Bioinformatica

Elia Fabbris, 793240
Matteo Fumagalli, 793670

A.A 2018/2019

Contents

1 Introduzione

Lo scopo del seguente progetto è stato quello di creare un tool che, prendendo in input un file formato FASTQ, produca in output un grafo di assemblaggio dal quale si può ottenere il relativo genoma.
Il grafo di assemblaggio che è stato scelto e utilizzato è l'*overlap graph* nel quale ogni nodo corrisponde ad un read e ogni arco identifica un overlap tra due nodi connessi.
Di seguito verranno prima illustrati i formati di input ed output utilizzati, ovvero contenenti i read e il grafo generato, dopodiché sarà presente una descrizione degli algoritmi impiegati e dei parametri scelti, quali soglia di qualità minima della basi del read, lunghezza minima di un read, lunghezza minima di overlap, etc.

2 Descrizione del progetto

2.1 Formato di input

Il formato di input accettato dal tool è FASTQ (estensione '.fq'), ovvero un formato testuale nel quale ad ogni read corrisponde una stringa rappresentante la qualità di ciascuna sua base. In particolare è formato da quattro righe:

1. Header con simbolo iniziale '@';
2. Sequenza primaria del read;
3. Header della sequenza di qualità con simbolo iniziale '+';
4. Sequenza rappresentante la qualità o sequenza dei Phred scores codificati in caratteri.

La conversione da Phred q in carattere avviene attraverso la seguente formula:

$$c = ASCII(min(q, 93) + 33) \tag{1}$$

Prima di passare alle fasi presentate successivamente è stato assunto che gli errori si concentrassero nei bordi delle reads, infatti alcuni caratteri della sequenza di qualità sono stati modificati diminuendo la qualità di questi ultimi.

2.2 Trimming

Per effettuare questa prima operazione è stato ritenuto opportuno salvare sia la stringhe rappresentanti le sequenze dei read sia le stringhe rappresentanti le qualità in due liste, in questo modo è stato possibile agire in maniera più adeguata sui singoli caratteri.
Grazie al formato FASTQ è stato possibile fissare una soglia minima q così che, attraverso l'operazione di trimming, è stata trovata la più lunga sottostringa della read originale in cui ciascuna base presenti una qualità maggiore o uguale a quella fissata.
La soglia stabilita è pari a 85 che corrisponde al carattere ASCII 'U' ed è pari al valore di qualità 52.
Dopo aver analizzato ciascun read contenuto nel file iniziale, la funzione restituisce due elementi: i reads originali e le più lunghe sottostringhe di ogni read con qualità sufficiente, entrambe riconvertite da liste in stringhe.
Per evitare di utilizzare read troppo corti è stata imposta una lunghezza minima di trimming, ossia se la sottostringa ottenuta risulti con una lunghezza inferiore a 60 caratteri verrebbe mantenuto il read originale altrimenti verrebbe sovrascritto con la sua sottostringa.
Il risultato è una lista contenente i read validi che possono venire utilizzati per costruire l'*overlap graph* (paragrafo **2.3**).

Un esempio preso dal file FASTQ che è stato utilizzato viene riportato di seguito.

Read originale.

@read2
GAGCTTCTGAAACTAGGCGGCAGAGGCGGAGCCGCTGTGGCACTGCTGCGCCTCTGCTGCGCCTCGGGTGTCTTTTGCGG
+
aaaa ‘ Ta ‘ aa ‘] __ ‘ aa ‘ _ [[_a ‘ ^ \ \WUWZ X^ UXaaaa ‘ aa ‘ aa ‘] __ ‘ aa ‘ _U [_a ‘ ^ \ \UVWZTX^ TXaaaa ‘ U [_

Trimming del read, *q = 52.* Lunghezza ottenuta pari a 61.

@read2
CTGAAACTAGGCGGCAGAGGCGGAGCCGCTGTGGCACTGCTGCGCCTCTGCTGCGCCTCGG
+
a ‘ aa ‘] __ ‘ aa ‘ _ [[_a ‘ ^ \ \WUWZ X^ UXaaaa ‘ aa ‘ aa ‘] __ ‘ aa ‘ _U [_a ‘ ^ \ \UVWZ

2.3 Calcolo del grafo di assemblaggio

Il passo seguente è stato quello di creare un grafo di assemblaggio. Come dichiarato precedentemente l'*overlap graph* è composto nel seguente modo:

- Ogni nodo corrisponde ad un read;
- Ogni arco è presente solamente se tra due read esiste un overlap e, in caso affermativo, è etichettato con la lunghezza di quest'ultimo.

In una prima fase vengono quindi definiti i nodi che comporranno il grafo tenendo conto del fatto che nel caso in cui vi fossero due read uguali ne venga creato solamente uno. Nella fase successiva è possibile gestire questo avvenimento grazie ad un dizionario in cui le chiavi corrispondono ai read e i valori corrispondono all'ID del nodo. Quindi nel caso in cui, nella fase di definizione dei nodi, venga letto un read di cui è già stato creato un nodo in precedenza esso viene ignorato e preso in considerazione il read successivo.
A questo punto è stato implementato un ciclo che sia in grado di analizzare l'overlap di ciascuna coppia possibile di read, eccetto nel caso in cui essi siano uguali per evitare che si creino dei cappi.
Per la rilevazione dell'overlap è stato scelto quindi di implementare il 'Problema 8' di allineamento semi-globale tra due sequenze, ovvero la ricerca del miglior prefisso di una stringa A di lunghezza *1,2,..,i,i+1,..,m* che si allinei con il suffisso di una stringa B di lunghezza *1,2,..,j,j+1,..,n.* E' stato necessario creare una matrice, nella quale il valore delle singole celle rispettasse le seguenti equazioni di ricorrenza e i casi base:

$$D(0,0) = 0, D(i,0) = i * d, D(0,j) = 0$$

$$D(i,j) = max\begin{cases} D(i-1,j-1) + \delta(a_i, b_j) \\ D(i-1,j) + d \\ D(i,j-1) + d \end{cases}$$

- D(i,j) si riferisce al valore della cella in posizione *i,j*;
- δ(i,j) (*scoring function*)si riferisce al confronto tra il carattere in posizione i della stringa A con il carattere in posizione j della stringa B. I punteggi che associa sono i seguenti:
  1. MATCH. Nel caso in cui i due caratteri siano uguali;
  2. MISMATCH. Nel caso in cui i due caratteri siano differenti;
- Il valore d si riferisce al punteggio di INDEL. Ovvero nel caso in cui ci sia un inserimento o una cancellazione di un carattere in una delle due stringhe.

I punteggi assegnati ai casi di MATCH, MISMATCH ed INDEL sono rispettivamente 5, -2 e -4.
E' stato assunto che nel caso in cui ci siano valori uguali provenienti da due o più equazioni di ricorrenza venga scelto il caso di sostituzione, ossia la cella in posizione *(i-1, j-1)*.
Per la ricostruzione dell'overlap è stata utilizzata una matrice di supporto, la quale mantiene per ogni cella la direzione da cui si proviene.
Una volta ottenuto il miglior overlap tra le due read è stato assunto che la lunghezza minima di quest'ultimo sia di almeno 25 caratteri e che siano presenti al massimo 10 differenze tra il suffisso di una read e il prefisso di un'altro read.
Se queste assunzioni vengono rispettate, tra i due nodi viene creato un link nel formato *GFA* (paragrafo **2.4**).

2.4 Formato di output

Il formato con il quale è stato assemblato il grafo è chiamato *Graphical Fragment Assembly* (GFA) e la versione utilizzata è la 1.0.
Le sequenze, come detto precedentemente, sono rappresentate dai nodi e chiamate *segmenti* (simbolo 'S'). La loro connessione, rappresentante un match tra il suffisso di un nodo e il prefisso di un altro, è definita dai *link* (simbolo 'L'). Ogni riga GFA che andrà a comporre il grafo è composta da vari campi separati da tab ed è definita come segue.

Segmento

Campo	Tipo	Valore	Descrizione
RecordType	Carattere	S	Tipo di record
ID	Stringa	[a-zA-Z0-9]*	Identificativo univoco del segmento
Sequenza	Stringa	[ACGT]*	Sequenza primaria del read

Link

Campo	Tipo	Valore	Descrizione
RecordType	Carattere	L	Tipo di record
ID_node1	Stringa	[a-zA-Z0-9]*	Identificativo del nodo di partenza (suffisso)
Nodo1_orientation	Stringa	[+-]	Orientamento del *node1*
ID_node2	Stringa	[a-zA-Z0-9]*	Identificativo del nodo di arrivo (prefisso)
Node2_orientation	Stringa	[+-]	Orientamento del *node2*
Overlap	Number	[0-9]*	Lunghezza di overlap tra il suffisso del *node1* ed il prefisso del *node2*.

Inoltre viene aggiunta un'ulteriore riga per definire la versione di GFA utilizzata, nel nostro caso la 1.0, e definita come segue:

Header

Campo	Tipo	Valore	Descrizione
RecordType	Carattere	H	Tipo di record
Version_number	Z	[12]	Versione di GFA utilizzata

Al termine dell'assemblaggio il grafo viene salvato localmente in un file chiamato 'graph' con estensione '.gfa'.

2.5 Ricostruzione del genoma

Per la ricostruzione del genoma viene importato il file *graph.gfa* creato negli step precedenti.
La funzione necessaria alla ricomposizione del genoma è stata fornita dalla libreria *gfapy*, in particolare è stata utilizzata la funzione *merge_linear_path(segmentPath)*.
Tale funzione unisce i segmenti dichiarati all'interno di una variabile *segmentPath* ed autonomamente assembla le stringhe tenendo in considerazione ciascuna lunghezza di overlap dichiarata precedentemente nei record di tipo *link*.
Il genoma risultante dal file utilizzato, contenente i read, è il seguente:

GGGCTTGTGGCGCGAGCTTCTGAAACTAGGCGGCAGAGGCGGAGCCGCTGTGGCACTGCT
AGGGGACAGATTTGTGACCGGCGCGGTTTTTGTCAGCTTACTCCGGCCAAAAAAGAACTG
CACCTCTGGAGCGGGTTAGTGGTGGTGGTAGTGGGTTGGGACGAGCGCGTCTTCCGCAGT
CCCAGTCCAGCGTGGCGGGGGAGCGCCTCACGCCCCGGGTCGCTGCCGCGGCTTCTTGCC
CTTTTGTCTCTGCCAACCCCCACCCATGCCTGAGAGAAAGGTCCTTGCCCGAAGGCAGAT

Per ottenere questa ricostruzione è stato assunto che l'ordine delle reads presenti sul file rappresenti l'effettivo ordine nel genoma completo.
Qui di seguito viene mostrato il grafo ottenuto. Il software utilizzato è *Bandage*, il quale ha permesso di verificare il nostro risultato e avere una visione grafica di tutti i reads. Questi ultimi sono rappresentati con diversi colori e con il proprio identificativo.

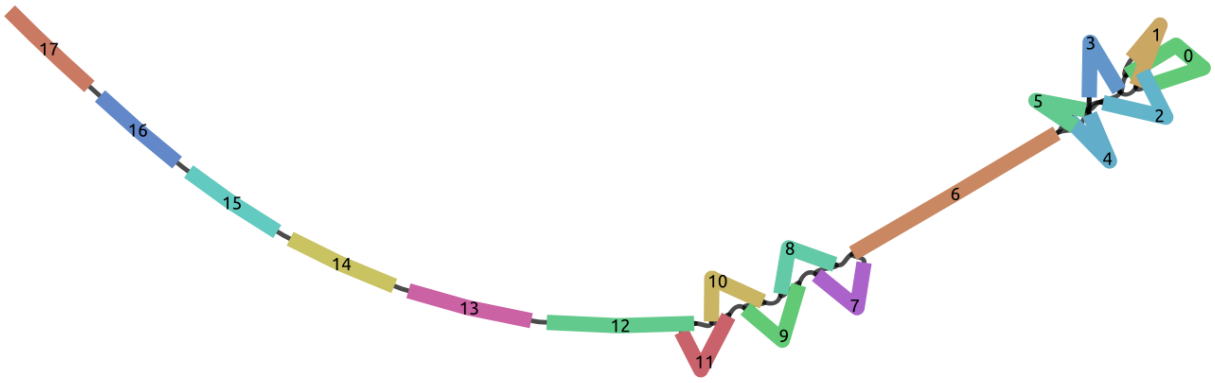

Figure 1: Overlap graph

3 Istruzioni di compilazione

Il risultato prodotto comprende quattro file Python: *trimming.py, overlapMatrix.py, overlapGraph.py, print_sequence.py.*

- *trimming.py*: recupera la miglior sottosequenza di ogni read, rispettando la qualità fissata
- *overlapMatrix.py*: implementa il problema P8, ottenendo l'overlap date due reads
- *overlapGraph.py*: crea il grafo, rispettando il formato della libreria gfapy
- *print_sequence.py*: restituisce l'intero genoma

Le librerie utilizzate per lo sviluppo del codice sono state: *gfapy, numpy* e *sys.*
La versione di Python utilizzata è la 3.6.7. In caso non siano installate le librerie sarà necessario utilizzare la funzione:

```
pip install nome_della_libreria
```

Per compilare il codice è sufficiente aprire il terminale nella cartella in cui sono contenuti gli script Python e inserire:

```
python3 overlapGraph.py
```

In questo modo verrà restituito l'intero genoma.

NOTE. Eseguendo il 'Run' si otterrà il risultato dato attraverso il file di read 'reads_mod.fq' che, come riportato precedentemente, è stato modificato per poter sfruttare alcune funzionalità che utilizzando il file originale non sarebbero utilizzate (gli esempi e la figura vista precedentemente fanno riferimento ad esso).
Per eseguire il tool sul file di read originale bisogna modificare il file 'trimming.py' selezionando il percorso al file 'reads.fq' (riga 1-2 del file).