FASTQ deduplication

The FASTQ format is a text-based format for storing both DNA-sequences and their corresponding quality scores. It was originally developed at the Wellcome Trust Sanger Institute, but has recently become the *de facto* standard for storing the output of high throughput sequencing instruments.

A FASTQ file contains a series of sequence records, with each record corresponding to four consecutive lines:

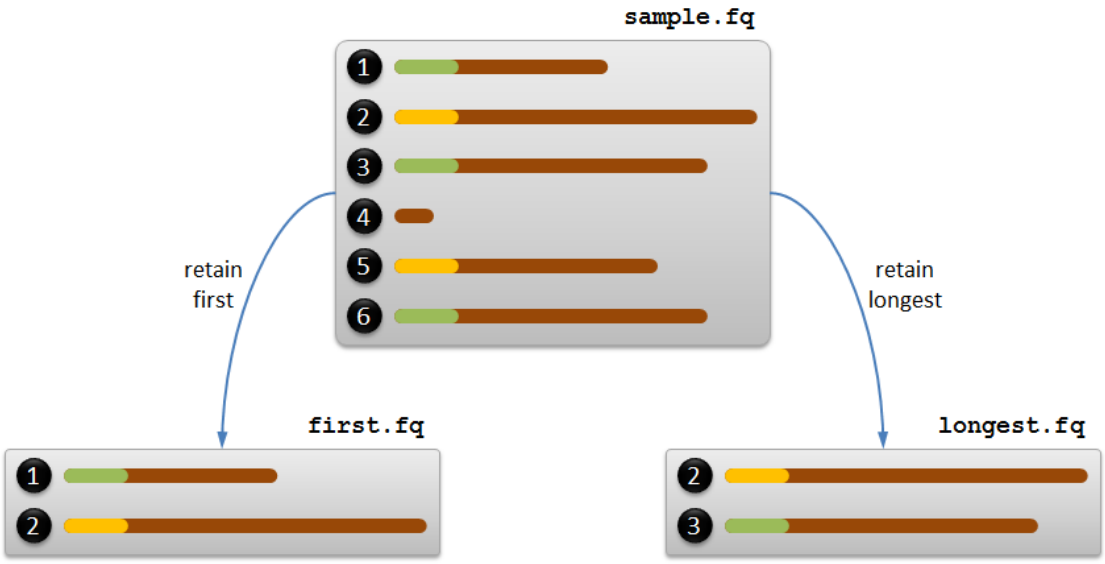
- line 1 begins with the at-sign (@) and is followed by a sequence identifier and an optional description
- line 2 contains a series of sequence letters
- line 3 begins with the plus-sign (+) and is optionally followed by the same sequence identifier and any description
- line 4 encodes the quality values for the sequence in line 2, and must contain the same number of symbols as there are letters in the sequence

Both the sequence letters and quality scores are encoded with a single ASCII character for brevity. As an example, the content of the FASTQ file sample.fq containing six sequence records looks like this:

```
@sample sequence 1
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCT
+sample sequence 1
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^a
@sample sequence 2
CTCCTTATAGCATCATACGCACAGCTGGTATCCCCATTGCTTATTAGGGTTCAGCCCTGTCGCTCTCCCC
+sample sequence 2
ababa``aaaababaaaabaa_``aa``]^aa_`aa_Z_aaaaa^baaaaaaaaaaaaaaaaaa`^```a
@sample sequence 3
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCTGGGGTTAGATTTTATGTCGT
+sample sequence 3
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^ababbaaabaaaaa`]`ba`]
@sample sequence 4
TATAC
+sample sequence 4
baab`
@sample sequence 5
CTCCTTATAGCATCATACGCACAGCTGGTATCCCCATTGCTTATTAGGGT
+sample sequence 5
ababa``aaaababaaaabaa_``aa``]^aa_`aa_Z_aaaaa^baaaa
@sample sequence 6
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCTGGGGTTAGATTTTATGTCGT
+sample sequence 6
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^ababbaaabaaaaa`]`ba`]
```

Prior to sequencing itself, a genomic library has to be constructed. As an artifact of the PCR amplification during library preparation, the same fragment may be read multiple times. That's why the DNA sequences in a FASTQ file are usually deduplicated, in order to avoid a possible bias when analyzing the sequences. Deduplication does not take into account the full length sequences, but only relies on a prefix formed by the first n bases of the sequence.

For example, if we consider prefixes of length 10, records 1, 3 and 6 in the sample FASTQ file share the same prefix GGAAGTCCAT. Similarly, records 2 and 4 also share the same prefix CTCCTTATAG. We have graphically depicted the records of this FASTQ file below, where the same prefixes have been given the same color.



In practice, two strategies are used for deduplicating the sequences in a FASTQ file. Both strategies have in common that all records whose sequence is shorter than n bases (the length of the prefix) are always removed. If $n = 4$, this is for example the case for record 4 in the sample FASTQ file.

In the first strategy — resulting in the left outcome show in the above figure — a FASTQ record is only retained if the prefix of its sequence is seen for the first time. To implement this strategy, the FASTQ records are read one by one and a **set** of prefixes is constructed to remember which prefixes have already been seen. A FASTQ record is then only retained in case its prefix was not yet in the set.

In the second strategy — resulting in the right outcome show in the above figure — for each prefix on the record with the longest sequence is retained. To implement this strategy, the FASTQ file now have to be processed twice. The first time, the records are read one by one to construct a **dictionary** that maps each prefix of sequences longer than n onto the length of the longest sequence with that prefix. The second time the records are again read one by one, and only those records are retained whose sequence length matches the length onto which the prefix of the sequence is mapped by the dictionary. In case the same prefix is found in multiple records whose sequence length matches the length onto which the prefix of the sequence is mapped by the dictionary, only the first record of those records needs to be retained. For example, the sequences of records 3 and 6 in the sample FASTQ file share the same prefix and the same sequence length (the maximal length of all records sharing that prefix), and thus only record 3 needs to be retained. The sequence of record 1 also shares the same prefix, but is sequence is shorter than those of records 3 and 6.

Assignment

In this assignment we only work with FASTQ files that contain DNA sequences. These DNA sequences are represented as strings that only contain the uppercase letters A, C, G en T. Your task:

- Write a function retain_first that takes three arguments: *i*) a string containing the location of a FASTQ file whose records need to be deduplicated, *ii*) a string containing the location of a new FASTQ file to which the retained records must be written in the same order as they appear in the given FASTQ file, and *iii*) the length $n \in \mathbb{N}_0$ of the prefixes that must be used for deduplication. The function must implement the first deduplication strategy outlined in the introduction of this assignment.
- Write a function find_longest that takes two arguments: *i*) a string containing the location of a FASTQ file and *ii*) an integer $n \in \mathbb{N}_0$. The function must return a dictionary that maps each prefix of length n from the FASTQ file (limited to prefixes of sequences having at least length n) onto the maximal length of the sequences having this prefix.
- Use the function find_longest to write a function retain_longest. The function takes the same three arguments as the function retain_first, with the same meaning. However, this time the function must implement the second deduplication strategy outlined in the introduction of this assignment.

In practice, FASTQ files contain the results o f *next-generation sequencing* experiments that produce millions of *reads* (short DNA sequences). Such files are simply too big to fit in computer memory. Therefore, avoid during the implementation of the deduplication functions to load the entire file content in computer memory, but only claim the memory needed to store the information of a single FASTQ record, together with a set (function retain_first) or a dictionary (function retain_longest) of prefixes that is used to decide whether or not the FASTQ records must be retained.

Example

In the following interactive session, we assume that the FASTQ file sample.fq is located in the current directory. This file has the same content as the sample file shown in the introduction of this assignment.

```
>>> retain_first('sample.fq', 'first.fq', 10)
>>> print(open('first.fq', 'r').read().rstrip())
@sample sequence 1
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCT
+sample sequence 1
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^a
@sample sequence 2
CTCCTTATAGCATCATACGCACAGCTGGTATCCCCATTGCTTATTAGGGTTCAGCCCTGTCGCTCTCCCC
+sample sequence 2
ababa``aaaababaaaabaa_``aa``]^aa_`aa_Z_aaaaa^baaaaaaaaaaaaaaaaaa`^```a

>>> retain_longest('sample.fq', 'longest.fq', 10)
>>> print(open('longest.fq', 'r').read().rstrip())
@sample sequence 2
CTCCTTATAGCATCATACGCACAGCTGGTATCCCCATTGCTTATTAGGGTTCAGCCCTGTCGCTCTCCCC
+sample sequence 2
ababa``aaaababaaaabaa_``aa``]^aa_`aa_Z_aaaaa^baaaaaaaaaaaaaaaaaa`^```a
@sample sequence 3
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCTGGGGTTAGATTTTATGTCGT
+sample sequence 3
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^ababbaaabaaaaa`]`ba`]
```

Het FASTQ formaat wordt gebruikt om zowel DNA-sequenties als de daarmee geassocieerde kwaliteitsscores op te slaan in een tekstbestand. Het formaat werd oorspronkelijk ontwikkeld

door het Wellcome Trust Sanger Institute, maar is ondertussen zowat de *de facto* standaard geworden om *next-generation sequencing* resultaten op te slaan.

Een FASTQ bestand bevat een reeks sequentierecords, waarbij elke record bestaat uit vier opeenvolgende regels:

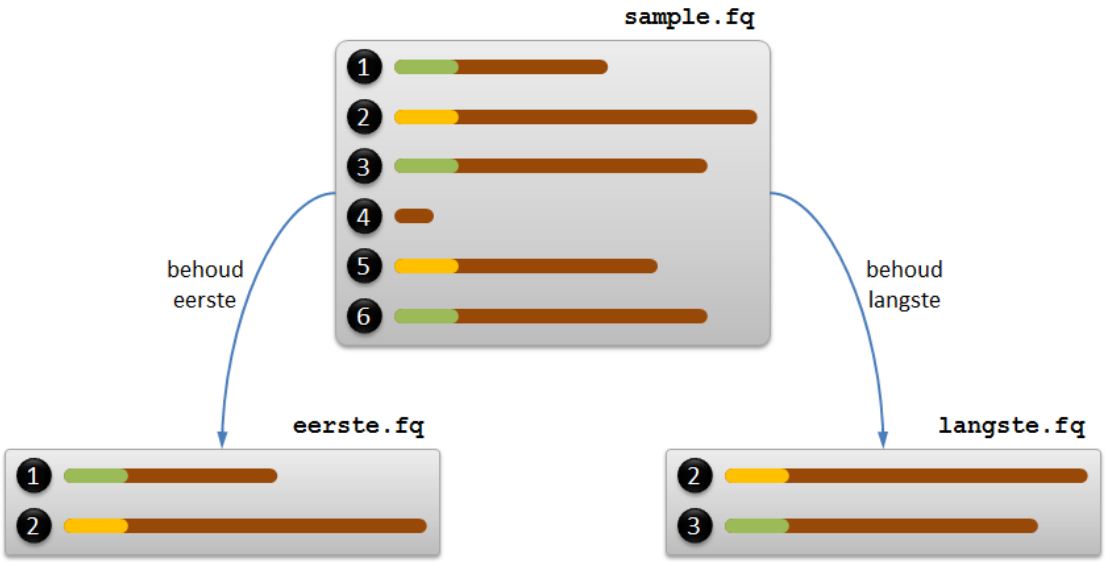
- regel 1 bevat een apestaartje (@) gevolgd door een *sequence identifier* en een optionele beschrijving
- regel 2 bevat een reeks sequentieletters
- regel 3 bevat een plusteken (+), optioneel gevolgd door dezelfde *sequence identifier* en eventueel een bijkomende beschrijving
- regel 4 bevat de corresponderende kwaliteitsscore voor elke sequentieletters uit regel 2

Omwille van de eenvoud worden in dit formaat zowel de sequentieletters als de kwaliteitsscores voorgesteld door één enkel ASCII karakter. Bij wijze van voorbeeld ziet de inhoud van het FASTQ bestand sample.fq met zes sequentierecords er als volgt uit:

```
@sample sequence 1
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCT
+sample sequence 1
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^a
@sample sequence 2
CTCCTTATAGCATCATACGCACAGCTGGTATCCCCATTGCTTATTAGGGTTCAGCCCTGTCGCTCTCCCC
+sample sequence 2
ababa``aaaababaaaabaa_``aa``]^aa_`aa_Z_aaaaa^baaaaaaaaaaaaaaaaaa`^```a
@sample sequence 3
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCTGGGGTTAGATTTTATGTCGT
+sample sequence 3
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^ababbaaabaaaaa`]`ba`]
@sample sequence 4
TATAC
+sample sequence 4
baab`
@sample sequence 5
CTCCTTATAGCATCATACGCACAGCTGGTATCCCCATTGCTTATTAGGGT
+sample sequence 5
ababa``aaaababaaaabaa_``aa``]^aa_`aa_Z_aaaaa^baaaa
@sample sequence 6
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCTGGGGTTAGATTTTATGTCGT
+sample sequence 6
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^ababbaaabaaaaa`]`ba`]
```

Voorafgaand aan het sequeneren zelf, wordt eerst een genomische bibliotheek opgebouwd. Als een artefact van de PCR amplificatie die daarmee gepaard gaat, kan het gebeuren dat eenzelfde fragment meerdere keren wordt uitgelezen. Daarom worden de DNA-sequenties in een FASTQ bestand doorgaans eerst ontdubbel, om een mogelijke bias te vermijden bij het analyseren van de sequenties. Bij het ontdubbelen worden niet de volledige sequenties in rekening gebracht, maar wordt enkel gekeken naar een prefix die bestaat uit de eerste n basen van de sequentie.

Als we bijvoorbeeld prefixen van lengte 10 beschouwen, dan zien we in het voorbeeld FASTQ bestand dat de records 1, 3, en 6 dezelfde prefix GGAAGTCCAT hebben, en dat ook de records 2 en 4 dezelfde prefix CTCCTTATAG hebben. We hebben de records uit dit FASTQ bestand hieronder grafisch voorgesteld, waarbij we dezelfde prefixen ook dezelfde kleur gegeven hebben.



Voor het ontdubbelen van de sequenties in een FASTQ bestand worden in de praktijk twee strategieën gebruikt. Beide strategieën hebben gemeenschappelijk dat alle records waarvan de sequentie korter is dan n (de lengte van de prefix) altijd verwijderd worden. Als $n = 10$, dan is dit bijvoorbeeld het geval voor record 4 in het voorbeeld FASTQ bestand.

Bij de eerste strategie — die het linkse resultaat uit bovenstaande figuur oplevert — wordt een FASTQ record enkel behouden als we de prefix van de sequentie voor het eerst te zien krijgen. Voor de implementatie van deze strategie volstaat het om de FASTQ records één voor één te overlopen en een **verzameling** bij te houden van de prefixen die reeds gezien werden. Een FASTQ record wordt dan enkel behouden als de prefix nog niet in de verzameling voorkomt.

Bij de tweede strategie — die het rechtse resultaat uit bovenstaande figuur oplevert — wordt voor elke prefix enkel de record met de langste sequentie behouden. Voor de implementatie van deze strategie moeten de records van het originele FASTQ bestand nu twee keer overlopen worden. De eerste keer worden de records overlopen om een **dictionary** op te bouwen die elke prefix van sequenties langer dan n afbeeldt op de lengte van de langste sequentie met die prefix. De tweede keer worden de records opnieuw één voor één overlopen, en worden enkel de records behouden waarvan de lengte van de sequentie correspondeert met de lengte waarop de prefix van de sequentie wordt afgebeeld door de dictionary. Indien er voor eenzelfde prefix meerdere records zijn waarvoor de lengte van de sequentie correspondeert met de lengte waarop de prefix van de sequentie wordt afgebeeld door de dictionary, dan moet enkel de eerste record daarvan behouden worden. In het voorbeeld FASTQ bestand hebben de sequenties van records 3 en 6 bijvoorbeeld dezelfde prefix en dezelfde lengte (de maximale lengte van alle records met die prefix), en moet dus enkel record 3 behouden worden. De sequentie van record 1 heeft ook dezelfde prefix, maar die sequentie is korter dan die van records 3 en 6.

Opgave

In deze opgave werken we enkel met FASTQ bestanden die DNA-sequenties bevatten. DNA-sequenties worden hierbij voorgesteld als strings die enkel bestaan uit de hoofdletters `A`, `C`, `G` en `T`. Gevraagd wordt:

- Schrijf een functie `behoud_eerste` waaraan drie argumenten moeten doorgegeven worden: *i*) een string met de locatie van een FASTQ bestand waarvan de records moeten ontdubbeld worden, *ii*) een string met de locatie van een nieuw aan te maken FASTQ bestand waar de

behouden records naartoe moeten weggeschreven worden in dezelfde volgorde waarin ze voorkomen in het gegeven FASTQ bestand, en *iii*) de lengte $n \in \mathbb{N}_0$ van de prefixen die moeten gebruikt worden bij het ontdubbelen. Voor het ontdubbelen moet de `eerste` strategie geïmplementeerd worden die in de inleiding van de opgave staat beschreven.

- Schrijf een functie `vind_langste` waaraan twee argumenten moeten doorgegeven worden: *i*) een string met de locatie van een FASTQ bestand en *ii*) een getal $n \in \mathbb{N}_0$. De functie moet een dictionary teruggeven, die elke prefix van lengte n die voorkomt in het gegeven FASTQ bestand (beperkt tot prefixen van sequenties die minstens lengte n hebben) afbeeldt op de maximale lengte van de sequenties met die prefix.
- Gebruik de functie `vind_langste` om een functie `behoud_langste` te schrijven. Aan deze functie moeten dezelfde drie argumenten doorgegeven worden als bij de functie `behoud_eerste`, met dezelfde betekenis. Voor het ontdubbelen moet nu evenwel de tweede strategie geïmplementeerd worden die in de inleiding van de opgave staat beschreven.

In de praktijk bevatten FASTQ bestanden de resultaten van *next-generation sequencing* experimenten, die bestaan uit enkele miljoenen *reads* (korte DNA-sequenties). Dergelijke bestanden zijn te groot om volledig in het computergeheugen te passen. Zorg er bij de implementatie van de ontdubbelingsfuncties daarom voor dat niet de volledige inhoud van het gegeven FASTQ bestand in het geheugen wordt geladen, maar dat enkel geheugen gebruikt wordt voor de informatie van één FASTQ record, samen met een verzameling (functie `behoud_eerste`) of een dictionary (functie `behoud_langste`) met prefixen die gebruikt wordt om te beslissen of die FASTQ record al dan niet moet uitgeschreven worden.

Voorbeeld

Bij onderstaande voorbeeldsessie gaan we ervan uit dat het FASTQ bestand sample.fq zich in de huidige directory bevindt. Dit bestand heeft dezelfde inhoud als het voorbeeldbestand uit de inleiding van deze opgave.

```
>>> behoud_eerste('sample.fq', 'eerste.fq', 10)
>>> print(open('eerste.fq', 'r').read().rstrip())
@sample sequence 1
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCT
+sample sequence 1
ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^a
@sample sequence 2
CTCCTTATAGCATCATACGCACAGCTGGTATCCCCATTGCTTATTAGGGTTCAGCCCTGTCGCTCTCCCC
+sample sequence 2
ababa``aaaababaaaabaa_``aa``]^aa_`aa_Z_aaaaa^baaaaaaaaaaaaaaaaaa`^```a

>>> vind_langste('sample.fq', 10)
{'GGAAGTCCAT': 60, 'CTCCTTATAG': 70}

>>> behoud_langste('sample.fq', 'langste.fq', 10)
>>> print(open('langste.fq', 'r').read().rstrip())
@sample sequence 2
CTCCTTATAGCATCATACGCACAGCTGGTATCCCCATTGCTTATTAGGGTTCAGCCCTGTCGCTCTCCCC
+sample sequence 2
ababa``aaaababaaaabaa_``aa``]^aa_`aa_Z_aaaaa^baaaaaaaaaaaaaaaaaa`^```a
@sample sequence 3
GGAAGTCCATGGAGTTATTTGCGGTAACCGGGACCGGCCTGGGGTTAGATTTTATGTCGT
+sample sequence 3
```

ba^aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^ababbaaabaaaaa`]`ba`]