Algorithm Assignment

Local Alignment

Due : 12/9

목표

수업시간에 배운 LCS(Longest Common sequence)를 바탕으로 Local Alignment를 구현하여 보자.

Local Alignment란?

생물학에서 두 유전자 서열을 나타내는 string을 align하는 문제를 LCS 문제로 이해할 수 있다. 특히, LCS는 Global alignment를 구하는 문제에 해당하는데, 여기서는 두 서열 전체를 가장 많이 일치하도록 align하는 optimal solution을 찾는다. 하지만 실제로 유전자 서열 사이의 유사성을 알아보기 위해 alignment를 할 때는 전체적으로 더 많은 수의 matched character를 찾는 것보다는 국지적으로라도 서로 잘 맞는 부분을 찾는 것이 유용할 때가 더 많다. 예를 들면 아래 그림에서 위에 나타낸 global

alignment는 전체적으로 더 많은 match를 포함하지만 아래에 나타낸 local alignment처럼 특정한 영역에서의 match가 훨씬 좋게 되는 alignment를 선호한다.

```
GCC-C-AGT--TATGT-CAGGGGGCACG--A-GCATGCAGA-
GCCGCC-GTCGT-T-TTCAG----CA-GTTATG--T-CAGAT
```

```
---G----C-----C--CAGTTATGTCAGGGGGCACGAGCATGCAGA
GCCGCCGTCGTTTTCAGCAGTTATGTCAG-----A------T-----
```

이를 위해서 alignment 에서 match, mismatch, gap의 각 경우에 대해 score 및 penalty를 정의한다. 아래와 같은 scoring matrix가 입력으로 주어졌을 때 local alignment를 위한 recurrence를 정의하고 이를 구현해 보시오. 위의 예시는 Global alignment에서 나쁜 점수를 받는 alignment라고 해도 local alignment로 보면 좋은 결과를 포함할 수 있다는 것을 설명하기 위한 것이고, 실제로 위의 예제에 대해 아래에 제시된 scoring matrix와 출력 형식에 따라 local alignment 를 수행한다면 아래와 같은 출력값을 갖는다.

```
CAGTTATGTCAG
CAGTTATGTCAG
```

Score(및 Penalty)

Local alignment를 위한 score(및 penalty)는 아래와 같이 3가지로 정의 할 수 있다.

1. **Match**
   두 character가 같을 때 주어지는 score
2. **Mismatch**
   두 character가 다를 때 주어지는 penalty
3. **Gap**
   두 Character를 Match나 Mismatch로 대응시키는 대신, Character를 공백으로 건너 뛰는 것을 Gap이라 한다.

Score Matrix

	A	**C**	**T**	**G**	**- (gap)**
A	5	-4	-4	-4	-10

C		5	-4	-4	-10
T			5	-4	-10
G				5	-10

예시는 아래와 같다.

S1 :TACAACTGTTGACTGGTTAC

S2 :TTAACTGCAACTGGTTCC

위의 두 String이 입력으로 주어졌을 때, 여러 Substring을 대상으로 local alignment를 해 볼 수 있는데, 하나의 예를 들면 아래와 같다.

S1 Substring : ACTGTT

S2 Substring : ACTGGTT

A	C	T	G	-	T	T
A	C	T	G	G	T	T

Score : 6 Match + 0 Mismatch + 1 Gap

= 6*(5) + 0*(-4) + 1*(-10) = 20

하지만 위의 Substring 보다 score가 더 높은 Substring 사이의 alignment가 존재하기 때문에, 위의 예시는 우리가 원하는 Local Alignment의 결과는 아니다. S1과 S2의 Substring중 score가 장 높은 Substring은 아래와 같다.

Local alignment 결과

S1 Substring : AACTGTTGACTGGTTAC

S2 Substring: AACTGCAACTGGTTCC

A	A	C	T	G	T	T	G	A	C	T	G	G	T	T	A	C
A	A	C	T	G	-	C	A	A	C	T	G	G	T	T	C	C

Score : 13 Match + 3 Mismatch + 1 Gap

= 13*(5) + 3*(-4) + 1*(-10) = 43

DP Backtrack시 우선 순위

DP Table을 채우고 난 후, Substring을 구하는 과정에서 동일한 점수를 나타내는 경로가 여럿 있을 수 있는데, tie break을 할 때 아래와 같은 탐색 우선순위를 적용해서 일정한 결과를 출력하도록 한다.

1. Match/Mismatch
2. DB의 Gap
3. Query의 Gap

입력

1. **DB와 Query file의 절대 경로와 output file의 경로를 argument로 받는다.**
2. **DB와 Query의 Data format**
   DB와 Query의 format은 첫줄에 '>name'와 같은 형식으로 이름이 나오고, 그 다음줄부터 파일 끝까지 name에 해당하는 sequence가 나열된다. 예시는 아래와 같다.

```
>MT dna:chromosome chromosome:AnoCar2.0:MT:1:17223:1 REF
GTTATTGTAGCTTACAAATTAAAGCACGGCACTGAAAATGCCACGATGGGAAATTATGAT
TCCCCAAAAACATAAAGTTTTGGTCTTAAACTTTCTGTTAATTTTAACCAAAATTATACA
TGCAAGTATCCGCCCACCAGTGAAAATGCCCACTTAATATTAGGAGCAGGTATCAGGCAC
AAACTATTTAGCCTAAAACACCTTGCTATGCCACACCCCCACGGGTATTCAGCAGTAATA
AAAATTAAGCCATAAGCGACAAATTATTTTAACAAGCTAGACTTAGTTATGGTTTTAAGG
GCCGGTCAATCTCGTGCCAGCCACCGCGGTTATACGAAAGGCCCAAAATAACAGACATCG
GCGTAAATAGTGACTAAAATATTACTTAAAAAATAAGAAATAAACAACAATCTAAAAGTA
AAATTAAGAAAATATGAAATCTCCCCTCTTAAATAAAAGTAATTTGATTCACGAAAGCTG
GGAAACAAACTAGGATTAGATACCCTACTATGCCTAGCCATTAACTGACACCCATTAACT
AAAGTGTTCGCCAGAATATTACGGGCGAAAAGCCTAAAACTCAAAAGACTTGGCGGTGCT
TCATACTCACCTTAGAGGAGCCTGTCCTATAATCGATACTCCACGATATACCCGACTACT
TTTTGCAAACTCAGCCTATATACCGCCGTCGCCAACTTACCCTGTGAAGGAAAAATAGTA
GGTAAAAAAGTCTACCACACAAACGTCAGGTCAAGGTGTAGCTTATAAAGTAGAAGAGGT
GGGCTACATTTTTTAAACAAAACACTACGACATAGTGCCCTGAAACTAGCACTTAAAGGA
GGATTTAACAGTAAAATAAGAAAGAAAACTTATTTTAAATTTGCTCTGAAGCGCGCACAC
ACCGCCCGTCACCCTCCTCACAAACAACTTAAAATACATAATAAAACAAAACAAAAAGAT
GAGGTAAGTCGTAACATGGTAAGCGCACTGGAAAGTGTGCTTAGATAACAAAAAGTAGCT
TAAAACAAAGCATTTAACTTACAATTAAACCATGTTAGAAAGTCTAACCTTTTTGCGCAA
AACAACAACCTAATTAAACAAACAAAATAGCACACACAACAAAACAAACCATTTGACAAA
AAAAGTAGAGGCGATCGAACCTTAACCGTAGAGTTTTATTGATAGTACCGCAAGGGAAAA
ATGAAATAATAGTGAAAACAAAAGCAAAACATAGCAAAGACTTCCCCTTGTACCTCTTGC
ATCACGATCTAGCAAGCATAAAACAAGCAAGAAGAAAACACAGCCTGTTTCCCCGAACTC
AAGTGAGCTATTTTAAAGCAACTAAAGAGTTAACCCGTCCCTGTAGCAAAAGGGTGGGAA
GACTTTAAAATAGAGGTGAAAAGCCAACCGAACTTGCTGATAGCTGGTTACCAGATAGAC
GAATTTTAGTTCAACTTAAAACTTTATTAACCCGCCCTTAGTAAAAACTTAGTTTTTAAG
ATACTCAATGAGGGGACAGCCTTATTGAGCCAGAATACAGCCTGAACTAGAGAGAAACCT
CAACAAAAACAACAGTAGGCCTTAAAACAGCCATCTAATATAATAACGTCGCAGTCTTCA
TAATCAAAATACCAACCGCACTTTAAAACTCCTATACTACCACCTGGTAATTCAATAAAT
```

출력

Output file 이름은 입력으로 받은 DBName_QueryName.txt로 하고, Local alignment 결과를 아래와 같은 형식으로 출력한다. (출력 예시는 Test 폴더 아래의 DB와 query를 사용해서 얻은 Test_answer를 참고)

DB : file name
Query : file name

<Alignment>

DB sequence

줄바꿈

Query sequence
(Test_answer의 출력 예시 참고)

제출

1. 보고서

a. TestDB를 DB로 사용하고 TestQuery# 을 query로 사용하였을 때, query의 길이 변화에 따른 수행시간, memory 사용량을 비교하여 분석하여 본다.
b. 3가지 생물의 서열이 ____MT.fa 파일로 각각 주어졌다. 이 세 종류의 서열을 서로 locally align 한 결과를 제출한다. 즉, 세 개의 서열 중 하나를 DB로, 다른 하나를 query로 사용해서 local alignment를 수행한 결과를 모두 구한다. 또한 수행시간과 memory 사용량을 비교하여 분석해 본다.
c. a와 b에서 구현한 recurrence(점화식)을 작성하여 제출한다.
d. GitLab 에 제출

2. 소스 코드 (Java 또는 C++로 구현)

- 컴파일 되지 않을 시 감점
- coding convention 참고하고 최대한 지킬 것
- inline document 없으면 0점