HW3: Finding hairpin structure sequences using LCS

Algorithms, Spring 2017, SNU

Due: June 5th 11:59 P.M.
Delay penalty: 10% per day (don't accept after 5 days)
Submission to Hongryul Ahn (hrahn@snu.ac.kr)
(mail title: [Algorithm HW3] StudentID Name)
What to submit: a zip file that includes README and your program. The file should be named with a string "hairpin" then your university id
ex) 2011_11111_hairpin.zip (that include README and source code)
README file should contain all command lines that should run by cut-and-paste. If you used C, then

```
$ gcc –o hairpin _2011_11111 hairpin_2011_11111.c
$ hairpin_2011_11111 DNAseq.fasta
```

Input: a long DNA sequence can be downloaded from
http://epigenomics.snu.ac.kr/teaching/2017/algorithm/HW3/DNAseq.fasta
(This is a sequence in FASTA format that begins with a single-line sequence id (starts with ">"), followed by lines of sequence data)

Output: Print out all the hairpin structure sequences with the LCS sequence and the loop sequence. Suppose that you find two hairpin structure sequences: "AACTGCTTCAA" with "AACT" LCS sequence and "GCT" loop sequence and "CTTAGATGGATC" with "CTAG" LCS sequence and "ATG" loop sequence. Then, the output example is

```
/*--------------------------Output example------------------------------
AACTGCTTCAA
AACT
GCT

CTTAGATGGATC
CTAG
ATG
--------------------------Output example------------------------------*/
```

Programming language: any of C, C++, Java, Perl, Python

Description:
In HW3, you will implement a program to detect the hairpin structure sequence using longest common subsequence (LCS). The human genome is 3.3M characters long, which embeds specific structure sequences, called "hairpin structure sequences". A hairpin structure sequence is defined as a partially palindromic sequence where the left and right sides of the sequence show palindromic relationship (i.e. the left and the right are in reverse order), but the middle part, called loop, may not be palindromic. The following pages describe an example of "hairpin structure sequence" and how to find hairpin structure sequences.

Toy example: A hairpin structure sequence embedded in a sub-sequence of long DNA sequence like below.

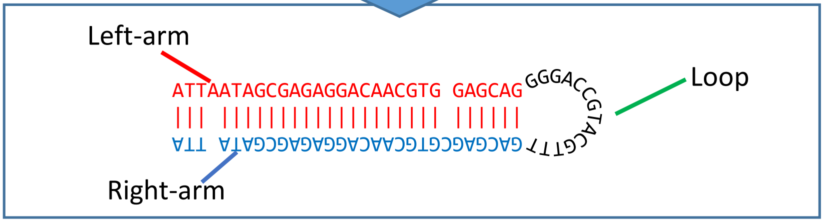
Left-arm Loop Right-arm

…GCACGATAGATATACGATAATTAATAGCGAGAGGACAACGTGGAGCAGGGGACCGTACGTTTGACGAGCGTGCAACAGGAGAGCGATATTAGGGTAGAGACAGTGTGAGCT…

A embedded hairpin structure sequence

A sub-sequence of long DNA sequence

It makes <u>hairpin structure</u>


Left-arm
ATTAATAGCGAGAGGACAACGTG GAGCAG
GGGACCGTACGTTT
Loop
GACGAGCGTGCAACAGGAGAGCGATA TTA
Right-arm

GOAL. Find out hairpin structure sequences using k-mer and LCS algorithm.

Initiation. Select the k-mer from the beginning to the end of the sequence and execute the following process.

Case 1. When you find both a k-mer and the reversed k-mer within a distance X (maximum expected loop length ≤ X ≤ maximum expected hairpin length)

여기서 조교가 의도한 X는 K-mer의 시작에서 K-mer'의 끝까지의 거리

…GCACGATAGATATACGATAATTAATAGCGAGAGGACAACGTGGAGCAGGGGACCGTACGTTTGACGAGCGTGCAACAGGAGAGCGATATTAGGGTAGAGACAGTGTGAGCT…

K-mer K-mer’

(K-mer’ is reverse of K-mer)

Step 1. Cut out the candidate region

Candidate region length = maximum expected hairpin length = $2|S_L|+2k+|S_M|$ ($\because |S_L|=|S_R|$)

GCACGATAGATATACGATAATTAATAGCGAGAGGACAACGTGGAGCAGGGGACCGTACGTTTGACGAGCGTGCAACAGGAGAGCGATATTAGGGTAGAGACAGTGTGAGCT

S_L S_M S_R

Step 2. Find LCS for (S_L, S_R’) and (S_M, S_M’). (S’ is reverse of S)

Then, they can be represented in an alignment form using “indel” to make matches located at the same position like below.

S_L S_M

```
--GCA--CGATAGA-TA-TACGAT---AATTAATAGCGAGAGGACAACGTG-GAGCAG---G--GGACC--GTACGTTTGACGAGCGTGCAACAG
  | |   | | ||  |  | |||    ||| ||||||||||||||||||| ||||||   |   | ||  |   |   |||||| |||||||||
TCG-AGT-G-T-GAC-AG-A-GATGGG-ATT-ATAGCGAGAGGACAACGTGCGAGCAGTTTGCATG-CCAGG---G---GACGAG-GTGCAACAG
```

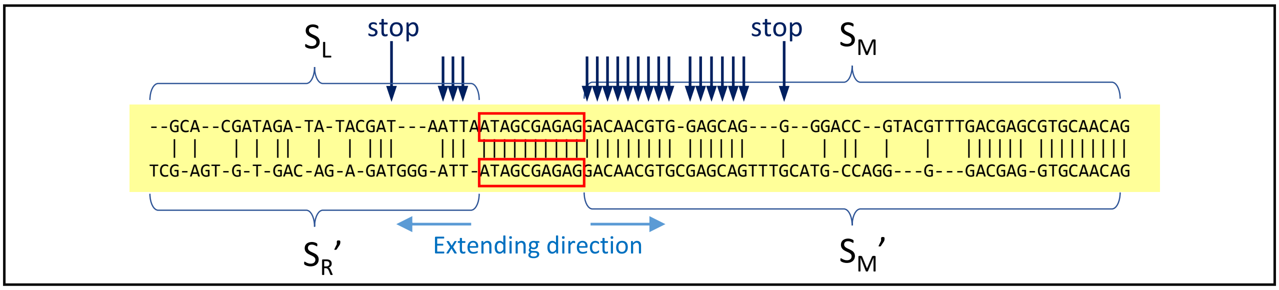
indel

match

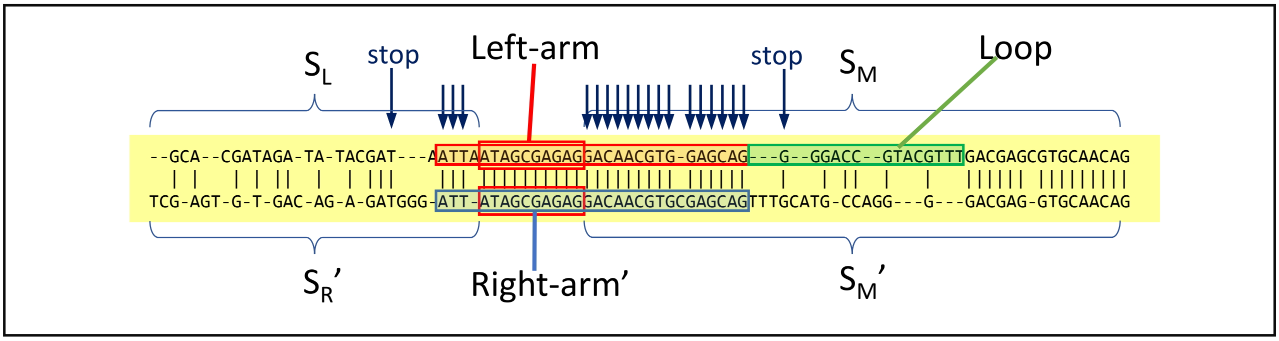
S_R’ S_M’

You can simulate LCS on the website (http://lcs-demo.sourceforge.net/)
You can transform the LCS in an alignment form by back-tracing along the ARROW matrix.
(↖: match, ← or ↑: indel)

Step 3. Check if the stop condition is satisfied at each of the left and right match positions near the k-mer. Extend positions until the stop condition is satisfied. (The stop condition will be explained later.)


S_L
stop
stop
S_M
--GCA--CGATAGA-TA-TACGAT---AATTAATAGCGAGAGGACAACGTG-GAGCAG---G--GGACC--GTACGTTTGACGAGCGTGCAACAG
TCG-AGT-G-T-GAC-AG-A-GATGGG-ATT-ATAGCGAGAGGACAACGTGCGAGCAGTTTGCATG-CCAGG---G---GACGAG-GTGCAACAG
S_R'
Extending direction
S_M'

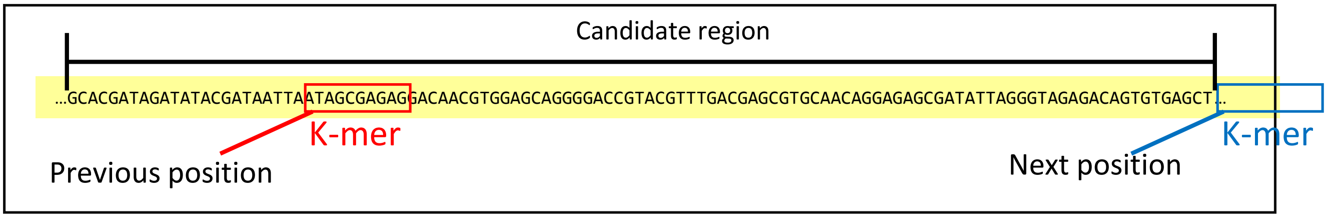
Step 4. Identify a candidate hairpin sequence.
(The boundary of Left and right arm is the match position before the stop position.)
(The boundary will be the last match position before the end of the sequence if there are no more positions to be checked.)


S_L
stop
Left-arm
stop
S_M
Loop
--GCA--CGATAGA-TA-TACGAT---AATTAATAGCGAGAGGACAACGTG-GAGCAG---G--GGACC--GTACGTTTGACGAGCGTGCAACAG
TCG-AGT-G-T-GAC-AG-A-GATGGG-ATT-ATAGCGAGAGGACAACGTGCGAGCAGTTTGCATG-CCAGG---G---GACGAG-GTGCAACAG
S_R'
Right-arm'
S_M'

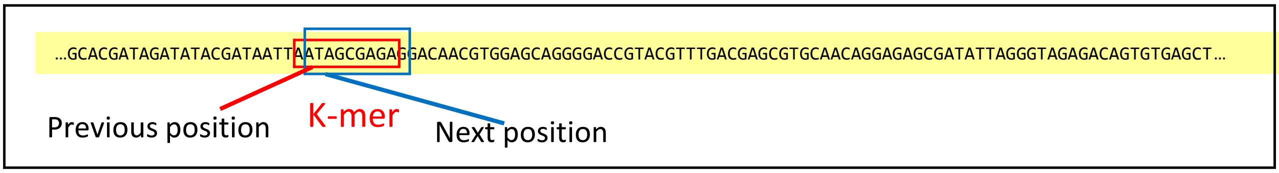
Step 5. Check if the candidate sequence satisfies the hairpin constraints. If satisfied, print out the hairpin sequence, the LCS sequence and the loop sequence. Otherwise, nothing.
(The hairpin constraints will be explained later.)

```
ATTAATAGCGAGAGGACAACGTGGAGCAGGGGACCGTACGTTTGACGAGGTGCAACAGCGAGAGCGACTATTA
ATTATAGCGAGAGGACAACGTGGAGCAG
GGGACCGTACGTTT
```

Step 6. Move the search position to out of candidate region, then continue.

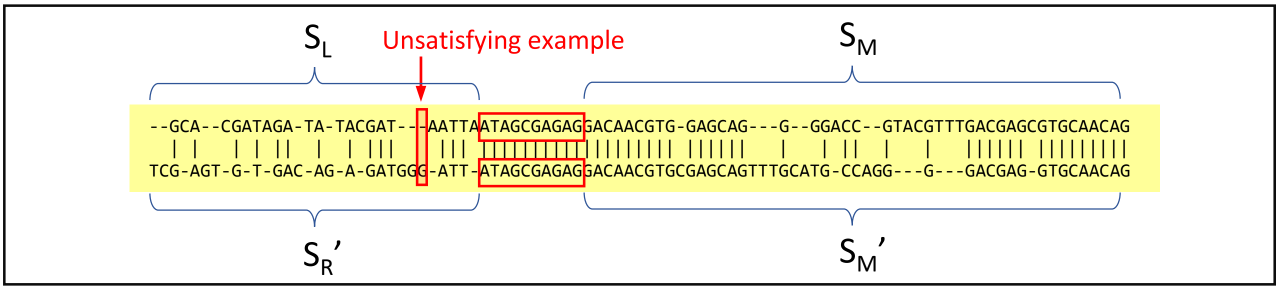

Candidate region
…GCACGATAGATATACGATAATTAATAGCGAGAGGACAACGTGGAGCAGGGGACCGTACGTTTGACGAGCGTGCAACAGGAGAGCGATATTAGGGTAGAGACAGTGTGAGCT…
K-mer
Previous position
K-mer
Next position

Case 2. When you does not find a reversed k-mer within a distance X.
(maximum expected loop length < X < maximum expected hairpin length)
Move the search position to the next character in the k-mer, then continue.

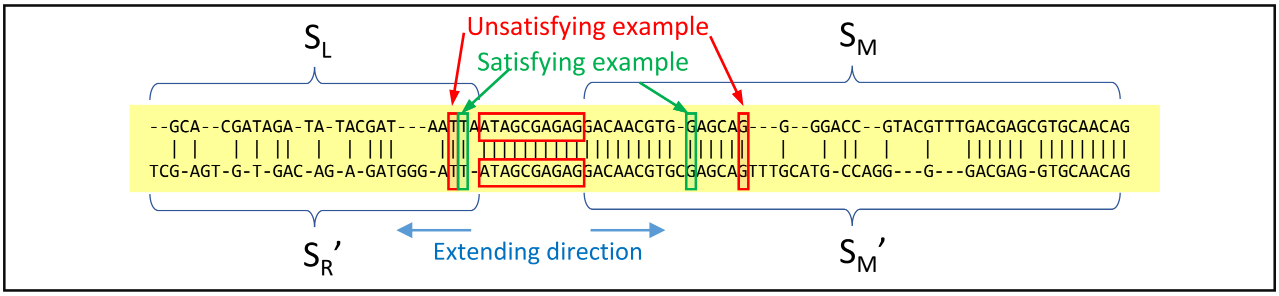

…GCACGATAGATATACGATAATTAATAGCGAGAGGACAACGTGGAGCAGGGGACCGTACGTTTGACGAGCGTGCAACAGGAGAGCGATATTAGGGTAGAGACAGTGTGAGCT…
Previous position
K-mer
Next position

STOP condition. If a position in an alignment satisfies all three conditions below, then it is a stop position.

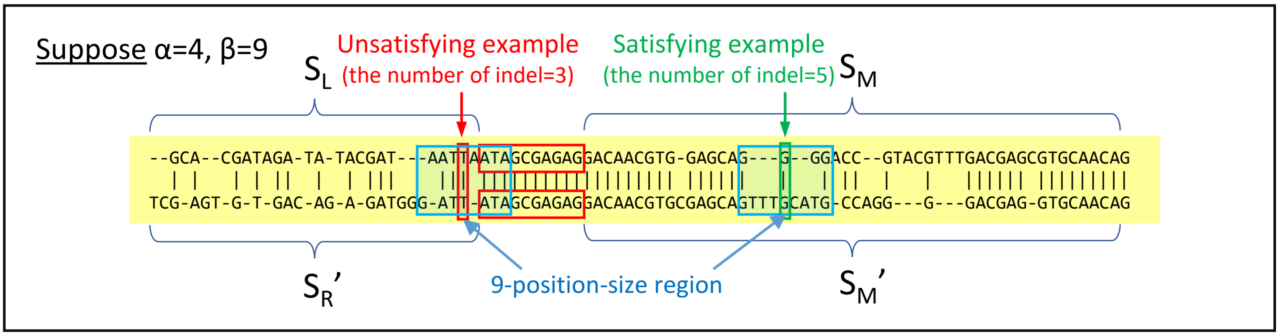
1) The position is the match position


S_L
Unsatisfying example
S_M
--GCA--CGATAGA-TA-TACGAT---AATTAATAGCGAGAGGACAACGTG-GAGCAG---G--GGACC--GTACGTTTGACGAGCGTGCAACAG
TCG-AGT-G-T-GAC-AG-A-GATGGG-ATT-ATAGCGAGAGGACAACGTGCGAGCAGTTTGCATG-CCAGG---G---GACGAG-GTGCAACAG
S_R'
S_M'

2) The previous position relative to the extending direction is the indel position.


S_L
Unsatisfying example
Satisfying example
S_M
--GCA--CGATAGA-TA-TACGAT---AATTAATAGCGAGAGGACAACGTG-GAGCAG---G--GGACC--GTACGTTTGACGAGCGTGCAACAG
TCG-AGT-G-T-GAC-AG-A-GATGGG-ATT-ATAGCGAGAGGACAACGTGCGAGCAGTTTGCATG-CCAGG---G---GACGAG-GTGCAACAG
S_R'
Extending direction
S_M'

3) The number of indel is greater than or equal to α within the β-position-size region.
(β is an odd number and β-region extends evenly in the both of left and right direction of the position)
(When you reaching near the end, consider the position outside of the sequence as match position to count indel)


Suppose α=4, β=9
S_L
Unsatisfying example
(the number of indel=3)
Satisfying example
(the number of indel=5)
S_M
--GCA--CGATAGA-TA-TACGAT---AATTAATAGCGAGAGGACAACGTG-GAGCAG---G--GGACC--GTACGTTTGACGAGCGTGCAACAG
TCG-AGT-G-T-GAC-AG-A-GATGGG-ATT-ATAGCGAGAGGACAACGTGCGAGCAGTTTGCATG-CCAGG---G---GACGAG-GTGCAACAG
S_R'
9-position-size region
S_M'

REMIND: You can simulate LCS on the website (http://lcs-demo.sourceforge.net/)
You can transform the LCS in an alignment form by back-tracing along the ARROW matrix.
You can find match positions and count indels by tracing arrow (↖: match, ← or ↑: indel).

Hairpin constraint. If a sequence satisfies all four conditions below, then it is a hairpin structure sequence.

1) A k-mer and the reversed k-mer are found within a distance X
(maximum expected loop length ≤ X ≤ maximum expected hairpin length)
2) The sequence is maximally extended from the k-mer and No stop position is found in the arm sequence region.
3) 0 ≤ Loop length ≤ maximum expected loop length (i.e. no loop is possible)
4) minimum expected hairpin length ≤ hairpin sequence length ≤ maximum expected hairpin length

HW3 Notice

1. The program will be tested using the uploaded long DNA sequence with modification of some characters. (the toy example sequence is not used)
2. The following six parameters and values are used to test programs in HW3.
    1) k=10
    2) minimum expected hairpin length = 200
    3) maximum expected hairpin length = 400
    4) maximum expected loop length = 50
    5) α = 4
    6) B = 9
3. The program should be completed within 10 mins because TA should test about 100 programs.
   (if it takes more than 10 mins, you may have no score.)