Bloom Filter Tool

Tool creato utilizzando linguaggio Python e composto da 3 files:

- Bloomfilter.py
  File contenente la classe responsabile della creazione del BloomFilter e controllo della presenza della read (r) data come input
- Utils.py
  File di Utils con metodi per la lettura del file FASTA e splicing delle reads in kmers
- Main.py
  File contenente l'istanza del BloomFilter e il suo utilizzo, scrive il risultato del test su output.txt

Utils.py

- **read_input(path)**
  metodo che prende in input il file FASTA e ne riporta i contenuti in una lista di oggetti di tipo Sequence

- **splice(read, k)**
  metodo che prende in input una lista di reads e il valore numerico K e ritorna la lista di stringhe formata dai relativi k-mer

bloomfilter.py

class BloomFilter:

- **__init__(input, k)**
  Costruttore della classe, prende come input una lista di reads (R) e il parametro k per costruire i k-mers, dopo aver rimosso eventuali kmer ridondanti calcola i parametri che verranno utilizzati nella creazione del bloom filter:
  - **p** = probabilitá di falso negativo (statica a 0,01)
  - **n** = numero di elementi da aggiungere nel filtro (numero di kmers)
  - **size** = grandezza del bitarray calcolata in base a **p** e **n**
  - **num_hashes** = numero di funzioni di hash da usare per aggiungere i kmer nel filtro calcolata in base a **size** e **nf** con il metodo calculate_hashes (nel caso il numero calcolato per num_hashes sia minore di 0 viene settato a 1 come default)

- **addItem(item)**
  Metodo responsabile dell'aggiunta dei vari kmers nel filtro, vengono calcolate tutte le hash richieste per ogni kmer, i bit presenti poi in quella posizione vengono portati a 1 nel filtro
- **checkRead(item)**
  Metodo che controlla se la read (r) di input è presente o meno nel filtro, controllando che la read sia piú grande della lunghezza k dei k-mer, la read viene divisa in una lista di k-mer che verranno poi controllati uno ad uno dal metodo **checkItem**
- **checkItem(item)**
  Metodo che controlla se il kmer di input sia presente o meno nel BloomFilter, calcola semplicemente tutto il range di hash del kmer (in base a **num_hashes**) e controlla poi che in quelle posizioni il bit nel BitArray sia uguale a 1, in caso uno di questi non lo sia ritorna False e la ricerca termina con esito negativo

Main.py

- In questo file troviamo la sequenza di test del BloomFilter
  Viene letto il file FASTA e creata un istanza di BloomFilter usando la lista di Read (R) e il parametro K dei k-mers (in questo caso settato a k=20)
  Per testare il caso positivo viene presa una read (r) casuale dalla lista di input mentre per il caso negativo alla stessa read vengono aggiunte altre 3 basi 'AGT' come prefisso (questo test puó quindi, molto raramente, riportare un riscontro positivo), i risultati del test sono poi scritti nel file di testo output.txt

Esempio:

K = 20

Read (r) randomly chosen from sample_1.fa:

TAGCGTTTTTCCCTTTGCGGTTCGTCTTTCCACCAGAATATTAAAAATGCTAATCTTCGCGTTGTATTCCTGTGTTCACCTGTTCTCGGAGGGGCTTTCA

Result:

True

Read (r) for non-presence test in filter:

AGTAGCGTTTTTCCCTTTGCGGTTCGTCTTTCCACCAGAATATTAAAAATGCTAATCTTCGCGTTGTATTCCTGTGTTCACCTGTTCTCGGAGGGGCTTTCA

Result:

False