Tutorial Assignment 1

Dynamic Programming: Global, Local Alignment and BLAST

Class: BIM 3001

Name: 周喆媛

Student Number: 117010423

Date: 2020.03.08 (Sun.)

香 港 中 文 大 學（深 圳）

The Chinese University of Hong Kong, Shenzhen

① 1° Question 1 is actually global alignment.

v = CATGTATC ; w = CTATAATC

mismatch : -1 , match = 1 , indel = -3

		w	C	T	A	T	A	A	T	C
		0	1	2	3	4	5	6	7	8
V	0	0	← -3	← -6	← -9	← -12	← -15	← -18	← -21	← -24
C	1	↑ -3	↖ 1	← -2	← -5	← -8	← -11	← -14	← -17	← -20
A	2	↑ -6	↑ -2	↖ 0	↖ -1	← -4	↔ -7	↔ -10	← -13	← -16
T	3	↑ -9	↑ -5	↖ -1	↖ -1	↖ 0	← -3	← -6	↔ -9	← -12
G	4	↑ -12	↑ -8	↑ -4	↖ -2	↖ -2	↖ -1	↔ -4	↔ -7	↖ -10
T	5	↑ -15	↑ -11	↕ -7	↕ -5	↖ -1	↖ -3	↖ -2	↖ -3	← -6
A	6	↑ -18	↑ -14	↑ -10	↖ -6	↑ -4	↖ 0	↖ -2	↖ -3	↖ -4
T	7	↑ -21	↑ -17	↕ -13	↑ -9	↖ -5	↑ -3	↖ -1	↖ -1	↔ -4
C	8	↑ -24	↕ -20	↑ -16	↑ -12	↑ -8	↕ -6	↕ -4	↖ -2	↖ 0

result : v : C A T G T A T C

w : C T A T A A T C

2° Question 2 is local alignment

v = CATGTATC ; w = CTATAATC

{ match = +1
  mismatch = -1
  indel = -3

		w	C	T	A	T	A	A	T	C
		0	1	2	3	4	5	6	7	8
v	0	0	0	0	0	0	0	0	0	0
C	1	0	↖ 1	0	0	0	0	0	0	↖ 1
A	2	0	0	↖ 0	↖ 1	0	↖ 1	↖ 1	0	0
T	3	0	0	↖ 1	0	↖ 2	0	0	↖ 2	0
G	4	0	0	0	↖ 0	0	↖ 1	0	0	↖ 1
T	5	0	0	↖ 1	0	↖ 1	0	↖ 0	↖ 1	0
A	6	0	0	0	↖ 2	0	↖ 2	↖ 1	0	↖ 1
T	7	0	0	↖ 1	0	↖ 3	← 0	↖ 1	↖ 2	0
C	8	0	↖ 1	0	↖ 0	↑ 0	↖ 2	0	↖ 0	↖ 3

result 1 : v T A T
w T A T

result 2 : v = A T C
w = A T C

Question 2

First Part: implementation of local and global alignment

the code explanation is attached in comment, this part will mainly focus on the data structure function and results.

Program Design

Global Alignment

- Global table structure: 2-dimension array from NumPy, where the height equals to length of string v + 1, the width equals to the length of string w + 1. Each element is a binary tuple: the first element in the tuple is the value of the optimal score, the second element in the tuple is the possible directions to get the optimal value.
- Arrow direction notation: there are total seven possible results of the direction in each entry: left, up, slant, left + up, left + slant, up + slant, left + up + slant. We define number 1 stands for left, 2 stands for up, 3 stands for slant, and the further combination is the directly combo of numbers, for instance: left + slant denoted as 13.
- Backtracking: in order to find out all the paths, the backtracking is implemented by recursion. The start point of the recursion is the (len(v), len(w)). The return criteria is reaching the (0,0) point. Chopping with single direction in the entry is quite trivial, the problem lies on how to deal with multiple direction entries to find all the paths. For example, if in an entry, it can go both left and slant, than the recursion should be, without moving to next entry, apply recursion separately twice for left direction recursion and slant direction recursion.
- Backtracking result deal: the recursion won't return the exactly correct path. There are mainly two problems:
  firstly, each element is the accumulation of the former stack: for example, if v = TACGGGTAT, w = GGACGTACG, the first alignment and second alignment are listed as following picture:

```
['-', 'T', 'A', 'T', 'G', 'G', 'G', 'C', 'A', '-', 'T']
['G', 'C', 'A', 'T', '-', '-', 'G', 'C', 'A', 'G', 'G']
['-', 'T', 'A', 'T', 'G', 'G', 'G', 'C', 'A', '-', 'T', 'T', '-']
['G', 'C', 'A', 'T', '-', '-', 'G', 'C', 'A', 'G', 'G', 'G', 'G']
```

  To fix this problem, need to slice the list. The cut point is the length of the string minus the (current list length - former list length). Reconstruct the sequence as the [0, cut_point] + [different_part:end].
  Second problem is the list need to be reversed, which is quite simple by using reverse function for python list (unlike C, this is why people live python, it has everything!)

Local Alignment

- Local table structure: the frame of this table is the same as global, however, need to notice, when the score value is smaller or equal to zero, should change the score and direction in the tuple to zero, which denotes the termination.
- How to find the backtracking position?

Although Numpy provides the method to find the coordinates of the max value, the structure of our array is self-defined, the method is invalid for this case. Considering following, iterate the whole table and append the first element: score value to a specific list. Use built-in function max() to find the max value, iterate the list, if max value, note the index. The index for the one dimension array can be transformed to two-dim coordinate by using division and remainder

- The backtracking for local alignment don't need recursion and is quite simple, which applies the slant direction and stops when meets zero.

Results and Outputs

Example 1: from lec assignment 1 to show all paths

v = TACGGGTAT

w = GGACGTACG

match = +1; mismatch = -1; indel = -1

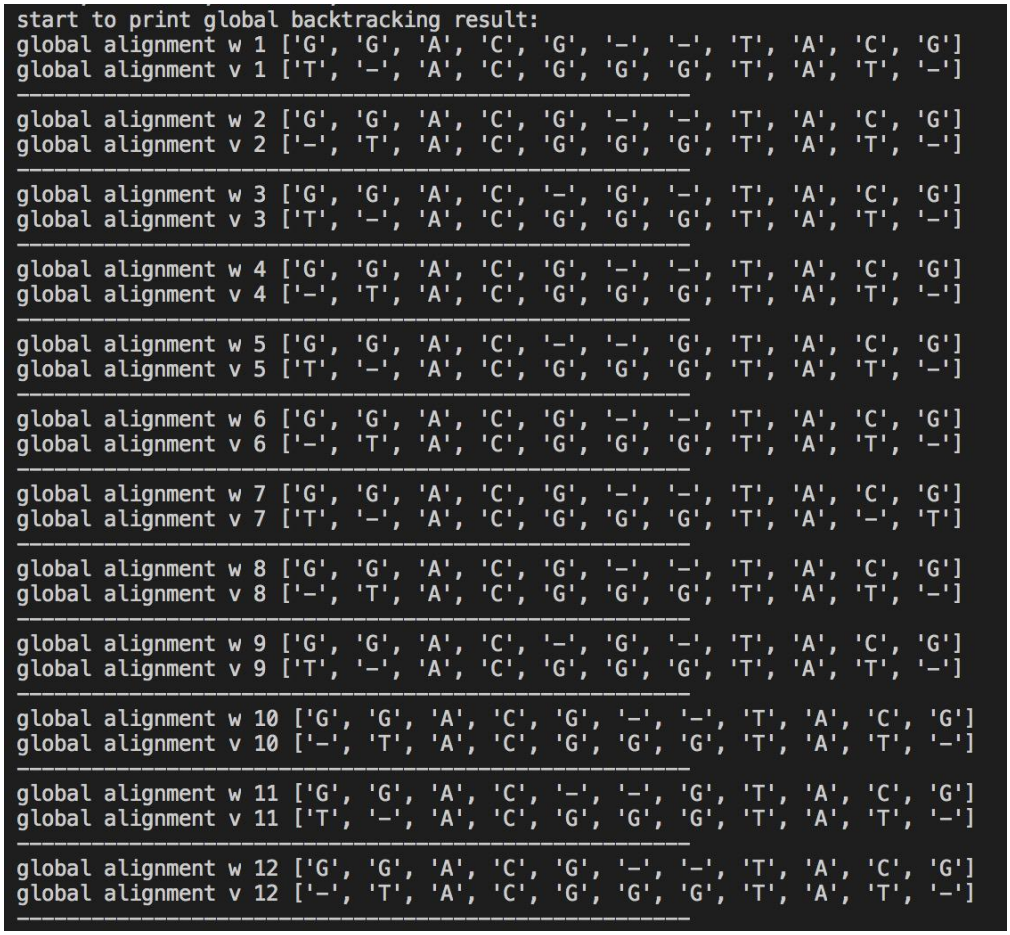
Figure 1: Initialize the Table

```
please choose the mode - l for local, g for global, q for quit: g
please input the v sequence: TACGGGTAT
please input the w sequence: GGACGTACG
start global alignment:
[[(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]]
```

Figure 2: Completed Global Table

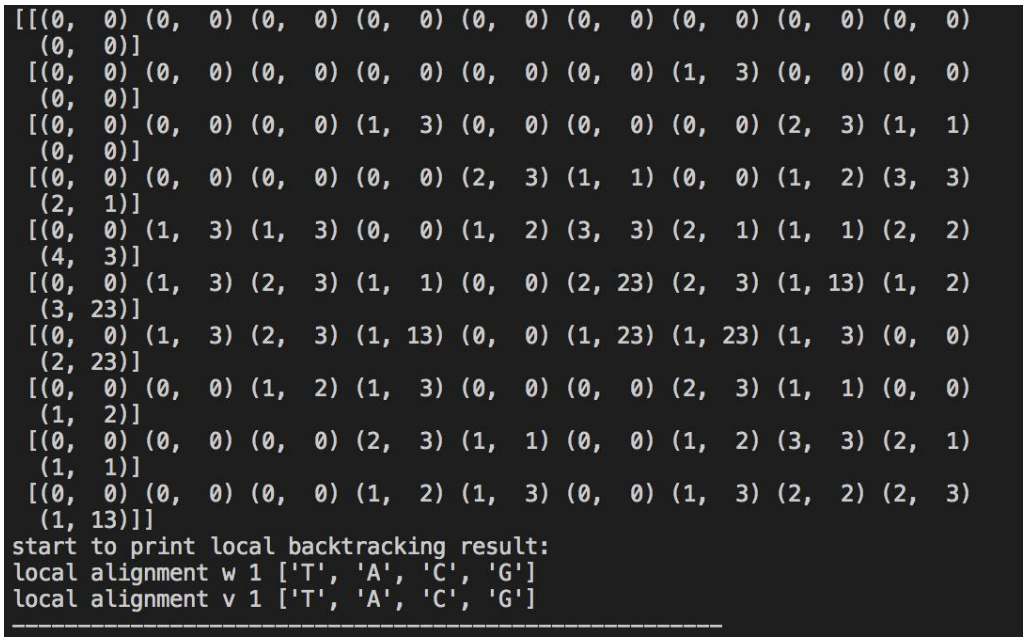
```
[[( 0,   0) (-1,   1) (-2,   1) (-3,   1) (-4,   1) (-5,   1) (-6,   1)
  (-7,   1) (-8,   1) (-9,   1)]
 [(-1,   2) (-1,   3) (-2,  13) (-3,  13) (-4,  13) (-5,  13) (-4,   3)
  (-5,   1) (-6,   1) (-7,   1)]
 [(-2,   2) (-2,  23) (-2,   3) (-1,   3) (-2,   1) (-3,   1) (-4,   1)
  (-3,   3) (-4,   1) (-5,   1)]
 [(-3,   2) (-3,  23) (-3,  23) (-2,   2) ( 0,   3) (-1,   1) (-2,   1)
  (-3,   1) (-2,   3) (-3,   1)]
 [(-4,   2) (-2,   3) (-2,   3) (-3,  12) (-1,   2) ( 1,   3) ( 0,   1)
  (-1,   1) (-2,   1) (-1,   3)]
 [(-5,   2) (-3,  23) (-1,   3) (-2,   1) (-2,   2) ( 0,  23) ( 0,   3)
  (-1,  13) (-2,  13) (-1,   3)]
 [(-6,   2) (-4,  23) (-2,  23) (-2,   3) (-3, 123) (-1,  23) (-1,  23)
  (-1,   3) (-2,  13) (-1,   3)]
 [(-7,   2) (-5,   2) (-3,   2) (-3,  23) (-3,   3) (-2,   2) ( 0,   3)
  (-1,   1) (-2,  13) (-2,   2)]
 [(-8,   2) (-6,   2) (-4,   2) (-2,   3) (-3,   1) (-3,   2) (-1,   2)
  ( 1,   3) ( 0,   1) (-1,   1)]
 [(-9,   2) (-7,   2) (-5,   2) (-3,   2) (-3,   3) (-4, 123) (-2,  23)
  ( 0,   2) ( 0,   3) (-1,  13)]]
```

Figure 3: All Global Alignments Results



```
start to print global backtracking result:
global alignment w 1 ['G', 'G', 'A', 'C', 'G', '-', '-', 'T', 'A', 'C', 'G']
global alignment v 1 ['T', '-', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 2 ['G', 'G', 'A', 'C', 'G', '-', '-', 'T', 'A', 'C', 'G']
global alignment v 2 ['-', 'T', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 3 ['G', 'G', 'A', 'C', '-', 'G', '-', 'T', 'A', 'C', 'G']
global alignment v 3 ['T', '-', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 4 ['G', 'G', 'A', 'C', 'G', '-', '-', 'T', 'A', 'C', 'G']
global alignment v 4 ['-', 'T', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 5 ['G', 'G', 'A', 'C', '-', '-', 'G', 'T', 'A', 'C', 'G']
global alignment v 5 ['T', '-', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 6 ['G', 'G', 'A', 'C', 'G', '-', '-', 'T', 'A', 'C', 'G']
global alignment v 6 ['-', 'T', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 7 ['G', 'G', 'A', 'C', 'G', '-', '-', 'T', 'A', 'C', 'G']
global alignment v 7 ['T', '-', 'A', 'C', 'G', 'G', 'G', 'T', 'A', '-', 'T']
--------------------------------------------------------
global alignment w 8 ['G', 'G', 'A', 'C', 'G', '-', '-', 'T', 'A', 'C', 'G']
global alignment v 8 ['-', 'T', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 9 ['G', 'G', 'A', 'C', '-', 'G', '-', 'T', 'A', 'C', 'G']
global alignment v 9 ['T', '-', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 10 ['G', 'G', 'A', 'C', 'G', '-', '-', 'T', 'A', 'C', 'G']
global alignment v 10 ['-', 'T', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 11 ['G', 'G', 'A', 'C', '-', '-', 'G', 'T', 'A', 'C', 'G']
global alignment v 11 ['T', '-', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
global alignment w 12 ['G', 'G', 'A', 'C', 'G', '-', '-', 'T', 'A', 'C', 'G']
global alignment v 12 ['-', 'T', 'A', 'C', 'G', 'G', 'G', 'T', 'A', 'T', '-']
--------------------------------------------------------
```

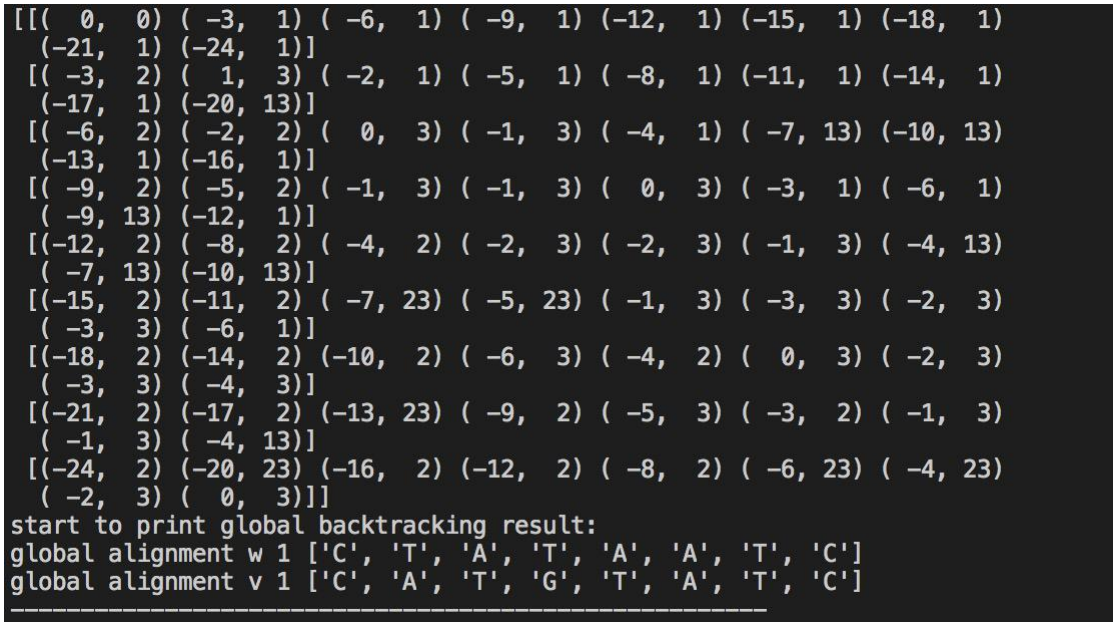
Figure 4: All Local Alignments Results



```
[[(0,  0) (0,  0) (0,  0) (0,  0) (0,  0) (0,  0) (0,  0) (0,  0) (0,  0)
  (0,  0)]
 [(0,  0) (0,  0) (0,  0) (0,  0) (0,  0) (0,  0) (1,  3) (0,  0) (0,  0)
  (0,  0)]
 [(0,  0) (0,  0) (0,  0) (1,  3) (0,  0) (0,  0) (0,  0) (2,  3) (1,  1)
  (0,  0)]
 [(0,  0) (0,  0) (0,  0) (0,  0) (2,  3) (1,  1) (0,  0) (1,  2) (3,  3)
  (2,  1)]
 [(0,  0) (1,  3) (1,  3) (0,  0) (1,  2) (3,  3) (2,  1) (1,  1) (2,  2)
  (4,  3)]
 [(0,  0) (1,  3) (2,  3) (1,  1) (0,  0) (2, 23) (2,  3) (1, 13) (1,  2)
  (3, 23)]
 [(0,  0) (1,  3) (2,  3) (1, 13) (0,  0) (1, 23) (1, 23) (1,  3) (0,  0)
  (2, 23)]
 [(0,  0) (0,  0) (1,  2) (1,  3) (0,  0) (0,  0) (2,  3) (1,  1) (0,  0)
  (1,  2)]
 [(0,  0) (0,  0) (0,  0) (2,  3) (1,  1) (0,  0) (1,  2) (3,  3) (2,  1)
  (1,  1)]
 [(0,  0) (0,  0) (0,  0) (1,  2) (1,  3) (0,  0) (1,  3) (2,  2) (2,  3)
  (1, 13)]]
start to print local backtracking result:
local alignment w 1 ['T', 'A', 'C', 'G']
local alignment v 1 ['T', 'A', 'C', 'G']
--------------------------------------------------------
```

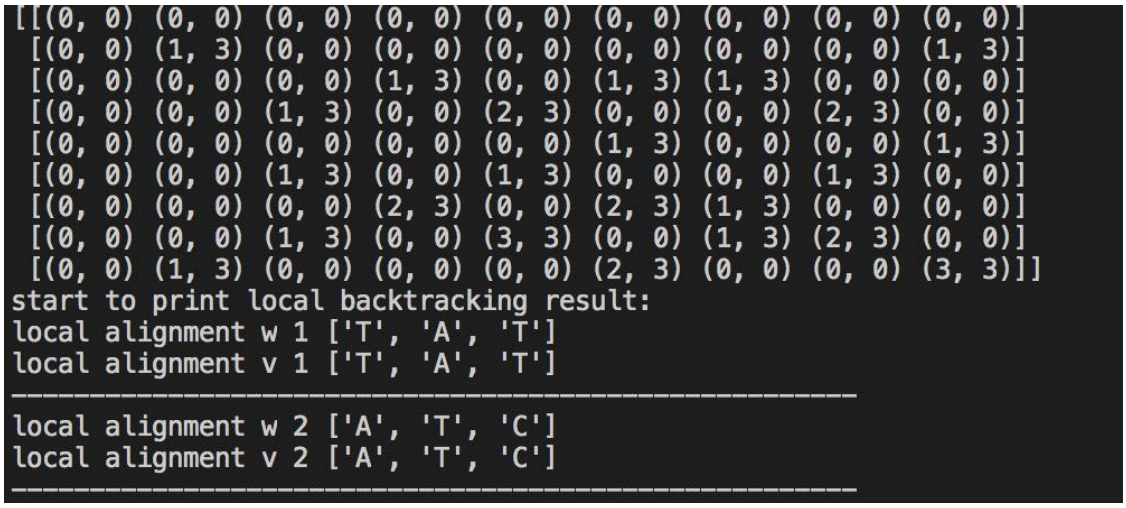
Example 2: Tutorial assignment question 1

Figure 5: Global Alignment



```
[[(  0,  0) ( -3,  1) ( -6,  1) ( -9,  1) (-12,  1) (-15,  1) (-18,  1)
  (-21,  1) (-24,  1)]
 [( -3,  2) (  1,  3) ( -2,  1) ( -5,  1) ( -8,  1) (-11,  1) (-14,  1)
  (-17,  1) (-20, 13)]
 [( -6,  2) ( -2,  2) (  0,  3) ( -1,  3) ( -4,  1) ( -7, 13) (-10, 13)
  (-13,  1) (-16,  1)]
 [( -9,  2) ( -5,  2) ( -1,  3) ( -1,  3) (  0,  3) ( -3,  1) ( -6,  1)
  ( -9, 13) (-12,  1)]
 [(-12,  2) ( -8,  2) ( -4,  2) ( -2,  3) ( -2,  3) ( -1,  3) ( -4, 13)
  ( -7, 13) (-10, 13)]
 [(-15,  2) (-11,  2) ( -7, 23) ( -5, 23) ( -1,  3) ( -3,  3) ( -2,  3)
  ( -3,  3) ( -6,  1)]
 [(-18,  2) (-14,  2) (-10,  2) ( -6,  3) ( -4,  2) (  0,  3) ( -2,  3)
  ( -3,  3) ( -4,  3)]
 [(-21,  2) (-17,  2) (-13, 23) ( -9,  2) ( -5,  3) ( -3,  2) ( -1,  3)
  ( -1,  3) ( -4, 13)]
 [(-24,  2) (-20, 23) (-16,  2) (-12,  2) ( -8,  2) ( -6, 23) ( -4, 23)
  ( -2,  3) (  0,  3)]]
start to print global backtracking result:
global alignment w 1 ['C', 'T', 'A', 'T', 'A', 'A', 'T', 'C']
global alignment v 1 ['C', 'A', 'T', 'G', 'T', 'A', 'T', 'C']
------------------------------------------------------------
```

Figure 6: Local Alignment



```
[[(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0)]
 [(0, 0) (1, 3) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (1, 3)]
 [(0, 0) (0, 0) (0, 0) (1, 3) (0, 0) (1, 3) (1, 3) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (1, 3) (0, 0) (2, 3) (0, 0) (0, 0) (2, 3) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (0, 0) (0, 0) (1, 3) (0, 0) (0, 0) (1, 3)]
 [(0, 0) (0, 0) (1, 3) (0, 0) (1, 3) (0, 0) (0, 0) (1, 3) (0, 0)]
 [(0, 0) (0, 0) (0, 0) (2, 3) (0, 0) (2, 3) (1, 3) (0, 0) (0, 0)]
 [(0, 0) (0, 0) (1, 3) (0, 0) (3, 3) (0, 0) (1, 3) (2, 3) (0, 0)]
 [(0, 0) (1, 3) (0, 0) (0, 0) (0, 0) (2, 3) (0, 0) (0, 0) (3, 3)]]
start to print local backtracking result:
local alignment w 1 ['T', 'A', 'T']
local alignment v 1 ['T', 'A', 'T']
------------------------------------------------------------
local alignment w 2 ['A', 'T', 'C']
local alignment v 2 ['A', 'T', 'C']
------------------------------------------------------------
```

Second Part: Virus Alignment Comparison

Spike protein lies on the virus envelope surface, there are virus membrane proteins arranged regularly, as if forming a certain morphological unit. Spike protein might be highly conserved if the viruses are homology, thus can be used to epidemic analysis.
According to the following definition, it is reasonable to apply BLAST to the three virus to discover the homology relationship.

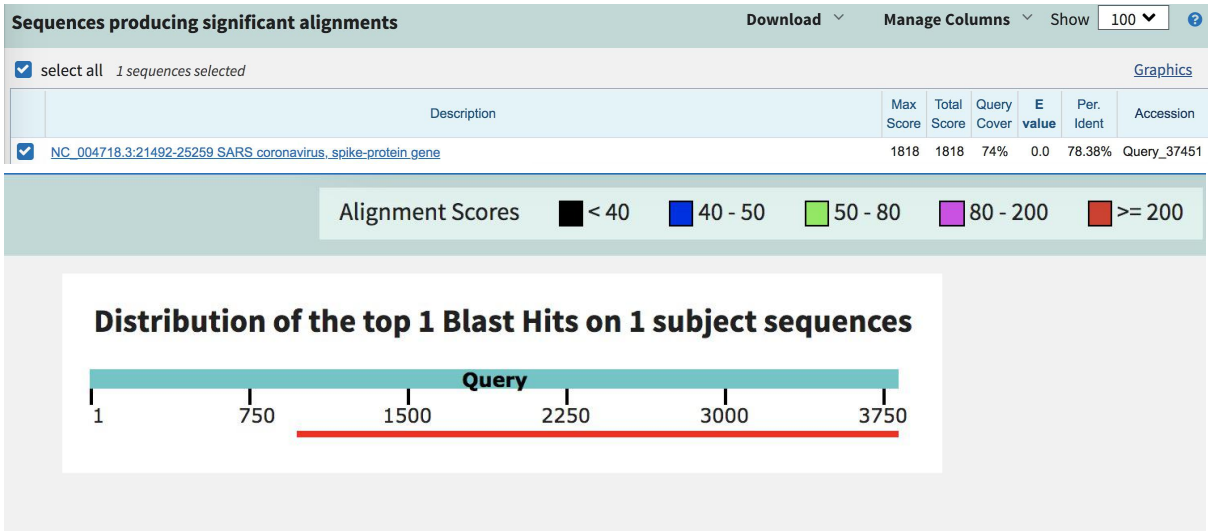
Wuhan Novel Coronavirus and SARS Coronavirus

Comment:

- tried two different scoring matrix, different scoring matrix will generate different alignment results the same pair. However, the difference is not large, about 5~9%.
- Wuhan Novel Coronavirus and SARS Coronavirus are quite similar in spike protein gene with about 78% conservation (even more similar in protein), which indicates the close biological relationship between two virus, which can be inferred that New Novel Coronavirus might be homology of SARS.
- The conserved region mainly lies on 750 - 3750 positions of the query.
- The E-value for the alignment results are extremely low, close to 0%, which indicates that the similarity is unlikely be generated by random.

1. Select match = 1, mismatch = -2, linear gap

Search Parameters	
Program	blastn
Word size	28
Expect value	10
Hitlist size	100
Match/Mismatch scores	1,-2
Gapcosts	0,2.5
Low Complexity Filter	Yes
Filter string	L;m;
Genetic Code	1



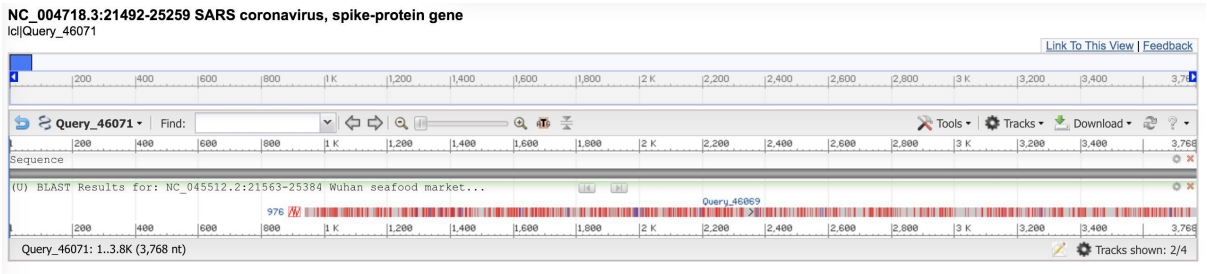
NC_004718.3:21492-25259 SARS coronavirus, spike-protein gene

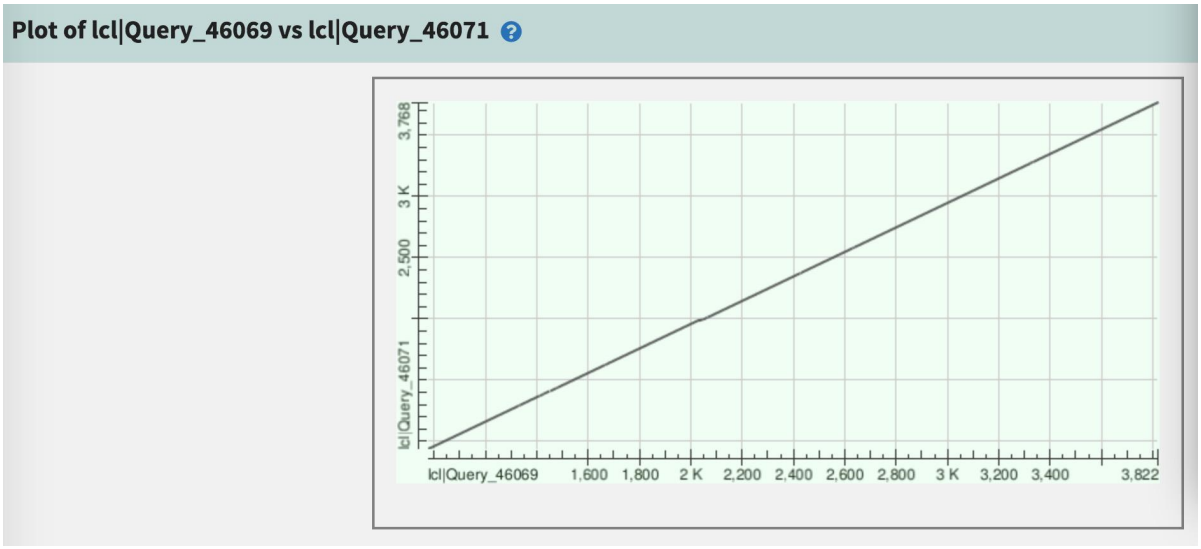
Sequence ID: Query_46071 Length: 3768 Number of Matches: 1

Range 1: 938 to 3768 Graphics ▼ Next Match ▲ Previous Match

Score	Expect	Identities	Gaps	Strand
1818 bits(984)	0.0	2240/2858(78%)	39/2858(1%)	Plus/Plus

```
Query  977   TTGTTAGATTTCCTAATATTACAAACTTGTGCCCTTTTGGTGAAGTTTTTAACGCCACCA  1036
              |||| ||||| ||||||||||||||||||||| |||||||| || |||||||| || || |
Sbjct  938   TTGTGAGATTCCCTAATATTACAAACTTGTGTCCTTTTGGAGAGGTTTTTAATGCTACTA  997

Query  1037  GATTTGCATCTGTTTATGCTTGGAACAGGAAGAGAATCAGCAACTGTGTTGCTGATTATT  1096
               |||  | ||||| ||||| ||| | || || | |||     || ||||||||||||| |
Sbjct  998   AATTCCCTTCTGTCTATGCATGGGAGAGAAAAAAAATTTCTAATTGTGTTGCTGATTACT  1057

Query  1097  CTGTCCTATATAATTCCGCATCATTTTCCACTTTTAAGTGTTATGGAGTGTCTCCTACTA  1156
              |||| || || || ||  |||  ||||| || |||||||| ||||| || ||| | ||||
Sbjct  1058  CTGTGCTCTACAACTCAACATTTTTTTCAACCTTTAAGTGCTATGGCGTTTCTGCCACTA  1117

Query  1157  AATTAAATGATCTCTGCTTTACTAATGTCTATGCAGATTCATTTGTAATTAGAGGTGATG  1216
              | || |||||||| |||||  | |||||||||||||||| |||||| | |  || ||||
Sbjct  1118  AGTTGAATGATCTTTGCTTCTCCAATGTCTATGCAGATTCTTTTGTAGTCAAGGGAGATG  1177

Query  1217  AAGTCAGACAAATCGCTCCAGGGCAAACTGGAAAGATTGCTGATTATAATTATAAATTAC  1276
              | || |||||||| || ||||| ||||||||    |||||||||||||||||||||| |
Sbjct  1178  ATGTAAGACAAATAGCGCCAGGACAAACTGGTGTTATTGCTGATTATAATTATAAATTGC  1237

Query  1277  CAGATGATTTTACAGGCTGCGTTATAGCTTGGAATTCTAACAATCTTGATTCTAAGGTT-  1335
              |||||||||| |  || || ||  | ||||||||| |||  ||  ||||| || |  |
Sbjct  1238  CAGATGATTTCATGGGTTGTGTCCTTGCTTGGAATACTAGGAACATTGATGCT-ACTTCA  1296

Query  1336  GGTGGTAATTATAATTACCTGTATAGATTGTTTAGGAA-GTCTAATCTCAAACCTTTTGA  1394
                |||||||||||||||     ||||| |   ||||  | | | || || |  || |||||
Sbjct  1297  ACTGGTAATTATAATTATAAATATAGGTATCTTAGACATGGC-AAGCTTAGGCCCTTTGA  1355
```





2. Select match = 1, mismatch = -1, affine gap penalty

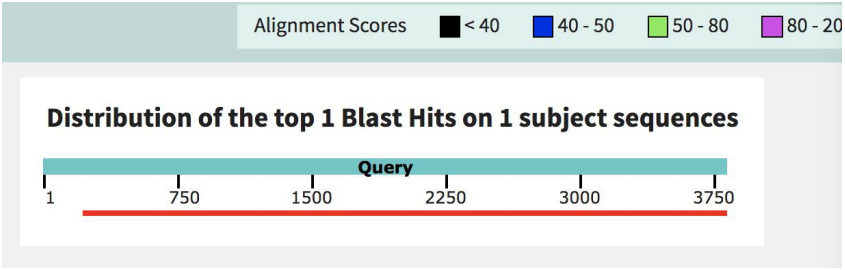
Search Parameters

Program	blastn
Word size	28
Expect value	10
Hitlist size	100
Match/Mismatch scores	1,-1
Gapcosts	5,2
Low Complexity Filter	Yes
Filter string	L;m;
Genetic Code	1

Sequences producing significant alignments Download Manage Columns Show 100

select all *1 sequences selected* Graphics

Description	Max Score	Total Score	Query Cover	E value	Per. Ident	Accession
NC_004718.3:21492-25259 SARS coronavirus, spike-protein gene	2621	2621	94%	0.0	74.19%	Query_64737



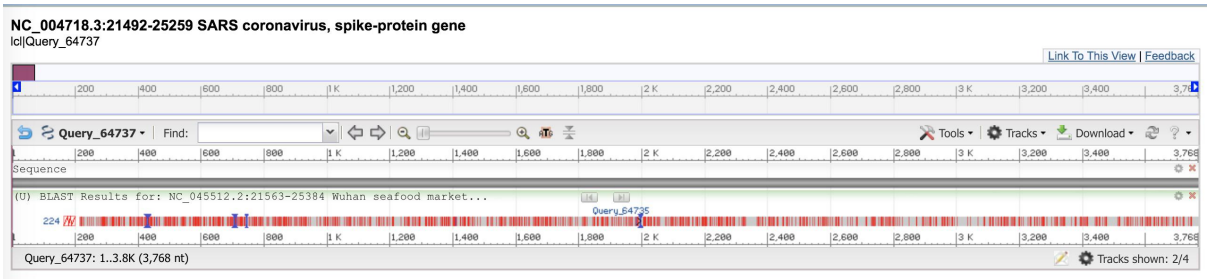
NC_004718.3:21492-25259 SARS coronavirus, spike-protein gene

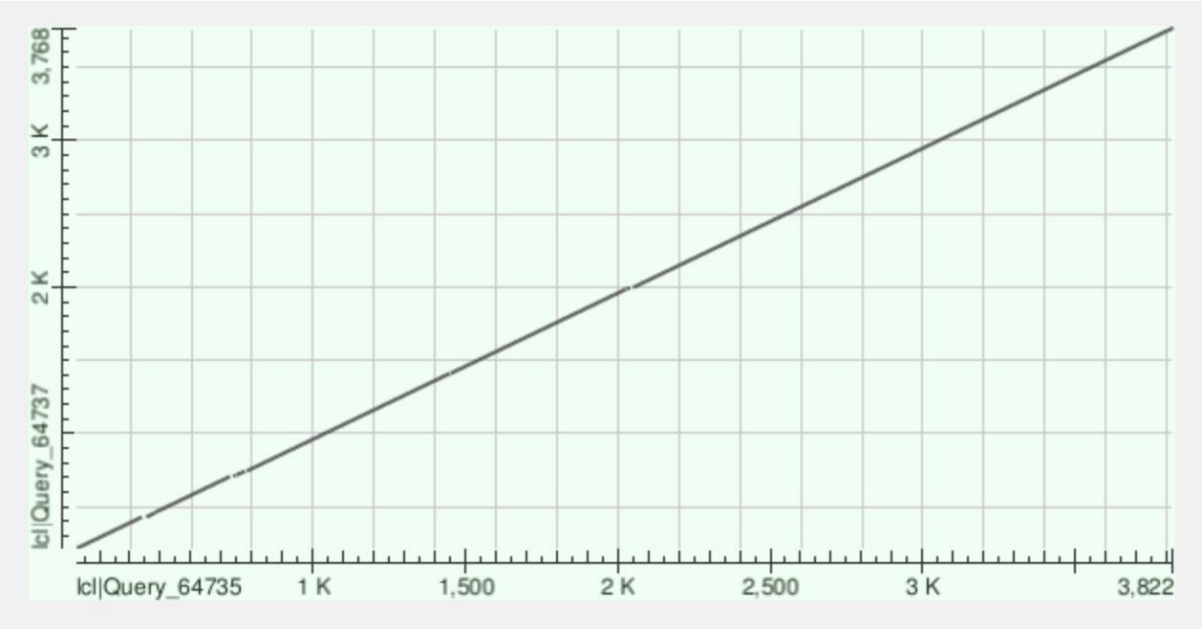
Sequence ID: Query_64737 **Length:** 3768 **Number of Matches:** 1

Range 1: 216 to 3768 Graphics Next Match Previous Match

Score	Expect	Identities	Gaps	Strand
2621 bits(1653)	0.0	2671/3600(74%)	49/3600(1%)	Plus/Plus

```
Query  225  TACTAAGAGGTTTGATAACCCTGTCCTACCATTTAATGATGGTGTTTATTTTGCTTCCAC  284
            ||  |  |  ||||||  ||||||||||  ||||  |||||  ||||||  ||||||||||||  ||||
Sbjct  216  TAATCATACGTTTGGCAACCCTGTCATACCTTTTAAGGATGGTATTTATTTTGCTGCCAC  275

Query  285  TGAGAAGTCTAACATAATAAGAGGCTGGATTTTTGGTACTACTTTAGATTCGAAGACCCA  344
             |||||  ||  ||   |   |   |  ||  |||  ||||||||  ||||   |   |     |||  |  ||
Sbjct  276  AGAGAAATCAAATGTTGTCCGTGGTTGGGTTTTTGGTTCTACCATGAACAACAAGTCACA  335

Query  345  GTCCCTACTTATTGTTAATAACGCTACTAATGTTGTTATTAAAGTCTGTGAATTTCAATT  404
            |||   |   |||||  ||||  ||   ||||||||||||||||    ||   |||  |  |||  ||||
Sbjct  336  GTCGGTGATTATTATTAACAATTCTACTAATGTTGTTATACGAGCATGTAACTTTGAATT  395

Query  405  TTGTAATGATCCATTTTTGGGTGTTTATTACCACAAAAACAACAAAAGTTGGATGGAAAG  464
             |||  |   |  ||  ||  ||  |  |||||  |  |  |  ||                |||  |   |  ||
Sbjct  396  GTGTGACAACCCTTTCTTTGCTGTTTCTAAACCCA--------------TGGGTACACAG  441

Query  465  TGAGTTCAGAGTTTATTC--TAGTGCGAATAATTGCACTTTTGAATATGTCTCTCAGCCT  522
              |   |    |   ||||||   ||  |||    ||||||||||||  ||  ||   |  |||  |   |
Sbjct  442  ACACATACTATGATATTCGATAATGCATTTAATTGCACTTTCGAGTACATATCTGATGCC  501

Query  523  TTTCTTATGGACCTTGAAGGAAAACAGGGTAATTTCAAAAATCTTAGGGAATTTGTGTTT  582
            |||     |  ||   ||   ||  |||     |||||||| |||  |   |   |  ||  |||||||||
Sbjct  502  TTTTCGCTTGATGTTTCAGAAAAGTCAGGTAATTTTAAACACTTACGAGAGTTTGTGTTT  561
```





Wuhan Novel Coronavirus and MERS Coronavirus

Comment:

- With the same scoring matrix as the former analysis, the significant similarity can not be found between Wuhan Novel Coronavirus and MERS Coronavirus. This indicates that it is basically impossible for New Novel Coronavirus and MERS be homology.

3. Select match = 1, mismatch = -2, linear gap

Search Parameters	
Program	blastn
Word size	28
Expect value	10
Hitlist size	100
Match/Mismatch scores	1,-2
Gapcosts	0,2.5
Low Complexity Filter	Yes
Filter string	L;m;
Genetic Code	1

Job Title	**NC_045512.2:21563-25384 Wuhan seafood market...**
RID	6BRWYYCR114 *Search expires on 03-10 14:18 pm* Download All
Program	Blast 2 sequences Citation
Query ID	lcl|Query_22229 (dna)
Query Descr	NC_045512.2:21563-25384 Wuhan seafood market pneumoni ...
Query Length	3822
Subject ID	lcl|Query_22231 (dna)
Subject Descr	NC_019843.3:21456-25517 Middle East respiratory syndrome ...
Subject Length	4062

⚠ No significant similarity found. For reasons why,click here

4. Select match = 1, mismatch = -1, affine gap penalty

Search Parameters	
Program	blastn
Word size	28
Expect value	10
Hitlist size	100
Match/Mismatch scores	1,-1
Gapcosts	5,2
Low Complexity Filter	Yes
Filter string	L;m;
Genetic Code	1

Job Title	**NC_045512.2:21563-25384 Wuhan seafood market...**
RID	6BRZ1C5111N *Search expires on 03-10 14:19 pm* Download All
Program	Blast 2 sequences Citation
Query ID	lcl|Query_38903 (dna)
Query Descr	NC_045512.2:21563-25384 Wuhan seafood market pneumoni ...
Query Length	3822
Subject ID	lcl|Query_38905 (dna)
Subject Descr	NC_019843.3:21456-25517 Middle East respiratory syndrome ...
Subject Length	4062

⚠ No significant similarity found. For reasons why,click here

Additional: SARS Coronavirus and MERS Coronavirus
Comment:
To confirm the result that Wuhan Novel Coronavirus and MERS Coronavirus can not be homology, apply BLAST to SARS Coronavirus and MERS Coronavirus again. If the former result is correct, then the SARS and MERS can not be homology, either.
And from the alignment result, the conclusion is quite correct .
Furthermore, change the searching mode to *"Somewhat Similar Pairwise",* it can be seen clearly, SARS and MERS share extremely low similarity.

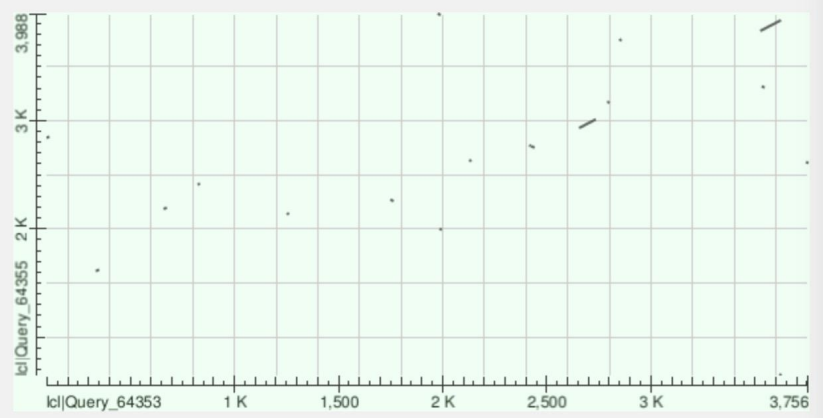
Highly similar mode: Select match = 2, mismatch = -3, linear gap penalty

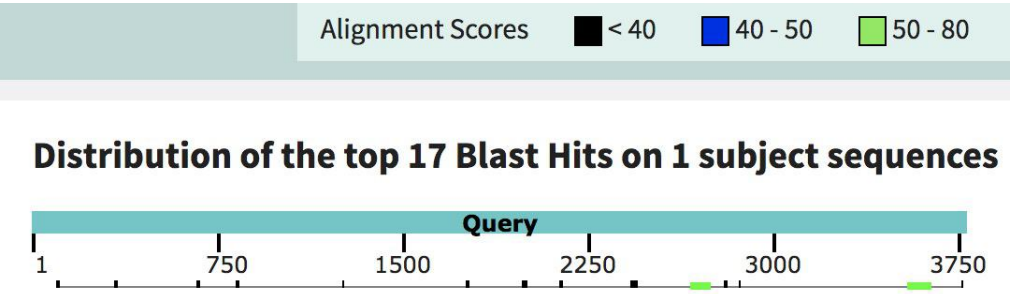
Job Title	**NC_004718.3:21492-25259 SARS coronavirus,...**
RID	6BS4NCGB11N *Search expires on 03-10 14:22 pm* Download All
Program	Blast 2 sequences Citation
Query ID	lcl|Query_29395 (dna)
Query Descr	NC_004718.3:21492-25259 SARS coronavirus, spike-protein gene
Query Length	3768
Subject ID	lcl|Query_29397 (dna)
Subject Descr	NC_019843.3:21456-25517 Middle East respiratory syndrome ...
Subject Length	4062

⚠ No significant similarity found. For reasons why,click here

Somewhat Similar Mode: Select match = 2, mismatch = -3, linear gap penalty

Search Parameters	
Program	blastn
Word size	11
Expect value	10
Hitlist size	100
Match/Mismatch scores	2,-3
Gapcosts	5,2
Low Complexity Filter	Yes
Filter string	L;m;
Genetic Code	1





Third Part: Local Alignment for Long Seq. & Comparison

Result and Output

Figure 7: Local Alignment

```
start to print local backtracking result:
local alignment w 1 ['T', 'A', 'A', 'A', 'C', 'C', 'T', 'C', 'A', 'T', 'G', 'T', 'G', 'C', 'T', 'T', 'G', 'A', 'A', 'G', 'A
', 'T', 'C', 'C', 'T', 'T', 'G', 'T', 'A', 'A', 'G', 'G', 'T', 'A', 'C', 'A', 'A', 'C', 'A', 'C', 'T', 'A', 'G']
local alignment v 1 ['T', 'A', 'A', 'A', 'C', 'C', 'T', 'C', 'A', 'T', 'G', 'T', 'G', 'C', 'T', 'T', 'G', 'A', 'A', 'G', 'A
', 'T', 'C', 'C', 'T', 'T', 'G', 'T', 'A', 'A', 'G', 'G', 'T', 'A', 'C', 'A', 'A', 'C', 'A', 'C', 'T', 'A', 'G']
```

Differences between Global Alignment and Local Alignment

- The result of global alignments is sub-sequences of the original sequences; the result of local alignment is sub-string of the original sequences.
- Global alignment focus on finding the path between vertices (0, 0) and (n, m) in the edit graph, but the local alignment wants to find paths between arbitrary vertices (i, j) and (i', j')
- The different purpose leads to the differences between algorithm: the global alignment can be denoted as, the table records the exactly optimal value even it is negative:

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

However, local alignment changes the algorithm by adding zero, which ensures each entry in the table is larger or equal to zero. This zero works as a predecessor and terminator.

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

- Because of different algorithm, the backtracking method is also different.
  Global alignment starts from (n,m) point and follows the arrow direction all the way to the (0, 0) point. In order to find every situation, it usually uses recursion to implement the backtracking for small size sequences.
  However, the backtracking of local alignment starts from the max score entry and only goes to slant arrow direction all the way to zero, which is understandable since the local

alignment wants to find the longest sub-string. Since each max score entry only corresponds to one alignment, once finish the table, the number of different local alignments is implicitly determined. Therefore, when considering algorithm for local alignment, people tend to choose iteration instead of recursion.

- Finally, the time efficiency for two alignments is different. To complete the whole table, both local alignment and global alignment need O(nm). However, different backtracking methods indicate different time efficiency. For local alignment, iteration takes O(nm), backtracking max time O(k(n^2+m^2)^(-1/2)); for global alignment, if we apply recursion in the backtracking, it costs exponentially.