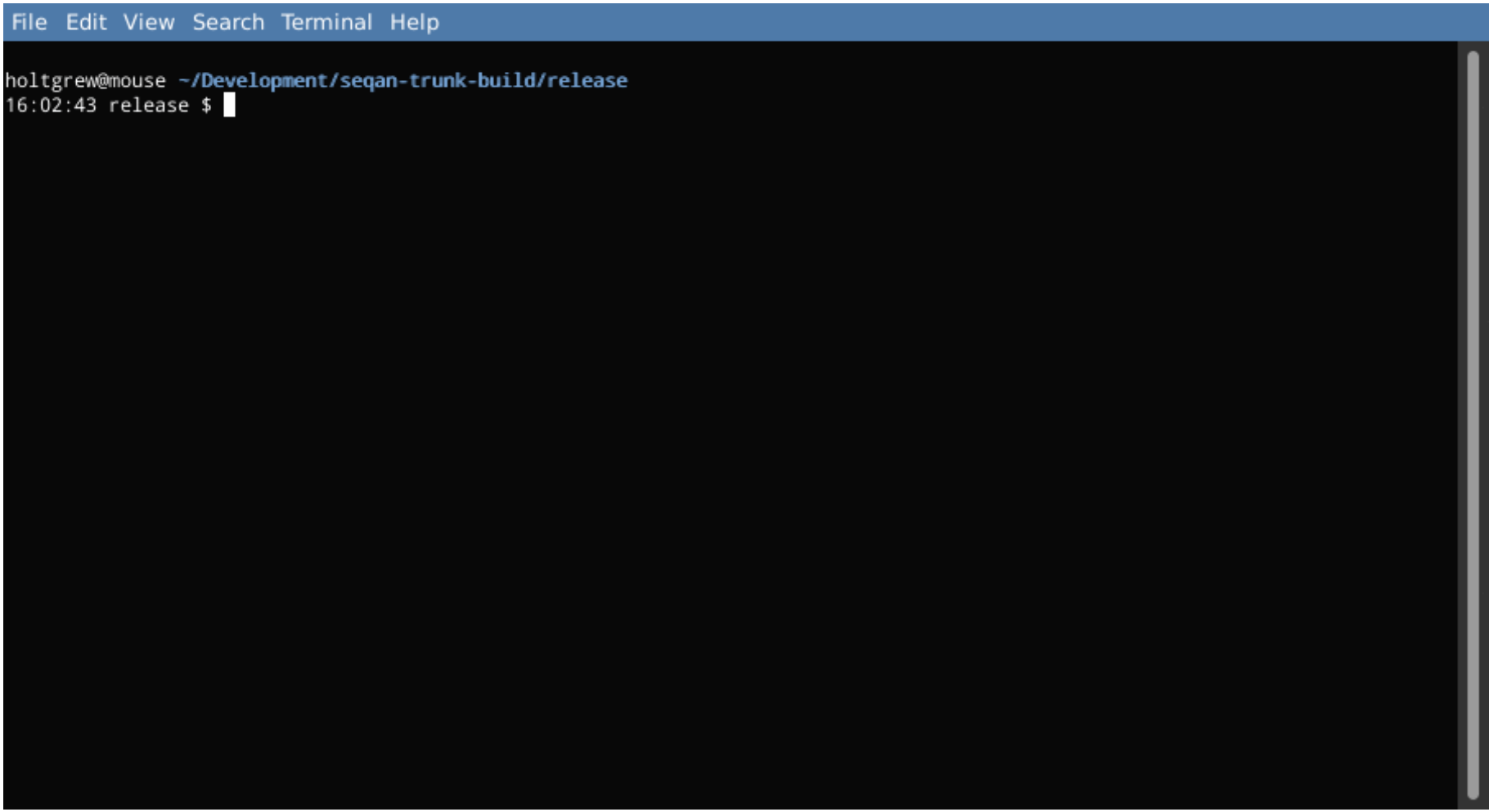
e) To make yourself familiar with various read mappers, you should reproduce the Rabema benchmark that was partly introduced in the lecture. […]

Preliminaries

- Download SeqAn through SVN and compile razers3 and Rabema.
- Download and install samtools.
- Download and install BWA.
- Download and install Bowtie2.
- Download and extract rabema-data.tar.gz from Rabema homepage

Exercise 2.2 e)

Use RazerS 3 to build gold standard SAM file.



Exercise 2.2 e)

Freie Universität Berlin

Prepare RazerS 3 “golden” SAM file for input to rabema_build_gold_standard.

```
File  Edit  View  Search  Terminal  Help
Process genome seq #8[rev]....
Process genome seq #9[fwd].......
Process genome seq #9[rev].......
Process genome seq #10[fwd]......
Process genome seq #10[rev]......
Process genome seq #11[fwd].........1M.
Process genome seq #11[rev].........1M.
Process genome seq #12[fwd].........1M
Process genome seq #12[rev].........1M
Process genome seq #13[fwd].......
Process genome seq #13[rev].......
Process genome seq #14[fwd].........1M.
Process genome seq #14[rev].........1M.
Process genome seq #15[fwd].........1M
Process genome seq #15[rev].........1M
Process genome seq #16[fwd]
Process genome seq #16[rev]Thread #0
  Masking duplicates took               0.00331764 seconds
  Compacting matches took               6.90451e-310 seconds
  Time for filtration                   1.39644 seconds
  Time for verifications                5.17978 seconds
Time for copying back                   0.00247702 seconds

Finding reads took                      6.74601 seconds

___FILTRATION_STATS____
Filtration counter:      5215634
Successful verfications: 25112
Dumping results took                    2.88325 seconds
holtgrew@mouse ~/Development/seqan-trunk-build/release
16:03:04 release $
```

Exercise 2.2 e)

Freie Universität Berlin

Build gold standard intervals (GSI) file with rabema_build_gold_standard.

```
File  Edit  View  Search  Terminal  Help
Process genome seq #12[fwd].........1M
Process genome seq #12[rev].........1M
Process genome seq #13[fwd].......
Process genome seq #13[rev].......
Process genome seq #14[fwd].........1M.
Process genome seq #14[rev].........1M.
Process genome seq #15[fwd].........1M
Process genome seq #15[rev].........1M
Process genome seq #16[fwd]
Process genome seq #16[rev]Thread #0
  Masking duplicates took               0.00331764 seconds
  Compacting matches took               6.90451e-310 seconds
  Time for filtration                   1.39644 seconds
  Time for verifications                5.17978 seconds
Time for copying back                   0.00247702 seconds

Finding reads took                      6.74601 seconds

___FILTRATION_STATS____
Filtration counter:      5215634
Successful verfications: 25112
Dumping results took                    2.88325 seconds
holtgrew@mouse ~/Development/seqan-trunk-build/release
16:03:04 release $ ./bin/rabema_prepare_sam -i out_gold.sam -o out_gold.prep.sam
holtgrew@mouse ~/Development/seqan-trunk-build/release
16:03:31 release $ samtools view -Sb out_gold.prep.sam >out_gold.bam
[samopen] SAM header is present: 17 sequences.
holtgrew@mouse ~/Development/seqan-trunk-build/release
16:03:37 release $ samtools sort out_gold.bam out_gold.by_coord
holtgrew@mouse ~/Development/seqan-trunk-build/release
16:03:41 release $
```

Exercise 2.2 e)

Freie Universität Berlin

Run RazerS 3 in lossy mode and evaluate results using rabema_evaluate.

```
File  Edit  View  Search  Terminal  Help
ref|NC_001139| (7/17) ..........1M
ref|NC_001140| (8/17) ......
ref|NC_001141| (9/17) .....
ref|NC_001142| (10/17) ........
ref|NC_001143| (11/17) .......
ref|NC_001144| (12/17) ..........1M
ref|NC_001145| (13/17) ..........
ref|NC_001146| (14/17) ........
ref|NC_001147| (15/17) ..........1M
ref|NC_001148| (16/17) ..........
ref|NC_001224| (17/17) .

Took 96.7955 s

____SMOOTHING ERROR CURVES_____________________________________________

Progress: 0%....10%....20%....30%....40%....50%....60%....70%....80%....90%....100% DONE

Took: 0.0366102 s

____WRITING OUTPUT_____________________________________________________

Writing to gold_standard.gsi ...
____POINT TO INTERVAL CONVERSION_______________________________________

Progress: 0%....10%....20%....30%....40%....50%....60%....70%....80%....90%....100% DONE
 DONE

 Took 0.0802896 s
holtgrew@mouse ~/Development/seqan-trunk-build/release
16:05:45 release $
```

Exercise 2.2 e)

Freie Universität Berlin

Run BWA and evaluate results using rabema_evaluate.

```
File  Edit  View  Search  Terminal  Help
Intervals found:                12187
Intervals found [%]             89.5313
Invalid alignments:             0
Additional Hits:                0

Number of reads:                8840
Number of reads with intervals: 8840

Mapped reads:                   8335
Mapped reads [% of total]:      94.2873
Mapped reads [% of mappable]:   94.2873

Normalized intervals found:     8284.04
Normalized intervals found [%]: 93.7109

Found Intervals By Error Rate

  ERR       #max          #found          %found        norm max        norm found      norm found [%]
-------------------------------------------------------------------------------------------------------
    0       2781            2781          100.00         2081.58           2081.58          100.00
    1       4221            4221          100.00         3062.82           3062.82          100.00
    2       2721            2721          100.00         1824.00           1824.00          100.00
    3       1118            1118          100.00          675.85            675.85          100.00
    4        758             758          100.00          387.45            387.45          100.00
    5        588             588          100.00          252.34            252.34          100.00
    6        513               0            0.00          208.19              0.00            0.00
    7        430               0            0.00          193.59              0.00            0.00
    8        482               0            0.00          154.18              0.00            0.00

holtgrew@mouse ~/Development/seqan-trunk-build/release
16:06:27 release $
```