

Bioinformatica

UniShare

Davide Cozzi
@dlcgold

Indice

1	Introduzione	2
2	Introduzione alla bioinformatica	3
2.1	Breve introduzione biologica	4
2.2	Progetto Genoma Umano	5
2.3	Variazioni	6
2.4	Pangenoma	7
2.5	Progetti attuali	9
2.6	Sequenziamento del DNA	10
3	Grafi di assemblaggio	12
3.1	Grafi in bioinformatica	13
3.1.1	Superstringhe e grafo di overlap	14
3.1.2	Grafi di De Bruijn e k-mers	21

Capitolo 1

Introduzione

Questi appunti sono presi a lezione. Per quanto sia stata fatta una revisione è altamente probabile (praticamente certo) che possano contenere errori, sia di stampa che di vero e proprio contenuto. Per eventuali proposte di correzione effettuare una pull request. Link: <https://github.com/dlccgold/Appunti>.

Capitolo 2

Introduzione alla bioinformatica

La genomica ha dimostrato negli ultimi anni ha dimostrato una capacità incredibile di produrre dati e questo ha portato alla nascita del bioinformatico, che diventa un esperto della gestione di questi dati sia dal punto di vista algoritmico che dal punto di vista sistemistico.

A partire dal 2000/2001 ma soprattutto poco prima del 2010 si ha una crescita dei **dati genomici** non indifferente. I dati genomici sono quelli provenienti dal sequenziamento del DNA. Negli ultimi anni questa crescita ha superato la curva della **legge di Moore** quindi la crescita in termini di hardware (che si stima migliorare ogni 18 mesi) non riesce più a soddisfare la stima di richiesta di hardware necessario per il sequenziamento. Questa stima di sequenziamento è basata su Illumina, che produce le più diffuse macchine fisiche per il sequenziamento. Case farmaceutiche e laboratori che studiano il sequenziamento hanno almeno una macchina Illumina. La quantità di dati ha raggiunto i livelli dei petabyte e quindi ci si aspetta (e in parte già è così) che l'hardware non sia più in grado di elaborare tali dati.

La bioinformatica riceve quindi questa tipologia di dati. La bioinformatica è cruciale nell'ambito della ricerca in biologia molecolare (riguardante prettamente DNA), dove sempre più si ha necessità dell'appoggio dell'informatica, avendo a che fare con dati, nel dettaglio grandi dati.

Un altro aspetto è quello legato alle nanotecnologie e alla così detta **DNA-based computation**. Un esempio è legato al fatto che ormai si è in grado di manipolare il DNA al punto di essere in grado di assemblarlo in laboratorio, tramite un meccanismo a *tiling* (*tasselli*), dove il tiling tendenzialmente è una figura regolare (triangolare, rettangolare, esagonale, etc...) con cui si compone del materiale biologico. Si riescono a fare letteralmente figure con il DNA (anche stelle, smile etc...) ma, soprattutto di questi tempi, vaccini,

che sono appunto manipolazione genetica di DNA o RNA. **Questa parte non è trattata nel corso.**

2.1 Breve introduzione biologica

Nel corso tratteremo prevalentemente sequenze di DNA. All'interno della cellula si hanno i **cromosomi** e un **genoma** altro non è che la collezione di cromosomi all'interno di un individuo. Il singolo cromosoma è rappresentato da filamenti di DNA “attorcigliati”. Il cromosoma sostanzialmente è formato dalla coppia di due filamenti che si uniscono in una parte centrale detta **centromero**. I cromosomi, dal punto di vista informatico, sono vere e proprie sequenze (con i 4 nucleotidi, adenina, citosina, guanina e timina, ricordando la complementarità delle basi A-T C-G), anche se si hanno varie regole per gestire questa “semplificazione”. Un altro aspetto è il passaggio dal DNA alle **proteine**, anche se nel corso non verrà trattata la **proteomica**, ovvero lo studio delle proteine in se. In merito al passaggio da DNA a proteine si ha che il DNA contiene i **geni** da cui poi derivano le proteine. Un gene può portare a più di una proteina e questo si è scoperto grazie al sequenziamento. Allo stato attuale per “leggere” il DNA di un individuo dobbiamo passare per macchine di sequenziamento che però non possono leggerlo interamente ma, prendendo il DNA da una provetta (anche a partire da una singola cellula nel **sequenziamento single-cell**), si ha in output un file con dei frammenti del DNA originale, replicati in coppie, dette **read**. Tramite vari algoritmi siamo poi in grado di arrivare a capire e studiare il DNA per poi arrivare, si spera, ad uno dei principali fini della bioinformatica, quello di curare la vita, tramite terapie mediche (si parla di **medicina traslazionale**, ovvero non curo un paziente tramite protocolli generali ma sulla base del DNA del paziente, che viene studiato ai fini di stabilire la migliore terapia, che diventa personalizzata per l'individuo). Le scoperte biologiche più attuali sono ottenute praticamente sempre grazie all'intervento anche dell'informatica e della bioinformatica.

Un esempio di uso delle sequenze è confrontare regioni genomiche di varie specie per valutare eventuali somiglianze. Un primo modo è diretto, un secondo è confrontare dopo l'allineamento, con l'inserimento di gap (studieremo la cosa nel dettaglio).

Il bioinformatico fornisce al biologo/biotecnologo la strumentazione necessaria per fare le varie analisi.

2.2 Progetto Genoma Umano

Un elemento chiave nella bioinformatica è il **Human Genome Project** (*progetto genoma umano*), progetto partito prima del 2000 (la prima base è del 1990) con vari obiettivi:

- identificare tutti i circa 30.000 geni nel DNA umano
- determinare le sequenze dei 3 miliardi di coppie di basi chimiche che compongono il DNA umano
- memorizzare queste informazioni in banche dati/db
- migliorare gli strumenti per l'analisi dei dati

La bioinformatica è andata avanti quasi sempre con progetti globali e il Progetto Genoma Umano è stato il primo di questi progetti, diciamo che lì nacque la bioinformatica. Si hanno vari *milestones*:

- *1990*: progetto avviato come sforzo congiunto del U.S. Department of Energy e del National Institutes of Health (NIH)
- *Giugno 2000*: completamento di una bozza di lavoro dell'intero genoma umano
- *Febbraio 2001*: vengono pubblicate le analisi della bozza di lavoro
- *Aprile 2003*: Il sequenziamento del Progetto Genoma Umano è completato e il progetto è dichiarato finito due anni prima del previsto

Quest'anno, nel 2020, è stato lanciato un progetto ulteriore in quanto ora si è anche in grado di sequenziare il DNA nei pressi dei **telomeri**, ovvero le terminazioni dei cromosomi, che sono le regioni più difficili da ricostruire tramite il sequenziamento. Per farlo si hanno algoritmi e software davvero molto sofisticati.

Vediamo qualche numero:

- il genoma umano contiene 3 miliardi (3×10^9) di basi nucleotidiche chimiche che sono 4:
 - adenina (A)
 - citosina (C)
 - guanina (G)

– timina (T)

- il gene mediamente è composto da 3000 basi, ma le dimensioni variano molto, con il più grande gene umano noto che è la Distrofina con 2.4 milioni di basi
- il numero totale di geni è stimato a circa 30000, molto inferiore alle stime precedenti da 80000 a 140000 (in quanto prima c'era il dogma che un gene codificasse una sola proteina, e si avevano circa 140000 proteine, che si conoscevano anche solo per le analisi del sangue)
- quasi tutte (99.9%) le basi nucleotidiche sono esattamente le stesse in tutte le persone. Basta lo 0.1% di differenze tra basi per “fare la differenza”, anche differenziando predisposizioni geniche per una certa malattia
- le funzioni sono sconosciute per oltre il 50% del gene scoperto

Vediamo anche qualche numero (in stima) in merito agli organismi più studiati dai bioinformatici (spesso organismi con poche basi), più l'attualissimo *sars-cov-2*:

organismo	numero basi	numero di geni
uomo (Homo sapiens)	3 miliardi	30000
topo di laboratorio (M. musculus)	2.6 miliardi	30000
arabetta comune (A. thaliana)	100 milioni	25000
nematoda (C. elegans)	97 milioni	19000
mosca della frutta (D. melanogaster)	137 milioni	13000
lievito (S. cerevisiae)	12.1 milioni	6000
batterio (E.coli)	4.6 milioni	3200
Human immunodeficiency virus (HIV)	9700	9
sars-cov-2	~27 milioni	~15

2.3 Variazioni

Una volta conosciuta la sequenza dell'uomo si è cercato di studiare quello 0.1% di differenze tra vari esseri umani. Queste differenze sono dette **SNPs** (*single nucleotide polymorphisms*) (detti a voce “snips”) che rappresentano la variabilità nella popolazione umana. Sono le differenze a livello di singolo nucleotide. Subito dopo il Progetto Genoma Umano è partito, sempre

tramite il National Institutes of Health (NIH), un progetto che confrontasse popolazione africana, asiatica e statunitense per calcolare queste differenze, individuate tramite tool informatici, tramite il cosiddetto **assemblaggio di aplotipi**, che è prettamente un problema informatico, *NP-complete*, la cui soluzione più recente è data da un **algoritmo parametrico**. Dagli aplotipi vengono estratti gli SNPs e questo sarà visto tra qualche lezione. Gli SNPs sono serviti a determinare differenze tra le varie popolazioni campione in merito, ad esempio alla predisposizione alla Talassemia nelle popolazioni mediterranee. Questi studi servono appunto capire le predisposizioni delle varie popolazioni. Se una popolazione ha, nella maggior parte dei casi, una certa base in una certa posizione allora si ha uno SNPs. Il famoso 0.1% forma questi SNPs, il 99.9% della popolazione porta il cosiddetto **allele di maggioranza** mentre lo 0.1% l'**allele di minoranza**.

Uno studio ha dimostrato che, in Italia, solo i Sardi hanno un profilo genetico ben definito, tutti gli altri sono dei “mix genetici” e questo si è scoperto studiando gli SNPs.

Dal Progetto genoma Umano si è poi passati a confrontare il genoma di piccolissimi campioni, ad esempio 1000 individui, con il 1000 Genomes Project, un altro progetto con sforzi internazionali, fatto per mappare le variazioni su una popolazione di 1000 individui. Si segnala che per sequenziare un individuo ci sono voluti 10 anni nel primo caso ma poi ci è voluto molto meno. Ora un singolo individuo si sequenzia in qualche ora, a costi molto ridotti. Dal DNA si sono anche ricavati i flussi migratori avvenuti nel corso della storia.

2.4 Pangenoma

Si vedrà, durante il corso, che dire **il genoma è una singola sequenza**, è ormai sostanzialmente errato. Avendo sequenziato milioni di individui si parla di **pangenoma** e le analisi devono ormai essere fatte non su un singolo genoma di riferimento ma si usa quello abbinato a tutta la serie di 0.1% di SNPs individuati finora. Nel dettaglio un pangenoma è una collezione di genomi multipli che sono correlati tra loro (variando solo in pochi punti). Si ha il pangenoma dell'uomo, di un batterio etc...

Dal punto di vista informatico diciamo comunque che il DNA è una sequenza sotto l'assunzione della **complementarietà delle basi**:

- adenina e timina sono complementari
- citosina e guanina sono complementari

e questo mi permette di poter studiare solo uno dei due filamenti del DNA.

Esempio 1. *Sia data la sequenza:*

$$S = acctacga$$

la complementare è:

$$S' = tggatgct$$

Se prendo la sequenza (o meglio una porzione di essa) di S_1 di un individuo h_1 e la sequenza S_2 di un individuo h_2 avrò un'alta somiglianza con eventualmente uno o più SNPs.

La posizione dello SNP è detto **locus**. Uno SNP si ha quando nel 99.9% dei casi tutti gli individui hanno una certa base in una data posizione, avendo l'*allele di maggioranza*, mentre lo 0.1% degli individui ne ha una diversa, avendo l'*allele di minoranza* (e lo rilevo confrontando una popolazione).

Esempio 2. *Si hanno:*

$$S_1 = acctacga$$

$$S_2 = accgacga$$

ho uno SNP nel locus 4. Ipotizzando che il 99.9% degli individui siano come l'individuo con la sequenza s_1 ho che la base t è un allele di maggioranza mentre la base g è un allele di minoranza.

L'uomo si dice essere **biallelico** in quanto le "opzioni" per una certa posizione sono solo due. Alcuni cambiamenti possono anche essere del tipo *inserzione/delezione* (anche per sequenze di più basi contigue), parlando di **variazioni strutturali** (che sono comunque più complesse e meno tipiche). Per rappresentare il fatto che si hanno più sequenze con queste variazioni, soprattutto se sono inserimenti e delezioni, ma considerando che il 99.9% delle basi è uguale (cercando quindi una rappresentazione che ottimizzi questa cosa), rappresentando quindi un pangenoma, dal punto di vista computazionale è un **grafo**. Ogni sequenza identica collassa in un solo nodo, avendo poi singoli nodi per le variazioni.

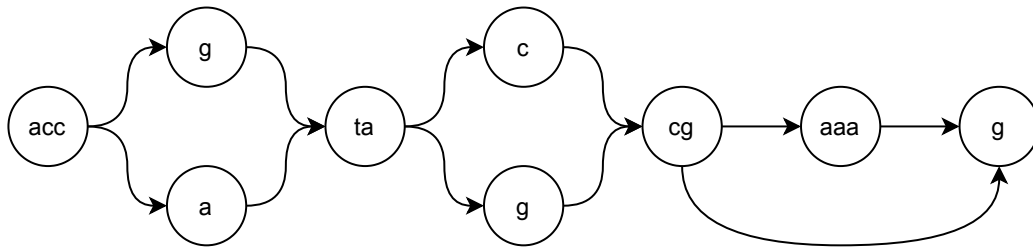
Esempio 3. *Ipotizzo di avere (con – per indicare delezioni):*

$$S_1 = accgtaccgaaag$$

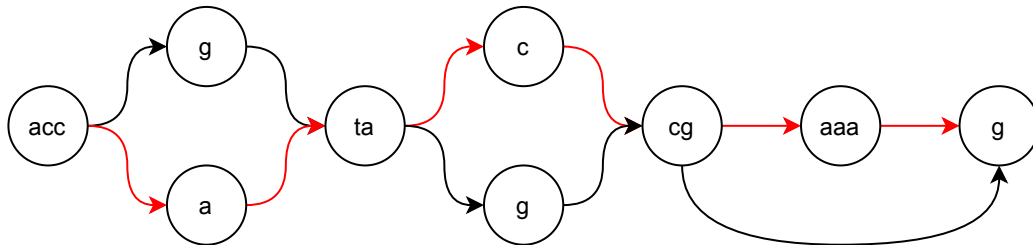
$$S_2 = accatagcgaaag$$

$$S_3 = accgtaccg---g$$

E ottengo un grafo del tipo:



Studiando i cammini dei grafi ottengo tutte le rappresentazioni. Questa rappresentazione però ha dei difetti, in quanto potrei avere cammini che non rappresentano nessuna sequenza di partenza. Pensando all'esempio sopra potrei avere il cammino in rosso che non rappresenta nessuna delle tre sequenze:



Rappresento quindi più di quello che voglio rappresentare. Un pangenoma è un grafo che rappresenta una popolazione senza fare grandi distinzioni, avendo percorsi che non sono riscontrabili in nessun individuo della popolazione. Si ha comunque che il concetto di sequenza non è più adeguato. Il grafo di una popolazione è enorme e comunque, tramite colori, si possono distinguere i vari percorsi della popolazione (distinguendo facilmente “tracce comuni”). Parlando quindi di **genoma di riferimento** o si parla di quello specifico di un individuo o si parla del pangenoma di una popolazione, con le varianti.

Dal punto di vista di *file* le varianti vengono date in un file **Variant Call Format (VCF)**. L'input classico dei software è quindi spesso un VCF, così come l'output.

2.5 Progetti attuali

Vediamo ora quali sono i grandi progetti su larga scala attualmente in corso:

- **The Cancer Genome Atlas Pan-Cancer Analysis Project (TCGA)**, che cerca di costruire un catalogo delle caratteristiche

genomiche dei tumori, ovvero un catalogo delle mutazioni genomiche associate a tumori (ad esempio quello del seno si sa che è legato alla mutazione del gene BRCA che si sa bene dov'è)

- **The 1000 Genomes Project Consortium: A global reference for human genetic variation**, che cerca di ricostruire e raffinare un sequenziamento di diversi genomi per costruire un genoma di riferimento per una popolazione, nel dettaglio umana, (in formato VCF)
- **Trans-Omics for Precision Medicine**, il progetto per la medicina traslazionale
- **The Computational Pangenome Consortium**, che mira a studiare nuovi strumenti software che possano trattare il grafo del pangenoma visto che la maggioranza del software attuale ancora funziona su sequenze e non su grafi

2.6 Sequenziamento del DNA

Il sequenziamento (che letteralmente significa “produrre la sequenza”) solitamente si svolge concatenando diverse operazioni:

1. estrazione del DNA
2. si ha una “libreria preparatoria” dove si mette del materiale genetico su un materiale preparatorio
3. si ha un meccanismo di “copie” tramite PCR o simili
4. si mettono i sample genomici in una macchina di sequenziamento che produce in output i dati

Un genoma non può essere letto “nucleotide per nucleotide” e i biologi, con la tecnologia attuale producono le cosiddette **read** del DNA originali. Si hanno due tipi di read:

- **read**, dette anche **short read**, lunghe circa 100 basi. Illumina produce tendenzialmente 100 o al più 150 basi
- **long read**, lunghe circa 10000 basi (se non di più, anche 20000)

Per ottenere il sequenziamento si ha un processo in cui:

- si divide il genoma in due parti, “aprendo” il filamento di DNA per permetterne la lettura
- si ha la **generazione delle read** da copie multiple del genoma tramite un processo biologico svolto dai macchinari, che sfruttano processi chimici
- si ha poi l'**assemblaggio dei frammenti**, ovvero un processo computazionale dove tramite algoritmi si assemblano le varie read per ottenere il genoma di partenza, avendo che le read hanno pezzi in *overlap*

Il problema del sequenziamento risale alla fine degli anni settanta con Sander e Gilbert che avevano studiato un processo di replicazione dando le basi allo studio del sequenziamento.

Dopo il sequenziamento dell'uomo si è passati a sequenziare molti altri organismi.

Oggi il sequenziamento è reso semplice dalla tecnologia. Un esempio è la tecnologia MinION, così piccola sta stare in una mano, che produce *long read* (anche se comunque con diversi errori). MinION è una tecnologia di *Oxford Nanopore*. MinION è USB ed è fatta per biologi che devono sequenziare in situazioni d'emergenza (esempio banale un biologo in Africa in piena emergenza Ebola). L'elaborazione dati viene fatta da un server.

Il primo sequenziamento è costato 3 miliardi di dollari per diversi anni, ora si fa in meno di 40 ore a 5000 dollari. Di recente si è passati addirittura a poche ore per un costo di circa 1000 dollari. Tornando alla *legge di Moore* si ha che il costo è collassato rispetto alla legge e quindi la capacità delle tecnologie di sequenziamento è molto maggiore della capacità di processare i dati, per quanto visto ad inizio capitolo. Si hanno quindi tanti dati ma non si è in grado di elaborarli.

Si tratterà anche il **confronto di genomi** per studiare poi gli aspetti evolutivisti, tramite **alberi evolutivi**, anche **alberi evolutivi tumorali**. Il **confronto tra sequenze** permette di studiare le evoluzioni, anche quelle tumorali, dove si hanno mutazioni radicali di DNA. Approfondiremo anche tali mutazioni e il loro effetto (basta il cambio di una base per portare, ad esempio, all'anemia falciforme). Studieremo quindi anche come fare gli **allineamenti**. Approfondiremo il discorso della **filogenesi** e della **filogenesi tumorale**.

Tutto questo, in questo ultimo anno, è stato applicato allo studio di **sars-cov-2**, avendo lo studio delle variazioni.

Verrà approfondito anche il discorso del **riarrangiamento**.

Capitolo 3

Grafi di assemblaggio

La prima tematica che affrontiamo è l'assemblaggio delle read tramite grafi. Per questo problema abbiamo quindi:

- **input:** collezioni di read (short read e/o long read)
- **output:** grafo di assemblaggio da cui estrarre un cammino o un'unica sequenza

Si hanno principalmente due tipi di grafo:

- **grafo di De Bruijn (*DBG*)** (*si legge “grafo di de broin”*), che si prestano più per *short read* (da 100 o 150 basi)
- **grafo di overlap**, più comodo in caso di *long read*

Si useranno per questi scopi varie nozioni, tra cui:

- relazione di prefisso/suffisso tra k-mers
- relazione di prefisso/suffisso tra read
- Longest Common Prefix tra sequenze
- estrazione di cammino di Eulero dal grafo
- estrazione di cammino Hamiltoniano dal grafo
- Maximal Exact Matches (*MEMs*)
- Burrows Wheeler Transform (*BWT*)
- indici succinti (come FM-Index)

- suffix tree e suffix array
- bloom filters, nati in ambito fisico e usati ora in ambito BigData
- min-hash e min-sketch, usati anche nelle reti neurali e nel Deep Learning quando si ha a che fare con grandi moli di dati

Studiare i grafi di assemblaggio può essere utile anche in ottica di applicare procedimenti simili ad altri problemi posti dai biologi.

3.1 Grafi in bioinformatica

In bioinformatica infatti uno strumento molto usato, anche oltre il sequenziamento, è quello dei **grafi**.

In letteratura la nozione di grafo compare nel 1735 con il **grafo di Eulero**, con Eulero che, si dice, fosse ossessionato dal problema dei **ponti di Königsberg**, volendo trovare il ciclo che attraversasse ogni ponte solo una volta. Ogni isola di Königsberg diventava un nodo e ogni ponte tra isole un arco tra nodi. Da qui la definizione del problema.

Definizione 1. *Il **problema del ciclo Euleriano** consiste nel trovare un ciclo in un grafo tale che visiti ogni arco una e una sola volta prima di tornare al punto di partenza. Si può passare dallo stesso nodo più volte.*

*Questo problema si dimostra risolvibile in **tempo lineare** sull'input $G = (V, E)$.*

Vediamo poi il “problema duale”, quello in cui si vuole fare un ciclo che non visiti due volte uno stesso nodo.

Definizione 2. *Il **problema del ciclo Hamiltoniano** consiste nel trovare un ciclo in un grafo tale che visiti ogni vertice una e una sola volta prima di tornare al punto di partenza.*

*Questo problema si dimostra essere **NP-complete**.*

La differenza di complessità di questi due problemi sarà qualcosa che bisognerà considerare parlando dello studio dei grafi in bioinformatica. Anche solo il problema dell'assemblaggio si vedrà essere riducibile alla visita di un grafo (quindi non potremo formularlo come un problema di ciclo Hamiltoniano, la cui soluzione potrebbe richiedere anni).

La comparsa dei grafi nel mondo chimico è intorno a metà del 1800 con Cayley che li usò per rappresentare strutture chimiche, nel dettaglio usò **alberi** (che ricordiamo esserei grafi connessi aciclici) per contare gli isomeri strutturali.

In biologia l'uso dei grafi è stato introdotto a metà 1900 con l'esperimento di Benzer, che capì l'importanza dei grafi mentre cercava di distinguere quando determinati virus attaccano determinati batteri. Benzer è riuscito a mostrare che il DNA di questi virus era *lineare* mentre prima si congetturava che il DNA avesse delle biforcazioni. Per capire che non avesse delle biforcazioni ha sfruttato la capacità di alcuni geni dei virus di aggredire batteri, rappresentando la cosa coi **grafi ad intervallo**.

Definizione 3. Nella teoria dei grafi, un **grafo d'intervallo** è il grafo d'intersezione di un multiinsieme di intervalli sulla linea reale. Ha un solo vertice per ciascun intervallo dell'insieme, e uno spigolo tra ogni coppia di vertici corrispondenti agli intervalli che intersecano.¹

In poche parole associo una lettera ad ogni intervallo e collego nel grafo i vertici corrispondenti alla lettera qualora i due intervalli abbiano sovrapposizioni.

Il punto di svolta si ha però nel 1977 col sequenziamento e i due metodi di Sanger (che è tutti gli effetti il primo metodo di sequenziamento) e Gilbert, entrambi chimici. Entrambi i metodi generano frammenti etichettati di lunghezza variabile che vengono “letti” tramite elettroforesi.

Non approfondiamo nel dettaglio i metodi, essendo prettamente chimici e biologici.

3.1.1 Superstringhe e grafo di overlap

L'assemblaggio dei frammenti del DNA è invece un problema prettamente computazionale, avendo l'assemblaggio dei singoli frammenti, ovvero delle **read** prodotte dal sequenziamento, anche in più copie, in un'unica sequenza genomica, detta **superstringa**. *Fino alla fine degli anni '90 l'assemblaggio di frammenti del genoma umano era visto come un problema intrattabile.*

Definizione 4. Definiamo **stringa** come la concatenazione di simboli di un alfabeto Σ .

In bioinformatica spesso si ha $\Sigma = \{a, c, g, t\}$

Definizione 5. Definiamo il **shortest superstring problem (SSP)** come la ricerca, dato un insieme di stringhe, di trovare la più corta superstringa che le contiene tutte. So hanno quindi:

- **input:** una collezione s_1, s_2, \dots, s_n di stringhe che possono anche essere lunghe uguali o a lunghezza variabile

¹https://it.wikipedia.org/wiki/Grafo_d%27intervallo

- **output:** una stringa s che contiene tutte le stringhe s_1, s_2, \dots, s_n dell'input come sottostringhe tale che $|s|$, ovvero la lunghezza della stringa s , sia **minima**

Questo problema è **NP-complete** e assume che non ci siano errori di sequenziamento nella produzione delle stringhe s_1, s_2, \dots, s_n .

La shortest superstring potrebbe non essere unica.

Esempio 4. Vediamo un esempio di shortest superstring. Si assume per semplicità alfabeto binario $\Sigma = \{0, 1\}$.

Si ha la collezione di stringhe binarie in input (che nel dettaglio sono tutte le possibili combinazioni di 3 simboli binari):

$$C_I = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

Si può verificare che la shortest superstring è:

$$s = 0001110100$$

Con la shortest superstring ho letteralmente assemblato le stringhe in input.

Le read determinano la **coverage (copertura)** del DNA. Per valutare il coverage vado a vedere ogni base da quante read è coperta. Con Illumina ho un coverage di almeno 50x, quindi ogni posizione è coperta da almeno 50 read (lunghe ciascuna ~150 basi). Per poter ricostruire la sequenza di DNA originale serve una certa quantità di coverage. Una coverage bassa potrebbe impedire la ricostruzione. Illumina va dal 50x minimo anche a 80x. MinION, della Oxford Nanopore, produce long read anche di 20000 basi ma con basso coverage, anche 3x, ma avendo read lunghe si riesce comunque ad assemblare. Quindi se ho long read mi basta un basso coverage mentre se ho short read mi serve un elevato coverage, avendo un insieme di read molto “fitto” e con poca “sparsità”, in quanto si avrebbero gap, con zone non coperte. Il coverage è comunque dato “per media” e quindi poter comunque avere buchi.

Il punto chiave che mi permette di ricostruire il DNA è la sovrapposizione tra le varie read. Il DNA inoltre ha ripetizioni e questo costituisce, purtroppo, un limite all'assemblaggio e in merito studieremo il **fragment assembly problem**, che serve anche in altri contesti, oltre a quello dell'assemblaggio del DNA. Fin'ora abbiamo anche trascurato anche un altro problema, gli **errori di sequenziamento**, dati dal fatto che il processo di sequenziare non è *ottimo*, ovvero privo di errori, dove con errore si intende che nel DNA si ha una certa base e nella read prodotta dal sequenziamento se ne ha un'altra. In fase di assemblaggio questo tipo di errore comporta che non si riesce a

sovrapporre bene le read, non potendo vedere più alcuni **overlap** tra coppie read. Si ha quindi **perdita di informazione dell'overlap** e diventa più complicato assemblare il DNA, non impossibile ma più complicato.

Esempio 5. *Si hanno un pezzo di DNA e tre read che sono sovrapponibili:*

	1	2	3	4	5	6	7	8
$DNA =$	a	c	c	g	t	a	c	g
$R_1 =$	a	c	c	g	t			
$R_2 =$			c	c	g	t	a	
$R_3 =$					g	t	a	c

Possiamo quindi assemblare il pezzo di DNA.

Ma se ipotizziamo di avere un errore di sequenziamento con la terza base della seconda read:

	1	2	3	4	5	6	7	8
$DNA =$	a	c	c	g	t	a	c	g
$R_1 =$	a	c	c	g	t			
$R_2 =$			c	c	c	t	a	
$R_3 =$					g	t	a	c

Diventa più difficile assemblare.

Il tasso di errore nei macchinari Illumina è dello 0.01%, avendo circa due errori per read lunga 150. Per MinION si ha un tasso d'errore anche di circa il 10%, quindi ogni 50 basi ho una serie d'errore. Di recente, in ambito long read, si stanno progettando i **PacBio HiFi** (con HiFi che qui sta per “high quality fragments”) che producono long read con tasso d'errore allo 0.1%, facendo ben sperare per il futuro.

Tra i primi informatici che hanno fatto sequenziamento abbiamo Gene Myers che era un esperto di algoritmi su stringhe e di pattern matching (parte attiva nella creazione dei suffix array), nonché responsabile della creazione dell'algoritmo di assemblaggio (famoso anche per BLAST). Gene Myers era un esperto del problema della shortest superstring. A partire dalla tecnica di costruzione della shortest superstring ha sviluppato l'algoritmo di assemblaggio. Vediamo quindi, in primis, come costruire la shortest superstring. Per farlo bisogna in primis capire come confrontare le varie stringhe in input e come “foldarle”. Per farlo faccio l'overlap che però a questo punto necessita di una definizione formale.

Definizione 6. Definiamo **overlap** tra una coppia di stringhe s_i e s_j in input come il più lungo prefisso di s_j che ha un match perfetto (coincide) con un suffisso di s_i . Posso anche dire che è il più lungo suffisso di s_i che ha un match perfetto con un prefisso di s_j , ribaltare la definizione non cambia. L'overlap tra le due stringhe si indica con:

$$ov(s_i, s_j)$$

Ricordiamo che una stringa la posso scrivere in modo scomposto in due modi:

- $s_i = s'_i x$, con x suffisso
- $s_j = x s'_j$, con x prefisso

Tendenzialmente si prende l'**overlap più lungo**.

Esempio 6. Siano:

$$s_i = accgtgtgt$$

$$s_j = gtgtgtccaa$$

Allora si ha che:

$$ov(s_i, s_j) = gtgtgt$$

con l'overlap lungo 6.

Proseguiamo quindi con il calcolo della shortest superstring dopo aver calcolato l'overlap di tutte le stringhe in input.

Creo un grafo con un nodo per ogni sequenza, etichettato con la sequenza stessa. Tracciamo quindi un arco tra due nodi sse i due nodi sono in overlap, associando all'arco la lunghezza dell'overlap.

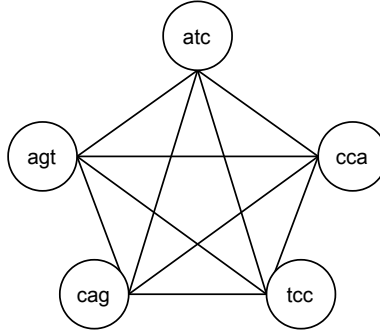
Una tecnica per fare il grafo consiste in:

- collegare a priori di tutti i nodi ottenendo un grafo completo non orientato
- per ogni coppia di stringhe s_i e s_j metto l'arco pesato con l'overlap massimo $ov(s_i, s_j)$, dando anche direzione all'arco. Eventualmente posso anche dare doppio peso all'arco in base alla direzione. Si è ottenuto il **grafo di overlap**, che quindi è un grafo orientato (se in entrambi i versi non ho overlap lo lascio per praticità senza orientamento con peso 0)

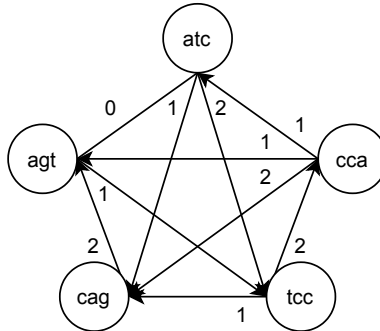
Esempio 7. Sia la collezione di stringhe in input:

$$C_i = \{atc, cca, cag, tcc, agt\}$$

e costruisco il grafo completo come detto sopra:



Aggiungo quindi i pesi relativi agli overlap dando l'eventuale orientamento e ottengo il grafo di overlap:



Per calcolare la shortest superstring dovremo calcolare un certo cammino sul grafo di overlap. Sicuramente un cammino che visita tutti i nodi mi porta ad avere una superstringa. Vediamo quindi una prima idea intuitiva:

Esempio 8. Riprendendo il grafo di overlap dell'esempio precedente faccio:

- parto dal nodo *atc* e lo aggiungo alla superstringa, che per ora è $s = atc$
- seguo l'arco di peso 2 e arrivo in *tcc*
- aggiungo *c* (ovvero la parte non in overlap) alla superstringa, che per ora è $s = atcc$
- seguo l'arco di peso 2 e arrivo in *cca*

- aggiungo *a* (ovvero la parte non in overlap) alla superstringa, che per ora è $s = atcca$
- seguo l'arco di peso 2 e arrivo in *cag*
- aggiungo *g* (ovvero la parte non in overlap) alla superstringa, che per ora è $s = atccag$
- seguo l'arco di peso 2 e arrivo in *agt*
- aggiungo *t* (ovvero la parte non in overlap) alla superstringa, che per ora è $s = atccagt$
- mi fermo avendo visitato tutti i nodi

Alla fine ho:

$$s = atccagt$$

che so essere una superstringa.

Si vede che il cammino, a conferma, tocca ogni vertice una e una sola volta, avendo un cammino Hamiltoniano ma, avendo i pesi, abbiamo a che fare con un **Traveling Salesman Problem (TSP)**. Dobbiamo però dimostrare che la superstringa ottenuta è anche la più breve.

Diamo però una piccola definizione formale del grafo di overlap.

Definizione 7. Definiamo il **grafo di overlap** $G_{ov} = (V, E)$ tale che, data una collezione di stringhe s_1, \dots, s_n :

- $V = \{s_1, \dots, s_n\}$
- E è definito in modo che ogni arco $(s_i, s_j) \in E$ è un arco orientato da s_i a s_j di peso $|ov(s_i, s_j)|$ (quindi pesato con la lunghezza dell'overlap)

Si dimostra poi che il **cammino Hamiltoniano di massimo costo** “produce” una shortest superstring. Facciamo una dimostrazione non formale. Innanzitutto per “produce” si intende che, dato il cammino prodotto da Hamilton di massimo costo, con i vertici etichettati dalle stringhe $s_{i,1}, s_{i,2}, \dots, s_{i,n}$, la superstringa si ottiene sapendo che una stringa $s_{i,j+1}$ che ha un prefisso in overlap con la precedente stringa $s_{i,j}$ la si può scrivere come:

$$s_{i,j+1} = ov(s_{i,j}, s_{i,j+1}) \cdot x_{i,j+1}$$

Possiamo anche dire che:

$$r(s_{i,j}, s_{i,j+1}) = x_{i,j+1}$$

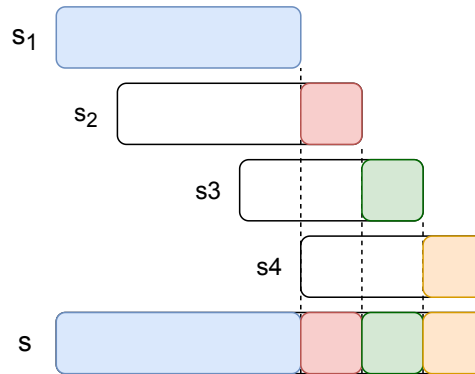


Figura 3.1: Esempio di formazione di una shortest superstring a partire da una collezione di stringhe sfruttando i “resti” degli overlap

indicando con $x_{i,j+1}$ la parte della stringa fuori dall’overlap, il “resto” possiamo dire. A questo punto so che la superstringa parte con $s_{i,1}$ e prosegue concatenando i vari $x_{i,j+1}$ (come si può vedere in figura 3.1):

$$s = s_{i,1} \cdot x_{i,2} \cdot x_{i,3} \cdot \dots \cdot x_{i,n}$$

Bisogna dimostrare che il cammino Hamiltoniano di massimo peso coincide con la shortest superstring. Bisogna dimostrare che:

- un cammino Hamiltoniano di massimo peso calcolato come sopra è una superstringa, e questo si dimostra perché tocca ogni vertice e quindi ogni stringa che di conseguenza viene inclusa
- la superstringa appena calcolata è la più breve e per dimostrarlo si ha l’intuizione che se massimizzo l’overlap “globale” minimizzo la lunghezza della superstringa

Il calcolo della shortest superstring può quindi risolvere l’assemblaggio di stringhe anche se **si ricorda che il problema del cammino Hamiltoniano è NP-complete**.

La miriade di read però, quando ci lavorò per primo Gene Myers, rendeva davvero difficile il calcolo (problema NP-complete e hardware storicamente poco potente). Servirono quindi anni per il primo calcolo, circa una quindicina, usando appunto il metodo della superstringa.

Si ha però un’euristica per calcolare la superstringa, usando un **algoritmo 2-approssimante** usando la **tecnica greedy**. In base a questa tecnica si sceglie sempre l’arco che pesa di più nel senso che ordino in ordine di peso tutti gli archi e faccio gli overlap tra le stringhe collegate. Dopo avere selezionato l’arco si prendono i due estremi e se ne fa la superstringa. Si continua

quindi cercando sempre gli archi che pesano di più creando poi la superstringa. **Non si vede nel dettaglio il funzionamento** e ovviamente non si ha la soluzione ottima e in realtà si congettura sia 2-approssimante ma non si ha una dimostrazione in merito, è un problema aperto da trent'anni e solo a livello sperimentale si è ipotizzata la 2-approssimazione.

Si ricorda che con il cammino Hamiltoniano ottimo si ottiene comunque una soluzione che potrebbe non essere unica.

3.1.2 Grafi di De Bruijn e k-mers

Il metodo della superstringa è stato quindi usato per l'assemblaggio del primo sequenziamento (quello con il metodo Sanger) ma l'appoggio ad un problema NP-complete (il *ciclo Hamiltoniano*) rendeva il tutto troppo dispendioso. Il primo *assemblatore*, quello di Celera, usava però questo metodo più lento per il *fragment assembly*.

Vediamo ora una soluzione diversa, basata sui **grafi di De Bruijn** che invece come problema sottostante ha il *ciclo Euleriano* che sappiamo avere soluzione lineare.

Vediamo in primis qualche definizione.

Definizione 8. Definiamo **k-mer** come è una sottostringa di lunghezza k . I k -mers sono quindi tutte le sottostringhe distinte di lunghezza k , non estraggo più volte lo stesso k -mer. Si segnala però che troppe ripetizioni dello stesso k -mer, che vengono trascurate, possono rendere difficile l'assemblaggio. Quindi il caso ideale è che tutti i k -mer estratti da una stringa siano distinti ma si seleziona il k in modo che ci siano al più due o tre ripetizioni dello stesso k -mer nella sequenza.

Definizione 9. Data una stringa s definiamo **spettro di s di dimensione/ampiezza l** è il **multiinsieme**, avendo quindi ripetizioni, di tutte le occorrenze di sottostringhe di lunghezza l (gli l -mers) e si indica con:

$$\text{spectrum}(s, l)$$

Esempio 9. Prendiamo una stringa s :

$$s = \text{tatggtac}$$

Fissiamo $k = 3$ e si ha lo spettro di dimensione 3:

$$\text{spectrum}(s, 3) = \{\text{tat}, \text{atg}, \text{tgg}, \text{ggt}, \text{gta}, \text{tac}\}$$

(che in questo caso, non avendo ripetizioni, è un insieme di 3-mers.)

Dati i frammenti/read (che sono short read lunghe circa 150 basi) si estraggono da essi i **k-mers**, con k usualmente pari a 32, 31 o 28 per avere il minor numero di ripetizioni (i numeri sono stati identificati sperimentalmente). Si segnala che questa “proprietà” di avere sequenze circa lunghe 32 che non si ripetono è probabilmente legata anche al fatto che il DNA, preso come sequenza di simboli, è difficile da comprimere con i tool standard (zip, uso della BWT etc...), creando non pochi problemi alle banche dati anche se ancora non si è scoperto bene ne perché ne come risolvere la cosa.

Il problema di assemblaggio diventa quindi ricostruire una stringa da un insieme di k-mer:

- **input**: un insieme di stringhe s_1, s_2, \dots, s_n
- estraggo i k-mer da tutte le s_i in input, per un k fissato
- assemblo i k-mer usando i grafi di De Bruijn

Ma prima di introdurre i grafi di De Bruijn vediamo un esempio di cosa significhi assemblare k-mers, partendo dal caso **senza ripetizioni**, avendo quindi un **insieme di k-mers** e non uno **spettro** (che è un multiinsieme).

Esempio 10. Si prenda in input una collezione di k-mers con $k = 3$ (quindi 3-mers):

$$C = \{atg, agg, tgc, tcc, gtc, ggt, gca, cag\}$$

Non essendo coincidenti se facessi gli overlap tra ogni coppia avrei al più overlap di lunghezza 2. Ipotizzando di fare il grafo di overlap avrei il seguente cammino Hamiltoniano (uno dei possibili):

$$atg \rightarrow tgc \rightarrow gca \rightarrow cag \rightarrow agg \rightarrow ggt \rightarrow gtc \rightarrow tcc$$

che produrrebbe la superstringa:

$$s = atgcagggtcc$$

Vedendo che anche con i k-mer posso ragionare in ottica di superstringa.

Ma con l’approccio che vogliamo ora non si passa per ogni vertice una e una sola volta ma per ogni arco una e una sola volta, usando il cammino Euleriano.

Dobbiamo fare sì che quindi prendere ogni arco una e una sola volta corrisponde a prendere tutti i k-mers, avendo quindi che ogni arco deve essere associato ad un k-mer. Costruiamo quindi un grafo che soddisfi queste condizioni (grafo che poi definiremo essere un **grafo di De Bruijn**). Si ha quindi

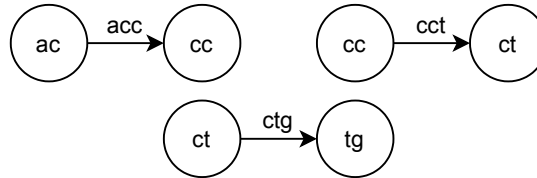
un grafo dove gli archi sono i k -mer mentre i vertici all'estremo di un arco sono etichettati con il prefisso di lunghezza $k - 1$ e il suffisso di lunghezza $K - 1$ del k -mer (quindi i suffissi e i prefissi unici formano l'insieme dei vertici). L'arco è orientato dal prefisso al suffisso.

In altri termini i vertici agli estremi di un arco etichettato con il k -mer x altro non sono che i due unici $(k-1)$ -mer estraibili da x .

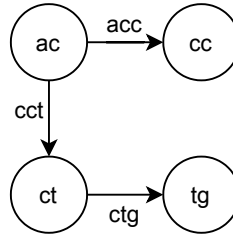
Esempio 11. Prendendo una collezione di k -mer:

$$C = \{acc, cct, cgt\}$$

so che, per il grafo $G = (V, E)$, ho $E = C$ mentre per V so che:



Quindi $V = \{ac, cc, ct, tg\}$.
Costruiamo quindi il grafo:



Quindi il cammino di Eulero (qui banale) e otteniamo la superstringa (e quindi l'assemblaggio) concatenando le etichette degli archi nell'ordine in cui vengono visitati, ragionando nello stesso modo in cui si faceva coi grafi di overlap, quindi partendo con la prima etichettata e proseguendo concatenando solo i resti dei vari overlap tra etichette degli archi:

$$s = acctg$$