

Elementi di Bioinformatica

Gianluca Della Vedova

Univ. Milano-Bicocca  
http://gianluca.dellavedova.org

30 novembre 2018

Grafi di assemblaggio

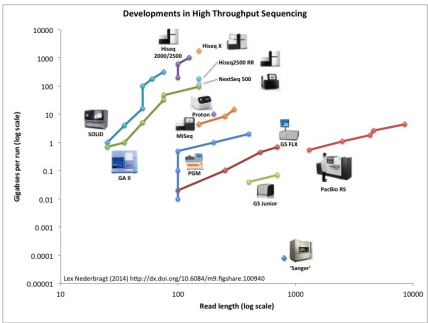
Tecnologie

- Porzioni di genoma chiamate **read**
- 50–10000bp (base pairs)
- spesso in coppie (**mate pairs**)
- posizione originaria ignota

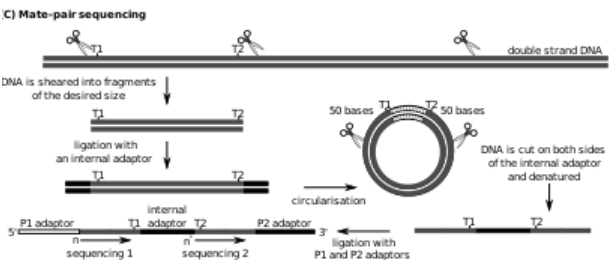
Obiettivo

Ricostruire il genoma: circa 3 miliardi bp

Evoluzione tecnologica



Mate pairs



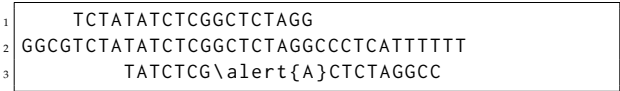
Regola 1

Suffisso di una read può essere prefisso di un'altra read: overlap

Overlap — sovrapposizione



Probabile motivo



Errore oppure organismi diploidi

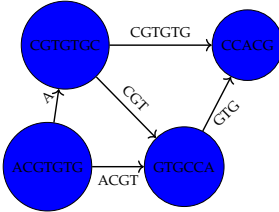
Grafo di overlap

Read

ACGTGTG CGTGTGC GTGCCA CCACG

Arco fra tutte le coppie di read con overlap abbastanza lungo

Grafo



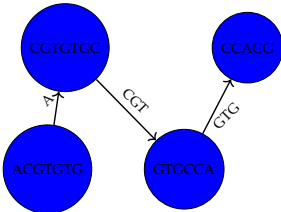
String Graph

Read

ACGTGTG CGTGTGC GTGCCA CCACG

Si rimuovono gli archi transitivi dal grafo di overlap

Grafo



Shortest superstring

Istanza

Insieme  $S = \{s_1, \dots, s_n\}$  di stringhe

Soluzioni ammissibili

Superstring  $T$  di  $S$ . Ogni  $s_i$  è sottostringa di  $T$

Funzione obiettivo

$|T|$

$T$  è il genoma assemblato,  $S$  le read

Problema

Regioni ripetute

# Algoritmo ingordo

## Algoritmo

- 1 Fondere le due stringhe con massimo overlap
- 2 Finchè non rimane una stringa sola

# Esempio: a\_long\_long\_long\_time

- 1 ng\_lon\_long\_a\_long long\_l ong\_ti ong\_lo long\_t g\_long g\_time ng\_tim
- 2 ng\_time ng\_lon\_long\_a\_long long\_l ong\_ti ong\_lo long\_t g\_long
- 3 ng\_time g\_long\_ng\_lon a\_long long\_l ong\_ti ong\_lo long\_t
- 4 ng\_time long\_ti g\_long\_ng\_lon a\_long long\_l ong\_lo
- 5 ng\_time ong\_lon long\_ti g\_long\_a\_long long\_l
- 6 ong\_lon long\_time g\_long\_a\_long long\_l
- 7 long\_lon long\_time g\_long\_a\_long
- 8 long\_lon g\_long\_time a\_long
- 9 long\_long\_time a\_long
- 10 a\_long\_long\_time

# Problema del commesso viaggiatore (TSP)

## Istanza

Grafo orientato  $G = \langle V, A \rangle$ , con archi pesati  $w : A \mapsto \mathbb{Q}^+$

## Soluzioni ammissibili

Permutazione  $\Pi = \langle \pi_1, \dots, \pi_n \rangle$  of  $V$

## Funzione obiettivo

$$w(\pi_n, \pi_1) + \sum_{i=1}^n w(\pi_i, \pi_{i+1})$$

- Una soluzione è un percorso che tocca ogni città esattamente una volta e torna al punto di partenza
- Il costo è il peso totale di tutti gli archi attraversati
- NP-completo, **ma risolvibile in pratica**

# Superstringa più corta e TSP

## Similarità

1 read = 1 città

## Differenze

- assemblaggio  $\neq$  ciclo
- lunghezza stringa  $\neq$  costo percorso TSP

## Proprietà

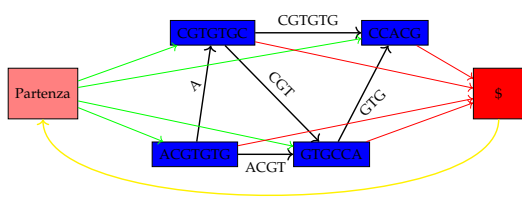
$|S| = \sum_{i=1}^n |s_i| - \sum_{i=1}^{n-1} |ov(s_i, s_{i+1})|$ , dove  $ov(\cdot, \cdot)$  è la lunghezza della sovrapposizione fra le stringhe

# Grafo di overlap — TSP

## Read

ACGTGTG   CGTGTGC   GTGCCA   CCACGG

## Grafo



# Reverse and complement

- Non si conosce lo strand
- Versione canonica (minima fra  $x$  e  $revcomp(x)$ )
- complica il calcolo degli overlap

# Overlay — Layout — Consensus

## Passi

- 1 Overlay: calcolare le sovrapposizioni e costruire il grafo. Usare suffix array (esatto) o programmazione dinamica (errori).
- 2 Layout: Fondere i cammini per ottenere i **contigs**. Le ripetizioni (branching nodes) vengono rimosse.
- 3 Consensus: calcola i nucleotidi

# SBH

## DNA array

- Tecnologia vecchia
- Per ogni  $k$ -mero, si conosce se appare nel genoma
- $k \approx 8$

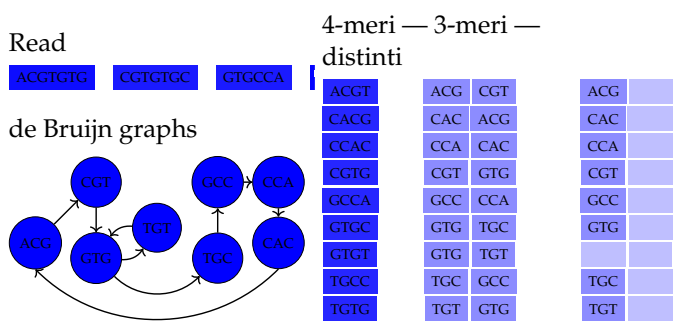
## Procedura

- 1 Ogni  $k$ -mero viene diviso in  $(k - 1)$ -meri
- 2 Un vertice per ogni  $(k - 1)$ -mero
- 3 Un arco per ogni  $k$ -mero

## Adesso

Stessa procedura, a partire dai read

# Grafo di de Bruijn



# Problemi su grafi

- ## Ciclo Euleriano
- 1 Un assemblaggio valido è un cammino che attraversa **ogni arco** esattamente una volta
  - 2 Cammino Euleriano
- ## Ciclo Hamiltoniano
- 1 È un cammino che attraversa **ogni vertice** esattamente una volta
  - 2 Caso particolare di TSP
- ## Confronto
- Qual è più difficile da risolvere?

# Grafi Euleriani

**Definizione**

Sia  $G = \langle V, A \rangle$  un grafo orientato.  $G$  è semi-euleriano se esistono due vertici  $s, t$  tali che  $N_G^-(s) = N_G^+(s) + 1$ ,  $N_G^-(t) = N_G^+(t) - 1$ , mentre per ogni altro vertice  $w$ ,  $N_G^-(w) = N_G^+(w)$ .

**Definizione**

Sia  $G = \langle V, A \rangle$  un grafo orientato.  $G$  è euleriano se  $N_G^-(w) = N_G^+(w)$ , per ogni vertice.

**Teorema**

Un grafo connesso  $G = \langle V, A \rangle$  ha un cammino euleriano se e solo se  $G$  è semi-euleriano.  $G$  ha un ciclo euleriano se e solo se  $G$  è euleriano.

# Grafi Euleriani 2

**Teorema**

Sia  $G = \langle V, A \rangle$  un grafo semi-euleriano e sia  $P$  un cammino da  $s$  a  $t$ . Sia  $G_1$  il grafo ottenuto da  $G$  togliendo tutti gli archi di  $P$ . Allora  $G_1$  è euleriano.

**Teorema**

Sia  $G = \langle V, A \rangle$  un grafo euleriano e sia  $C$  un ciclo di  $G$ . Sia  $G_1$  il grafo ottenuto da  $G$  togliendo tutti gli archi di  $C$ . Allora  $G_1$  è euleriano.

# Altre fasi

- bubble popping
- tip removal

# Scaffolding

- Fondere contigs in scaffolds
- usando mate pairs
- anche con revcomp

# Licenza d'uso

Quest'opera è soggetta alla licenza Creative Commons: Attribuzione-Condividi allo stesso modo 4.0. (<https://creativecommons.org/licenses/by-sa/4.0/>). Sei libero di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire, recitare e modificare quest'opera alle seguenti condizioni:

- Attribuzione — Devi attribuire la paternità dell'opera nei modi indicati dall'autore o da chi ti ha dato l'opera in licenza e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera.
- Condividi allo stesso modo — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa.