Introduction

HW#1에서는 genome에서 frequent한 k-mer를 찾아내는 작업을 수행했습니다. HW#2에서는 이 작업을 확장하여 두 genome 간의 alignment를 수행합니다.

- Frequent k-mer는 alphabet처럼 사용되어 genome sequence를 단순화합니다.
- Dynamic programming 기반의 알고리즘을 구현해 두 genome의 Longest Common Subsequence (LCS)를 찾아냅니다.

Task Instructions

- **Frequent k-mer 계산:**

  Genome으로부터 frequent한 k-mer를 추출하기 위해, 제시된 Python 코드를 사용하세요.

  실행 명령어는 다음과 같습니다:

  $ python cnt_kmer.py [k] [fasta file]

  실행 예시:

  $ python cnt_kmer.py 17 NZ_CP009072.fasta

  출력 결과 예시:

```
GTAGGCCGGATAAGGCG,43
TAGGCCGGATAAGGCGT,40
AGGCCGGATAAGGCGTT,39
TTCACGCCGCATCCGGC,32
CGTCTTATCAGGCCTAC,31
GCGTCTTATCAGGCCTA,31
GGCGTTCACGCCGCATC,31
AAGGCGTTCACGCCGCA,30
AGGCGTTCACGCCGCAT,30
CGTTCACGCCGCATCCG,30
GCGTTCACGCCGCATCC,30
GTTCACGCCGCATCCGG,30
TAAGGCGTTCACGCCGC,30
TGTAGGCCGGATAAGGC,30
ATAAGGCGTTCACGCCG,29
GATAAGGCGTTCACGCC,29
```

  ...

```
TTTTTCCTGAGCTCACG,5
TTTTTTAATATTTCATT,5
TTTTTTCATTATGGCAC,5
TTTTTTCCAGTATCGAT,5
TTTTTTTCATTATGGCA,5
```

- **Dynamic Programming을 활용한 Alignment:**

  두 genome sequence (예시)

  - genome1: ···AGT···CGA···TCG···AGT···ACG···ATC···GTT···CGA···
  - genome2: ···CGA···GTT···CGA···TCG···AGT···CGA···AGT···TCG···

  1. Frequent k-mer 추출 (예시: k=3)

  제시된 Python 코드 cnt_kmer.py를 실행한 결과, 두 genome에서 가장 빈도가 높은 k-mer를 추출했다고 가정합니다.

  - genome1_3mer_top1000.txt:

    ```
    CGA
    TCG
    AGT
    ATC
    ⋮
    ```

  - genome2_3mer_top1000.txt:

    ```
    CGA
    TCG
    AGT
    GAG
    ⋮
    ```

  2. k-mer 필터링

  두 genome을 align하기 위해서는 k-mer 필터링이 필요합니다.

  이 필터링은 다음과 같은 기준에 따라 진행됩니다:

  (1) A k-mer should occur reasonably frequently, not too frequent or not unique.
  (2) k-mers should be distinct, non-overlapping.

  Too frequent → False positive
  Too few → False negative

  그렇다면 overlapping과 frequent를 정의할 필요가 있습니다.

  - 예시

    Overlapping

    - k-mer가 이미 선정된 k-mer와 절반 이상 겹치면 제거합니다.
    - 예시:
      다음 데이터를 고려하면:

ATGAC: 20
TGCTT: 12
GACAG: 10

TGCTT는 ATGAC와 절반 이상 겹치지 않으므로 유지됩니다.
GACAG는 ATGAC와 절반 이상 겹치므로 제거됩니다.

Too Frequent

- Genome의 크기와 무작위 발생 확률을 고려하여, 빈도가 기준치를 초과하는 k-mer를 제거합니다.
- 예시:

  $N = 1\text{Mbp}$ (genome 크기), $k = 10$, $\alpha = 5$인 경우,

  $$\text{무작위 발생 빈도} = \frac{1{,}000{,}000}{4^{10}} \approx 0.95$$

  $$\text{기준 빈도} = 5 \times 0.95 \approx 4.75$$

  즉, $k = 10$인 k-mer가 genome 내에서 5회 이상 나타나면 "too frequent"로 판단하여 제거합니다.

○ 필터링 후(예시):

- genome1: AGT, CGA, TCG, …
- genome2: CGA, TCG, AGT, …

3. k-mer의 발생 위치 검색

두 genome의 서열에서 추출된 k-mer의 발생 순서를 alphabet처럼 사용합니다.

선정된 k-mer를 기반으로 두 genome을 다시 스캔하여, 각 k-mer의 발생 위치를 기록합니다. 예를 들어, k-mer "CGA"가 genome1에서 1003번째 위치, genome2에서 103번째 위치에 나타났다면, 아래와 같이 저장합니다.

- genome1 변환 결과(예시):

  genome1 = <AGT (102) – CGA (1003) – TCG (5003) – AGT (9013) – CGA (15003) – TCG (21003)>

- genome2 변환 결과(예시):

  genome2 = <CGA (103) – TCG (1203) – TGT (4903) – TCG (8003) – AGT (13003) – CGA (21093)>

4. Dynamic Programming으로 LCS 계산

Dynamic Programming 알고리즘을 사용하여 Longest Common Subsequence (LCS)를 계산합니다.

- LCS (예시):

  LCS = <CGA-TCG-AGT>

- 각 genome의 LCS 발 생 위치도 함께 출력합니다(예시):
  - genome1에서의 위치:

    <CGA(1003)–TCG(5003)–AGT(9013)>
  - genome2에서의 위치:

    <CGA(103)–TCG(1203)–AGT(13003)>

5. 결과물 저장

- 결과는 LCS 서열과 함께 각 genome에서의 LCS 발생 위치를 포함해야 합니다.
- 결과를 3개의 파일로 분리하여 저장합니다.

1. LCS 정보만 저장하는 파일

   파일명: [your_student_id]_[k]_[geneome1]_[genome2]_LCS.txt

   형식: 단일 텍스트 파일

   파일 내용 예시:

   ```
   CGA-TCG-AGT
   ```

2. genome1의 LCS와 위치 정보 파일

   파일명: [your_student_id]_[genome1]_[k]_LCS_positions.csv

   형식: CSV 파일

   파일 내용 예시:

   ```
   LCS k-mer,Position
   CGA,1003
   TCG,5003
   AGT,9013
   ```

3. genome2의 LCS와 위치 정보 파일

   파일명: [your_student_id]_[genome2]_[k]_LCS_positions.csv

   형식: CSV 파일

   파일 내용 예시:

   ```
   LCS k-mer,Position
   CGA,103
   TCG,1203
   AGT,13003
   ```

Example Run Command

제출한 코드는 다음 명령어로 실행되어야 합니다:

$ sh run.sh [k] [genome1.fasta] [genome2.fasta]

예시:

$ sh run.sh 17 NZ_CP009072.fasta NZ_CP064387.fasta

Expected Output Format

결과 파일은 "5. 결과물 저장"에서 언급된 바와 같이 3개의 파일로 분리하여 본인의 학번이 반드시 파일명에 포함되도록 저장되어야 합니다.

파일명:

1. [your_student_id]_[k]_[geneome1]_[genome2]_LCS.txt
   (예시: 2024-00001_17_NZ_CP009072_NZ_CP064387_LCS.txt)
2. [your_student_id]_[k]_[genome1]_LCS_positions.csv
   (예시: 2024-00001_17_NZ_CP009072_LCS_positions.csv)
3. [your_student_id]_[k]_[genome2]_LCS_positions.csv
   (예시: 2024-00001_17_NZ_CP064387_LCS_positions.csv)

Submission Guidelines

- 제출물 요구사항:
  - Python 또는 Java로 작성된 코드
  - 실행 스크립트(run.sh)
  - 제출물은 zip파일로 압축(예:2024-00001.zip)

Grading criteria

1. Genome 내 LCS의 유무와 위치
2. LCS의 길이
3. Run time