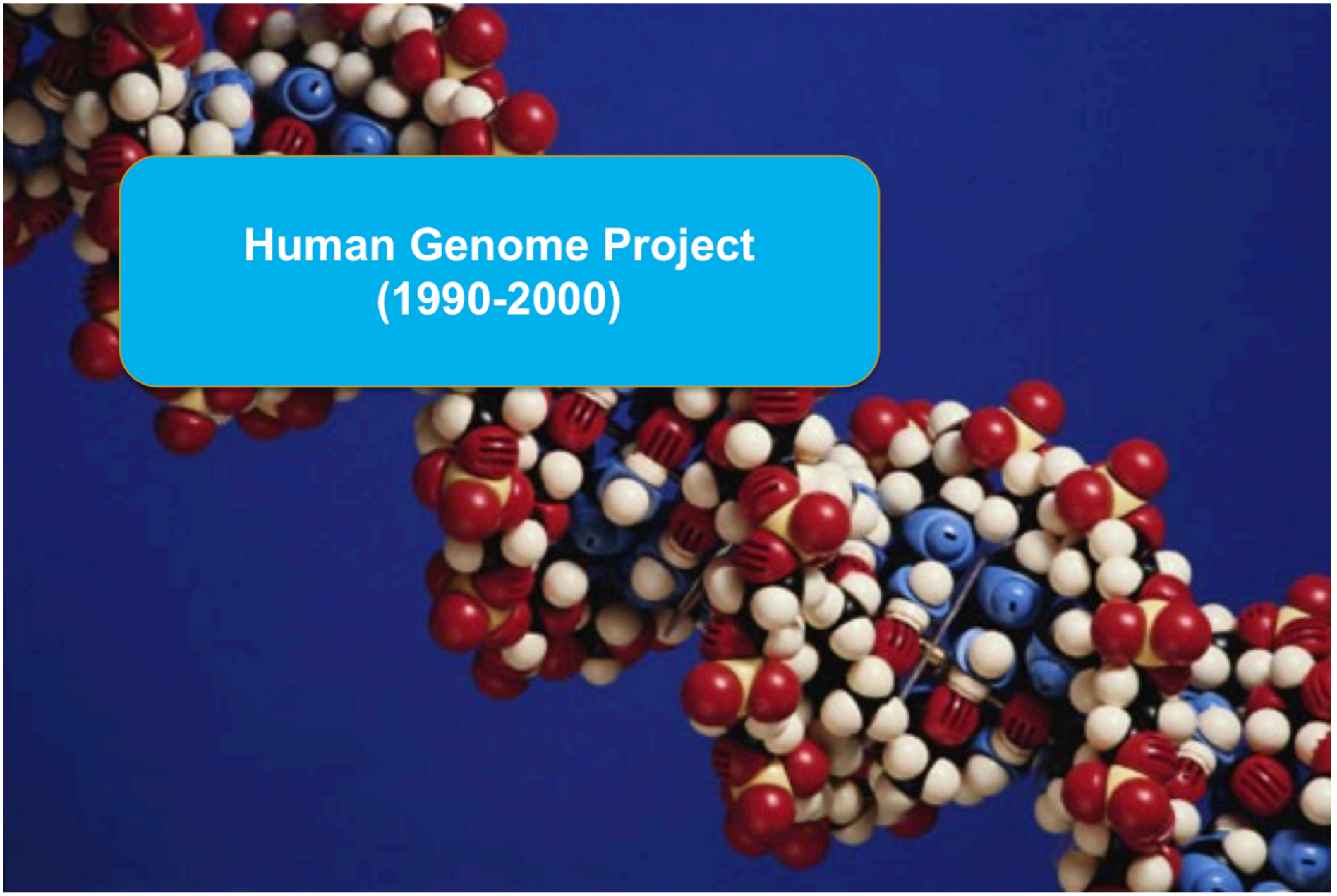
Sequenziamento

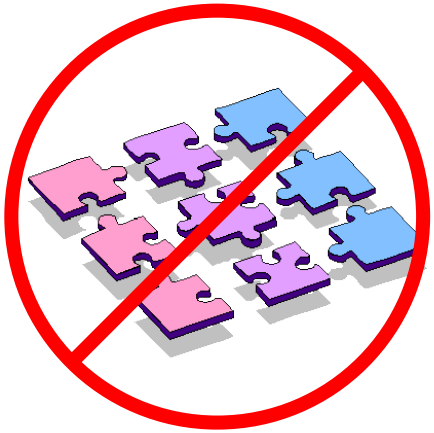
dal dato alle strutture dati (e algoritmi)

Shotgun Sequencing



What Makes Genome Sequencing Difficult?

- Modern sequencing machines cannot read an entire genome one nucleotide at a time from beginning to end (like we read a book)
- They can only shred the genome and generate short **reads**.
- The genome assembly is not the same as a puzzle: we must use *overlapping* reads to reconstruct the genome, a giant **overlap puzzle**!



Shotgun Sequencing→ reads

each fragment is individually sequenced —> primary structure

Fragment Assembly

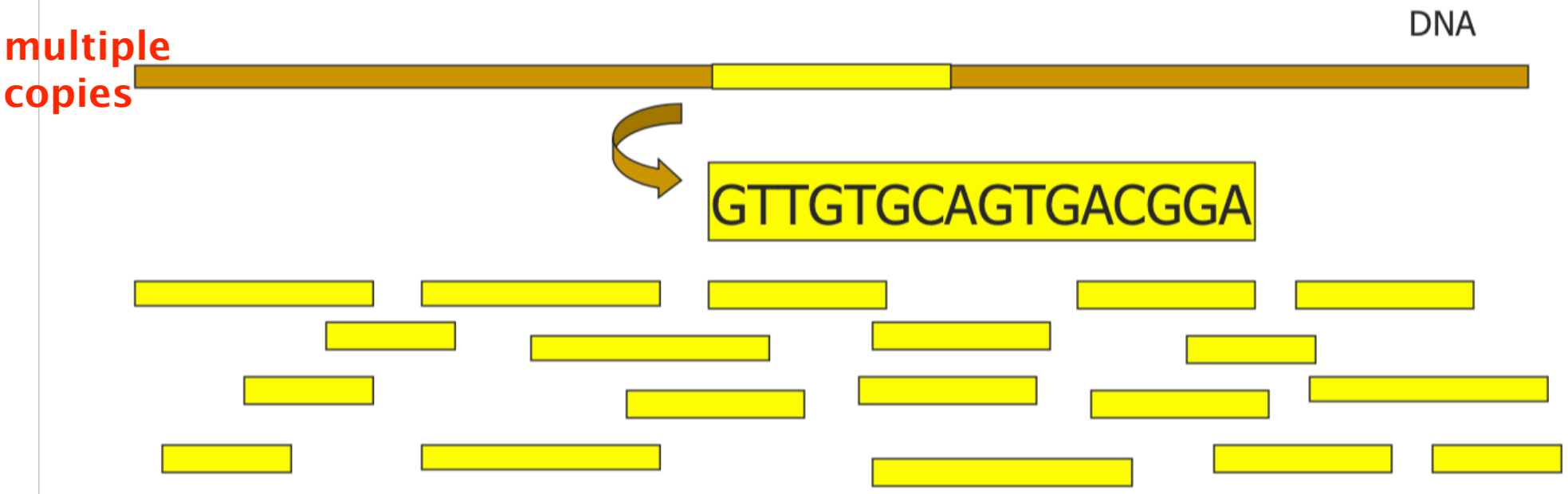
by using overlaps…

IL DATO DI SEQUENZIAMENTO

Single-end read

Cosa si ottiene da un esperimento di sequenziamento di una molecola di DNA (o RNA)?

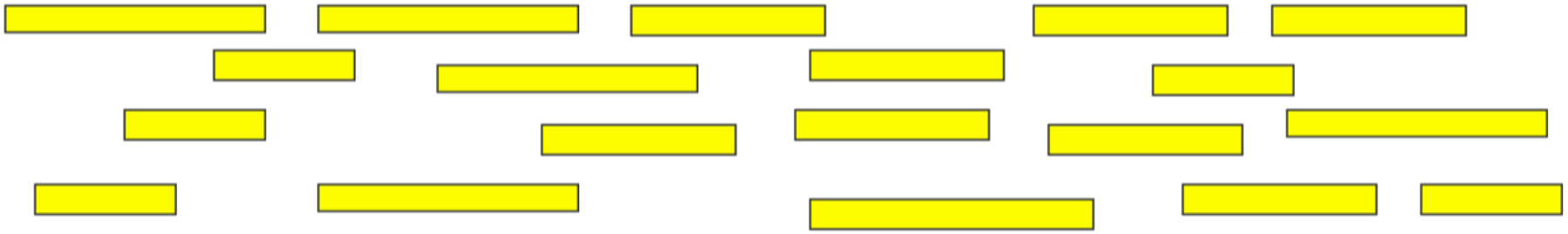
➔ single-end *read*



Single-end read

Cosa si ottiene da un esperimento di sequenziamento di una molecola di DNA (o RNA)?

➔ Il fragment assembly ha il compito di assemblare i reads sequenziata nella sequenza primaria originale



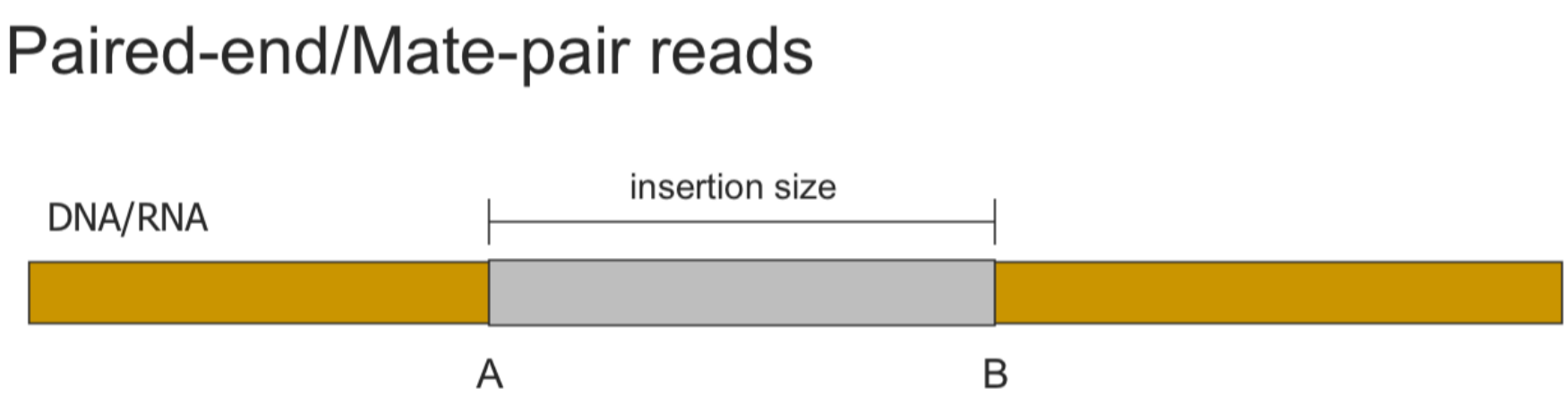
Single-end read

Cosa si ottiene da un esperimento di sequenziamento di una molecola di DNA (o RNA)?

➔ Il fragment assembly ha il compito di assemblare i reads sequenziata nella sequenza primaria originale

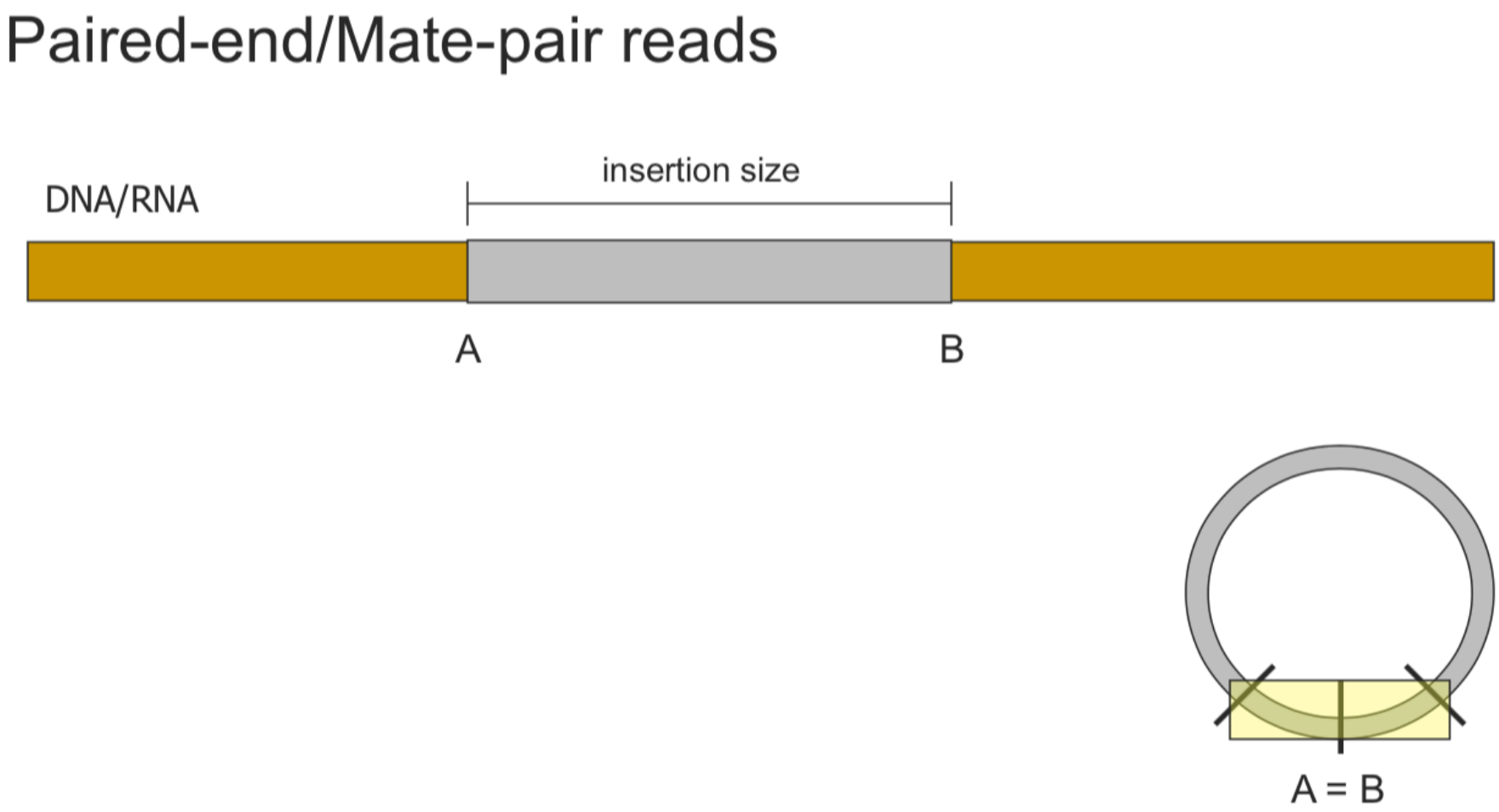
Pair-end / Mate-pair end

Altro tipo di dato ottenuto dal sequenziamento



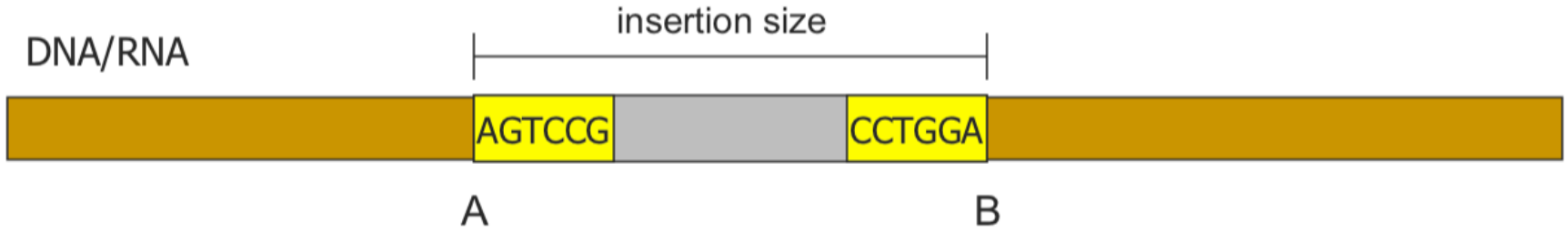
Considero la sottostringa tra A e B
(questo sarà sequenziato)

Pair-end / Mate-pair end

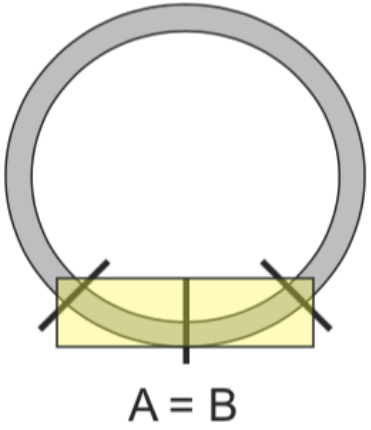


Pair-end / Mate-pair end

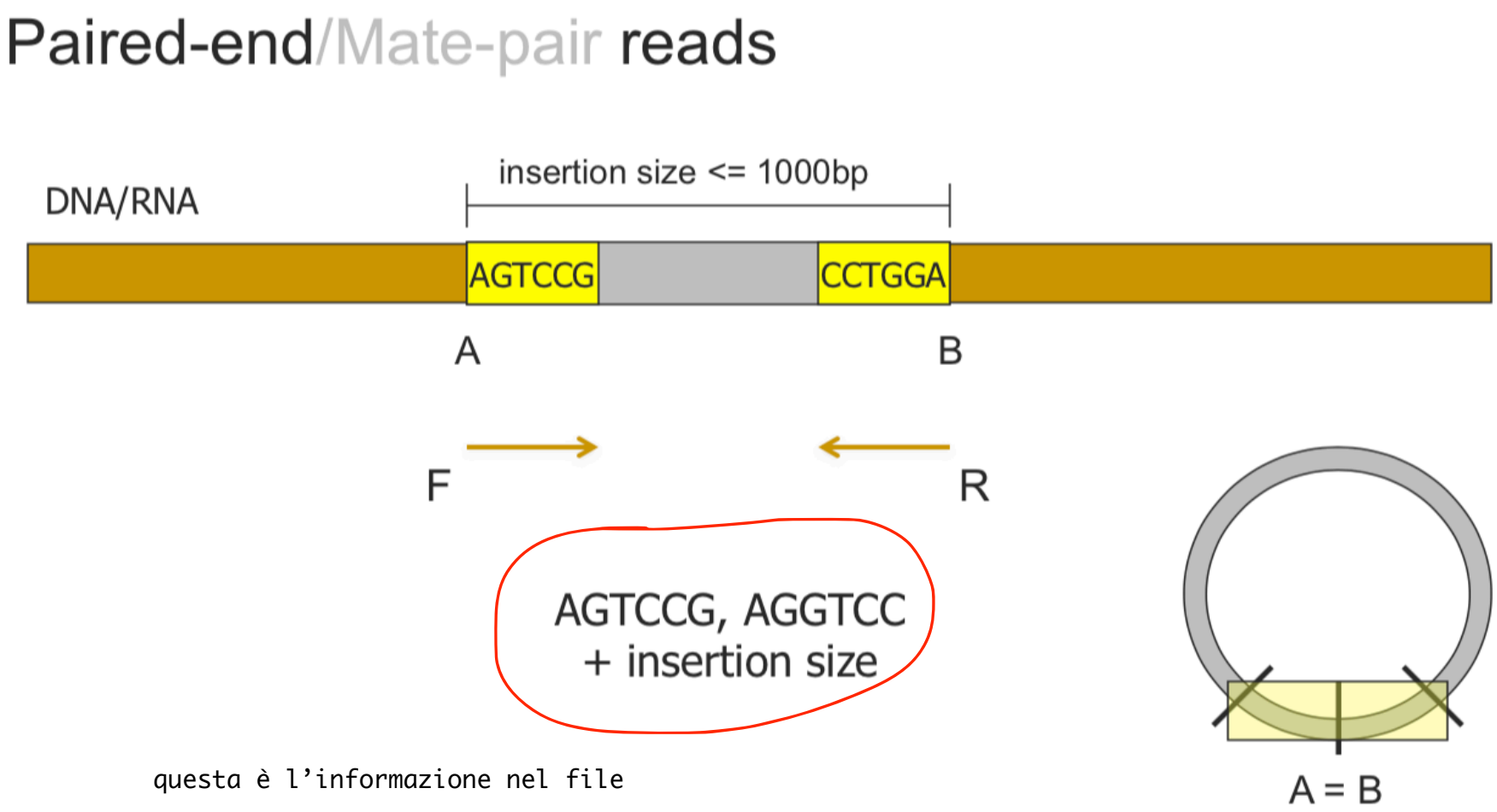
Paired-end/Mate-pair reads



i due frammenti sono logicamente accoppiati dall'insertion site (conosco la lunghezza che li separa)

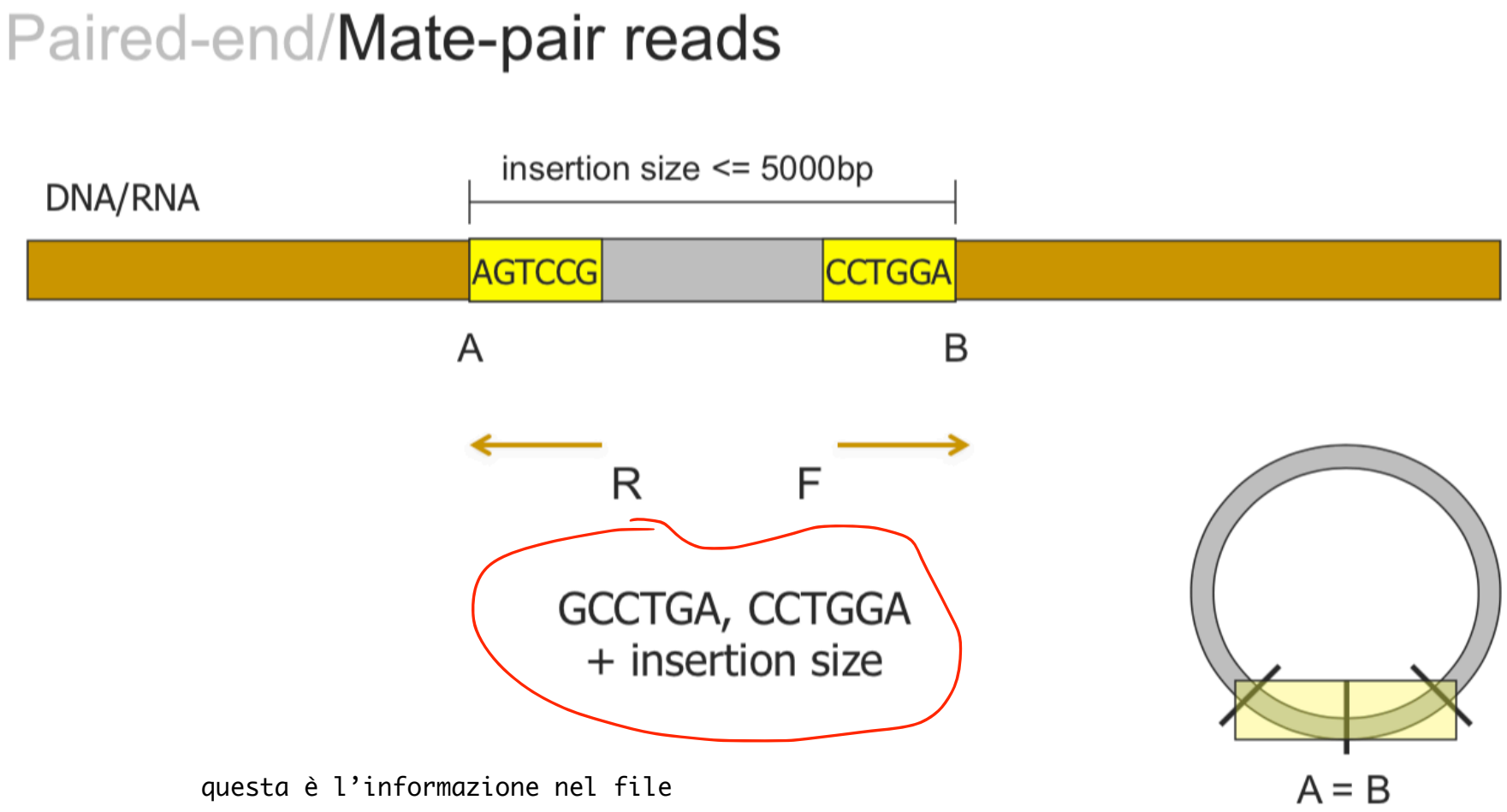


Pair-end / Mate-pair end



questa è l'informazione nel file

Pair-end / Mate-pair end



questa è l'informazione nel file

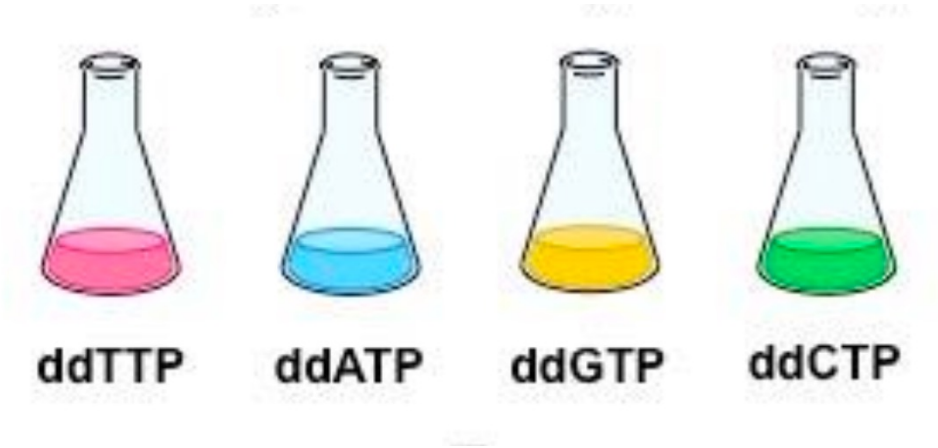
TECNICHE DI SEQUENZIAMENTO

Tecniche di sequenziamento

- ✓ 1977: Metodo tradizionale Sanger (first generation)
- ✓ 2000: Next-Generation Sequencing (NGS) (second generation)
- ✓ 2005: Next-Next-Generation Sequencing (third generation)

Metodo Sanger

Sanger → Chain-termination method



Componenti:

- ✓ DNA Template
- ✓ Primer
- ✓ DNA polimerasi
- ✓ Deossinucleosidi (dNTP → dATP, dCTP, dGTP, dTTP)
- ✓ Dideossinucleosidi (ddNTP –> ddATP, ddCTP, ddGTP, ddTTP)

Metodo Sanger

Sanger → Chain-termination method

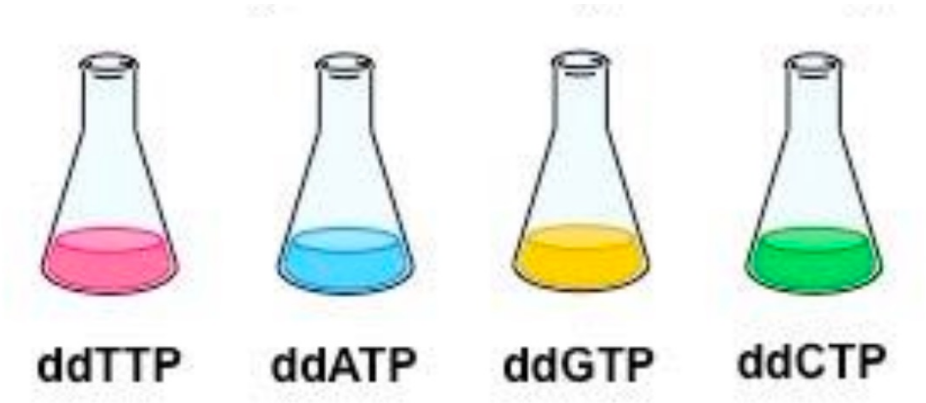
<u>Procedimento</u>:

- ✓ Denaturazione
- ✓ Annealing
- ✓ Elongazione
- ✓ Terminazione (con ddNTP)

T G C A G G C A T C T G A

| | | | | | | | | | | | |

A C G T C C G T A G A C T



Metodo Sanger

Sanger → Chain-termination method

Procedimento:

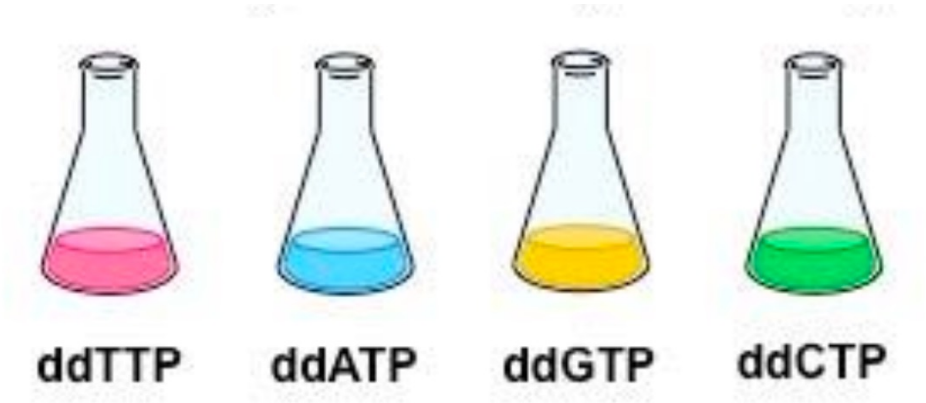
- ✓ Denaturazione
- ✓ Annealing
- ✓ Elongazione
- ✓ Terminazione (con ddNTP)

T G C A G G C A T C T G A

T G C

| | | | | | | | | | | | |

A C G T C C G T A G A C T



Metodo Sanger

Sanger → Chain-termination method

Procedimento:

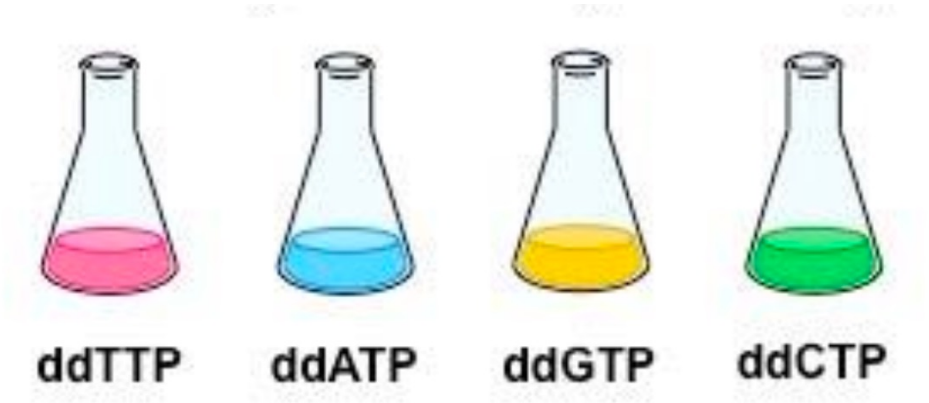
- ✓ Denaturazione
- ✓ Annealing
- ✓ Elongazione
- ✓ Terminazione (con ddNTP)

T G C A G G C A T C T G A

T G C A G G C

| | | | | | | | | | | | |

A C G T C C G T A G A C T



Metodo Sanger

Sanger → Chain-termination method

Procedimento:

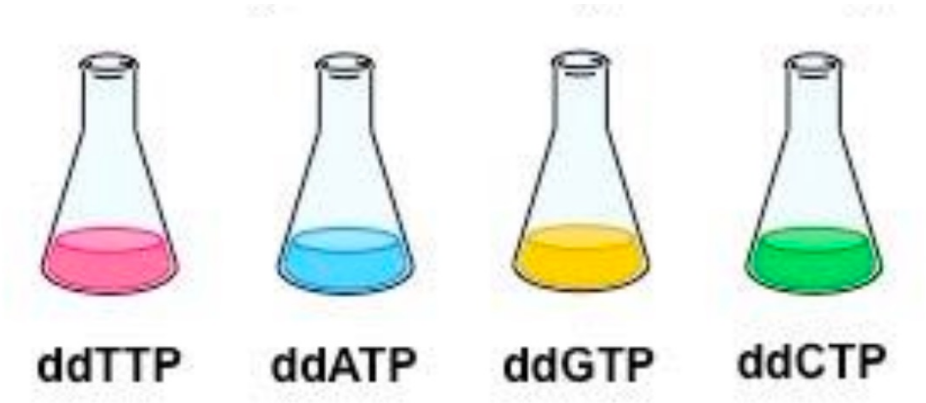
- ✓ Denaturazione
- ✓ Annealing
- ✓ Elongazione
- ✓ Terminazione (con ddNTP)

T G C A G G C A T C T G A

T G C A G G C A

| | | | | | | | | | | | |

A C G T C C G T A G A C T



Metodo Sanger

Sanger → Chain-termination method

Procedimento:

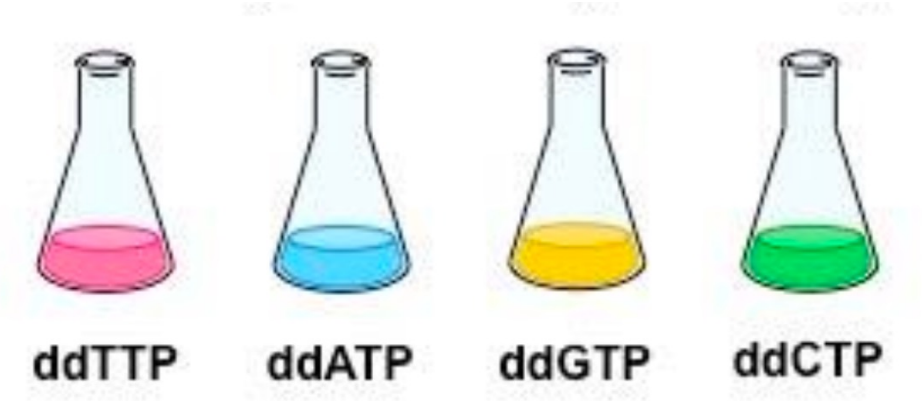
- ✓ Denaturazione
- ✓ Annealing
- ✓ Elongazione
- ✓ Terminazione (con ddNTP)

T G C A G G C A T C T G A

T G C A G G C A

| | | | | | | | | | | | |

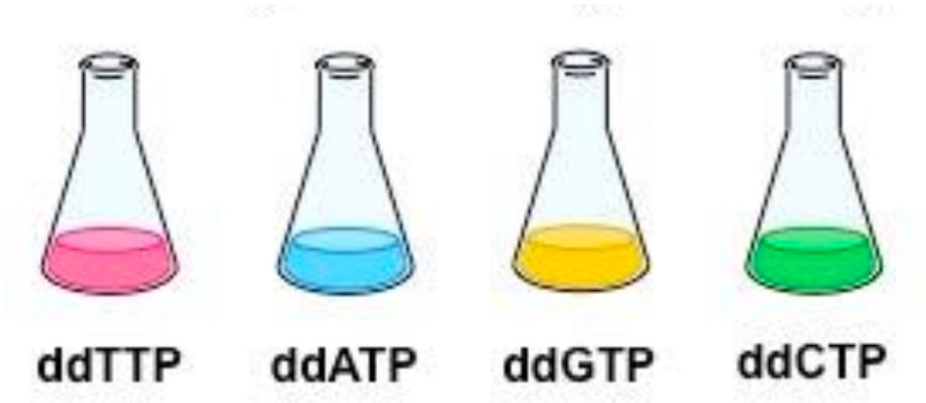
A C G T C C G T A G A C T



Metodo Sanger

Sanger → Chain-termination method

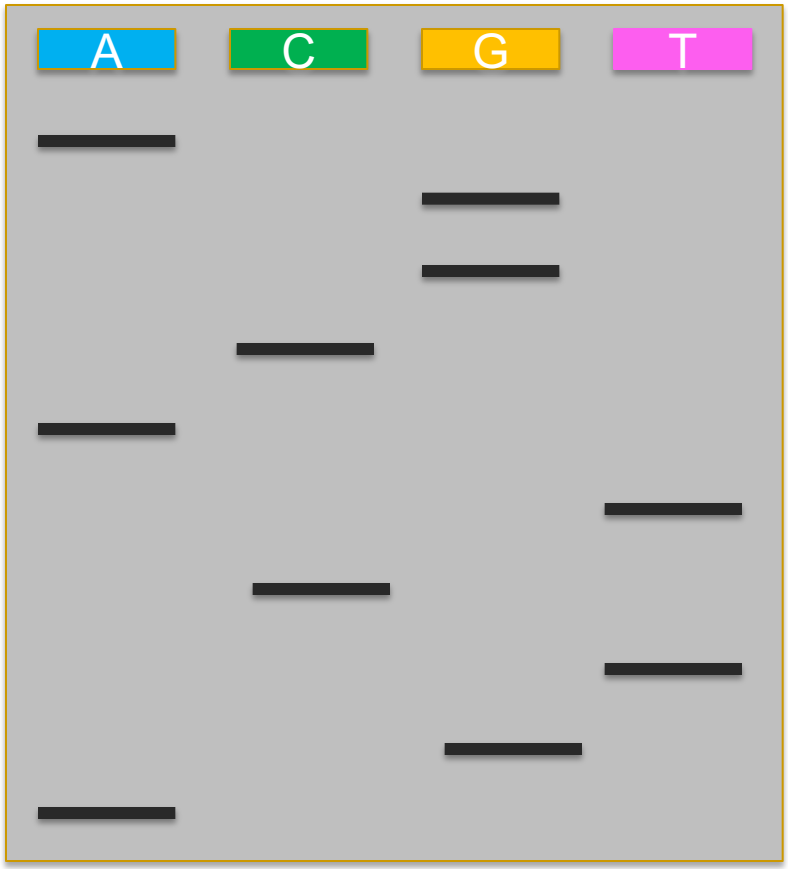
TGCA
TGCAGGCA
TGCAGGCATCTGA
TGCAG
TGCAGG
TGCAGGCATCTG
TGCAGGCATC
TGCAGGC
TGCAGGCAT
TGCAGGCATCT



Metodo Sanger

Sanger → Chain-termination method

TGCA
TGCAGGCA
TGCAGGCATCTGA
TGCAG
TGCAGG
TGCAGGCATCTG
TGCAGGCATC
TGCAGGC
TGCAGGCAT
TGCAGGCATCT

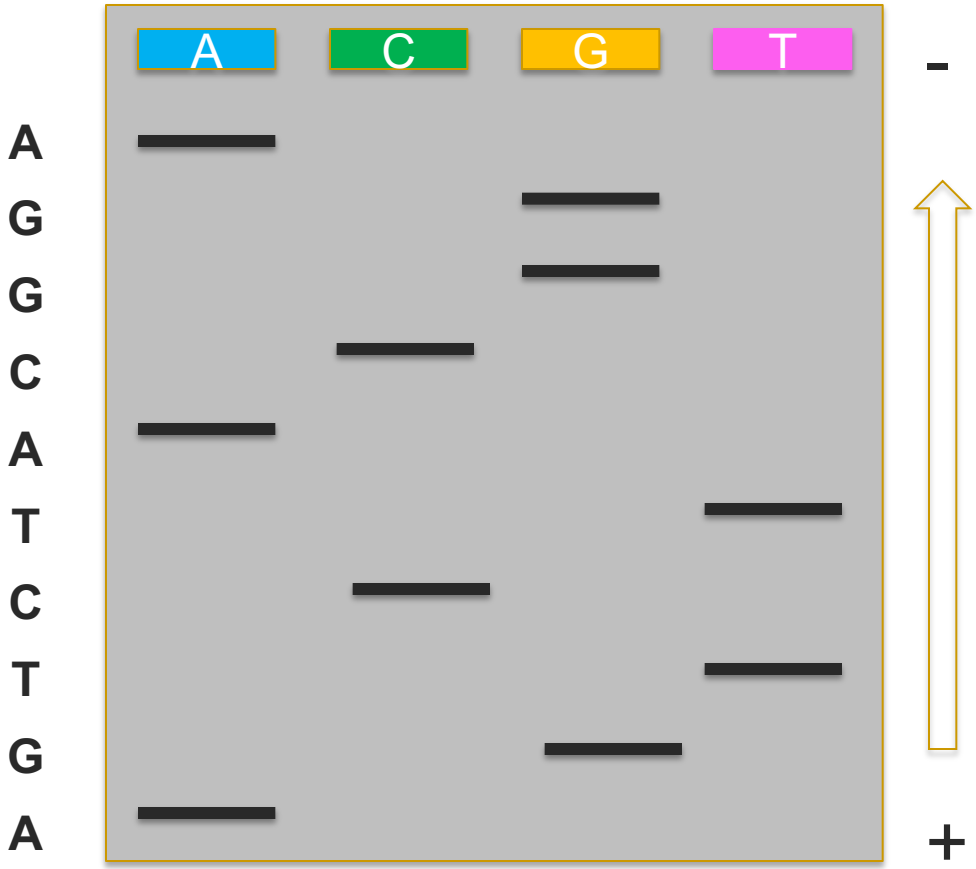


gel di poliacrilamide

Metodo Sanger

Sanger → Chain-termination method

TGCA
TGCAGGCA
TGCAGGCATCTGA
TGCAG
TGCAGG
TGCAGGCATCTG
TGCAGGCATC
TGCAGGC
TGCAGGCAT
TGCAGGCATCT



gel di poliacrilamide

Metodo Sanger

T G C A G G C A T C T G A

| | | | | | | | | | | | | |

A C G T C C G T A G A C T

Sanger → Chain-termination method

T G C A

T G C A G G C A

T G C A G G C A T C T G A

T G C A G

T G C A G G

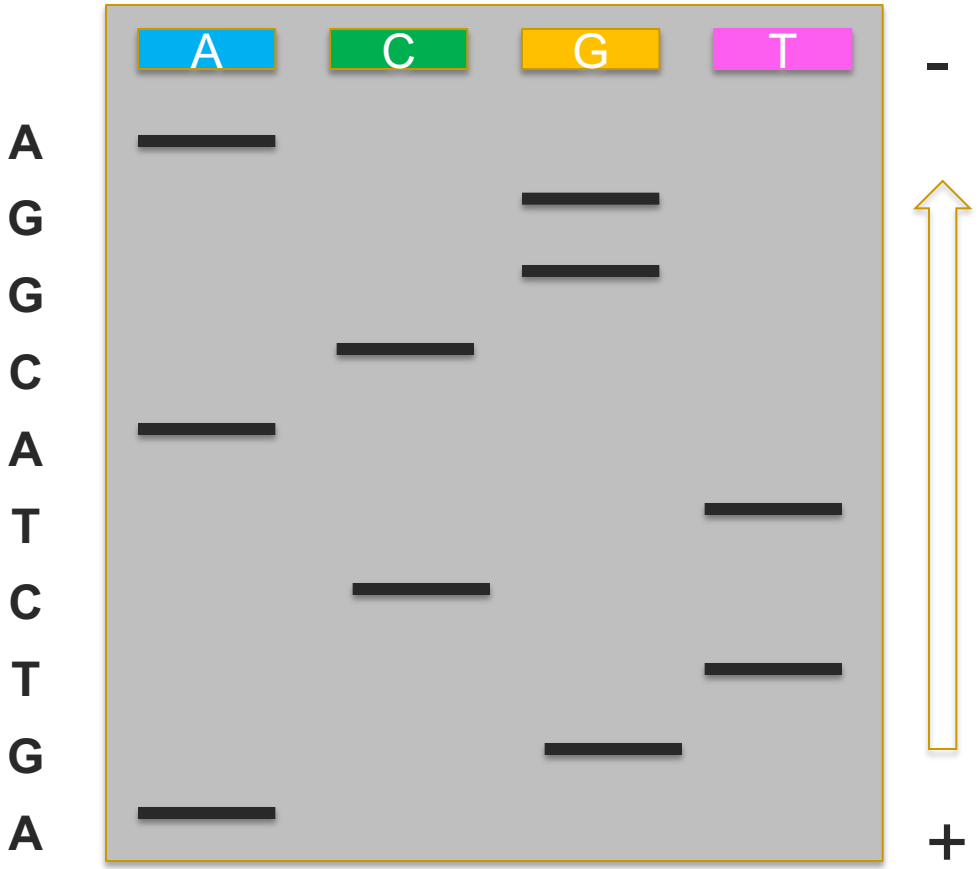
T G C A G G C A T C T G

T G C A G G C A T C

T G C A G G C

T G C A G G C A T

T G C A G G C A T C T



gel di poliacrilamide **sottoposti a RX**

Metodo Sanger

Sanger → Chain-termination method

metodo alternativo: ogni terminatore ha una sostanza fluorescente diversa

T G C A

T G C A G G C A

T G C A G G C A T C T G A

T G C A G

T G C A G G

T G C A G G C A T C T G

T G C A G G C A T C

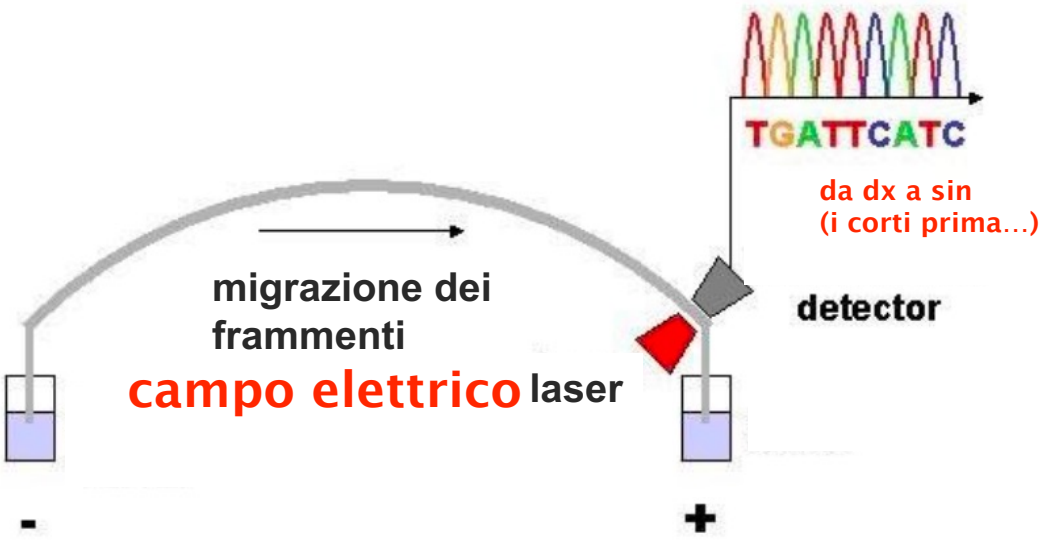
T G C A G G C

T G C A G G C A T

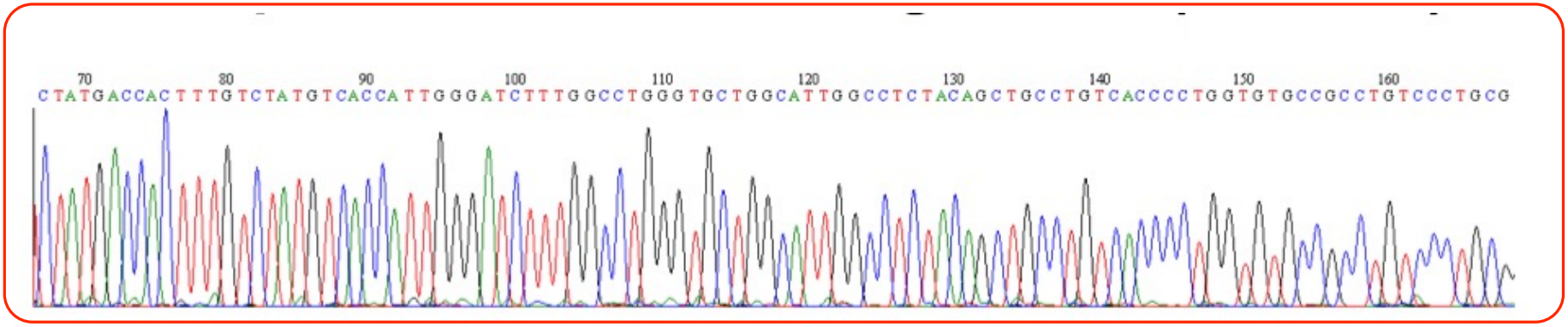
T G C A G G C A T C T

1 provetta

Elettroforesi capillare



Metodo Sanger (automatici)



cromatografia (prodotta dal sequenziatore automaticamente)

Base Caller

software che legge la cromatografia

Sequenza primaria del DNA template

Metodo Sanger

Il metodo Sanger produce un dato con:

- ✓ qualità elevata
- ✓ *coverage* bassa (5x-8x)
- ✓ lunghezza fino a 1000bp

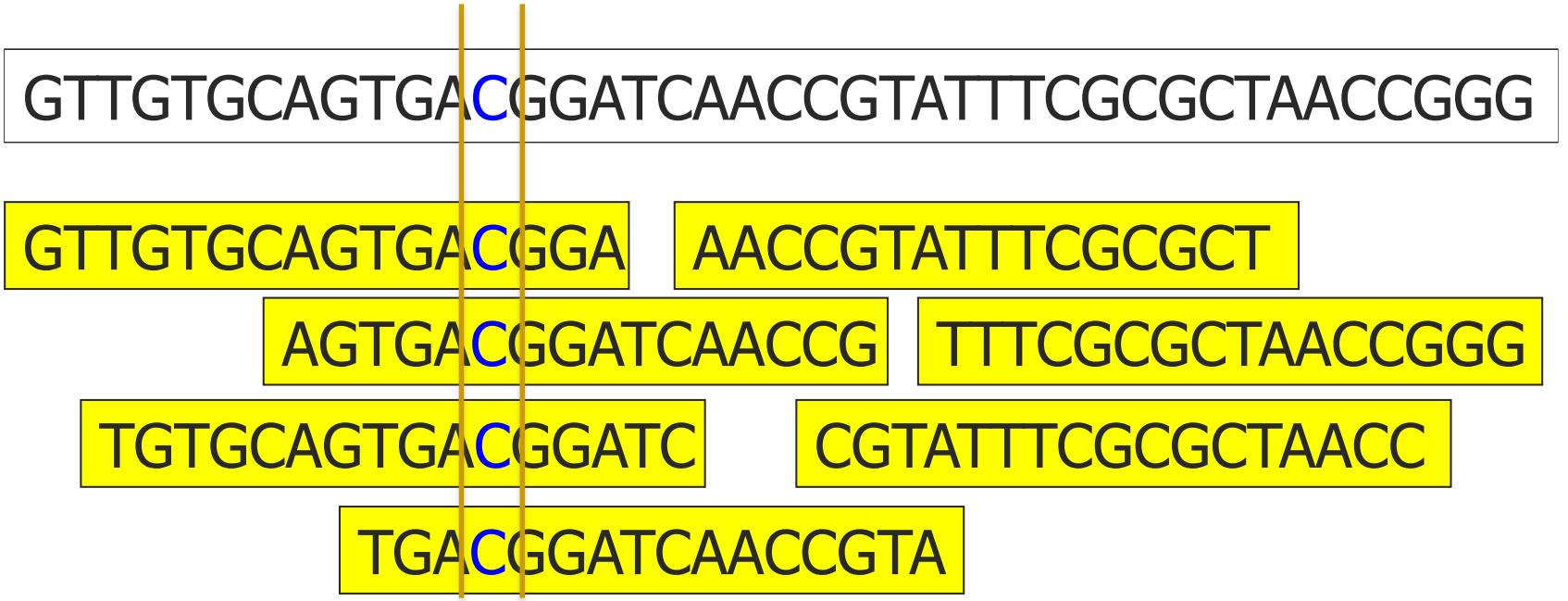
😢 Il metodo Sanger è molto costoso (3 miliardi di dollari per HGP)

1$ per nucleotide!

IL DATO DI SEQUENZIAMENTO

Coverage

numero medio di volte che una base viene analizzata, quindi corrisponde al numero di reads che mappano sulla base.



coverage = n l / L

n → #reads
l → lunghezza dei reads (fissata)
L → lunghezza della molecola

Qualità

delle read prodotte
dal sequenziamento

GTTGTGCAGTGACGGATCAACCGTATTTCGCGCTAACCGGG

GTTGTGCTGTGACGGA

AACCGTATTTCG-GCT

AGTGACGGATCAACCG

TTTCGCGCTAACCGGG

TGTGCAGTGACTGGATC

CGTATTTCGCGCTAACC

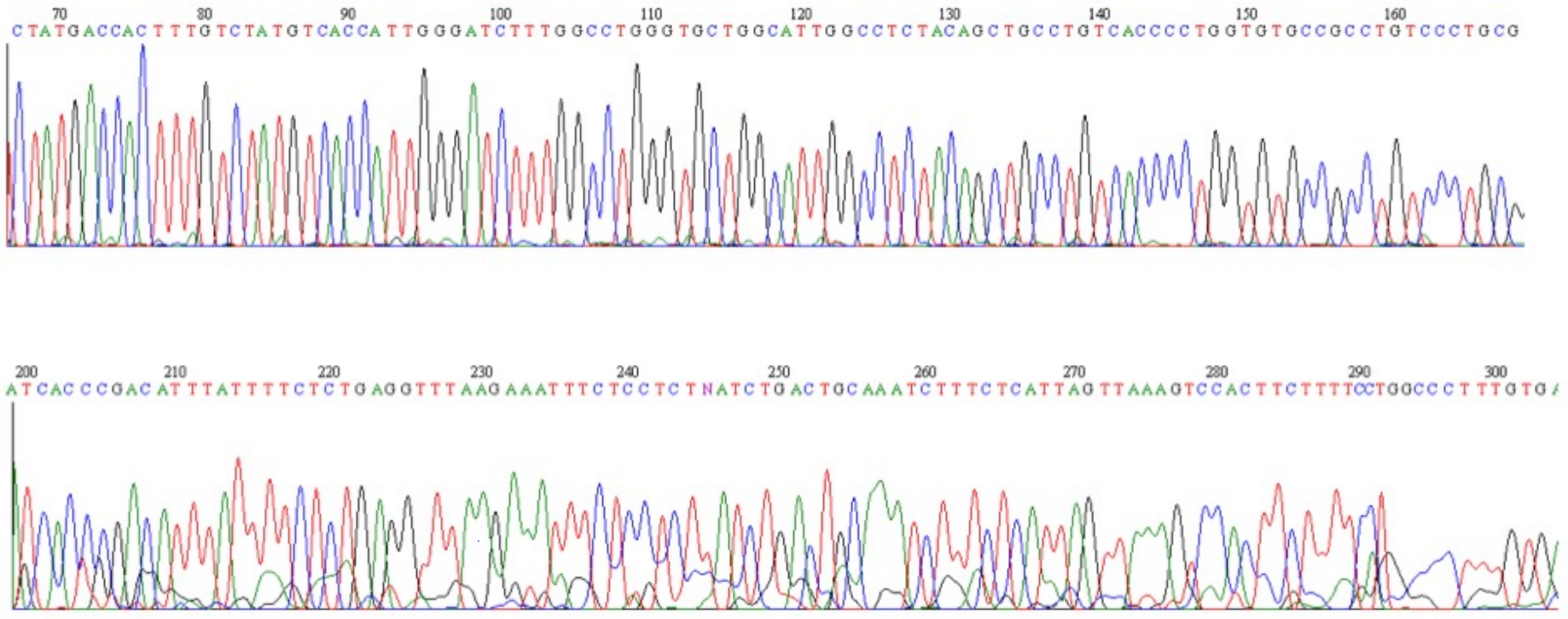
TGACGGATCAACGGTA

Questa T non c'è nel template (sostituzione)

Questa T non c'è nel template (inserita)

La C del template non c'è in questa read

Qualità



questo cromatogramma non consente di evincere chiaramente la sequenza delle basi.
Posso dare la lettera ma quanto sarà affidabile?
Il FASTQ esprime anche questa qualità

Tecnologie NGS

- ✓ Tecnologie ideate a partire dagli anni 2000
- ✓ Tecnologie altamente parallele (high-throughput)
- ✓ Produce un dato con:
  - ✓ qualità variabile
  - ✓ *coverage* elevatissima
  - ✓ lunghezza fino a 300bp
- ✓ Veloci e poco costose (1500$ per sequenziare un genoma)

milioni di frammenti (anche che coprono 33 genomi umani…)

Tecnologie NGS

Illumina (Solexa)

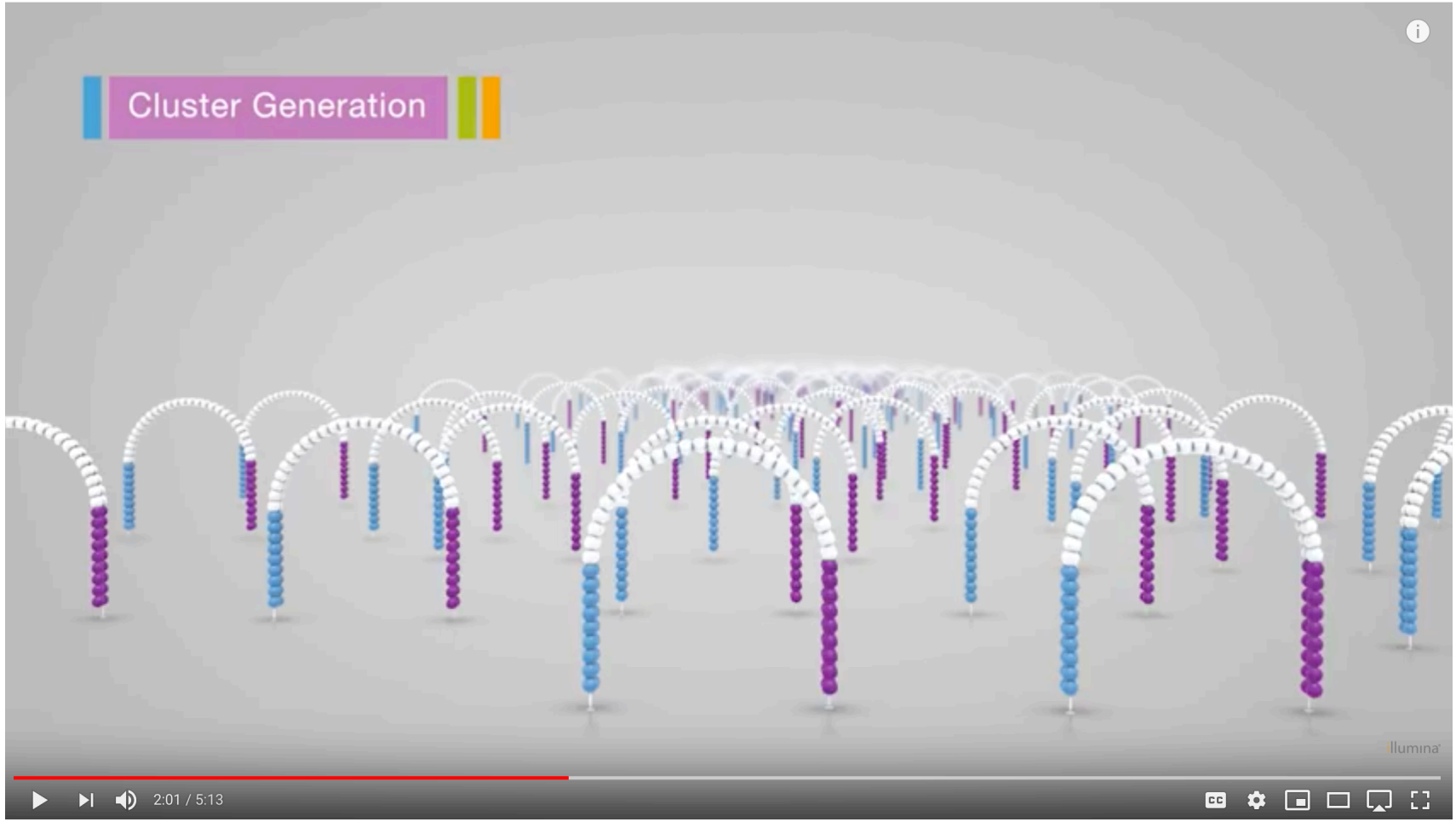
- HiSeq System
- Genome analyzer Iix
- MySeq simil Sanger

Ion Torrent - Life technologies

- Personal Genome Machine (PGM)
- Proton

altra tecnica

Come sono sequenziate le reads?



https://www.youtube.com/watch?v=fCd6B5HRaZ8

MiSeq

PGM (Personal Genome Machine)

	MiSeq	PGM
lunghezza reads (paia di basi)	300	200
throughput (giga basi G)	15	1
numero di reads per run (milioni)	25	11
Accuratezza (%)	99,90	99,00
Tempo di run (h)	65	4,5

Tecnologie NGS

Illumina (Solexa) → **coverage: 50x-80x**

- HiSeq System
- Genome analyzer Iix
- MySeq → **300bp**

Ion Torrent - Life technologies:

- Personal Genome Machine (PGM) → **200 bp**
- Proton

Tecnologie NGS

Illumina (Solexa) → **coverage: 50x-80x**

- HiSeq System
- Genome analyzer Iix
- MySeq → **25M reads/run**

Ion Torrent - Life technologies:

- Personal Genome Machine (PGM) → **11M reads/run**
- Proton

Tecnologie Next-NGS

> 1000 bp **Long Reads**

Pacific Biosciences → lunghezza: 10-15 Kb (quasi tutto un trascritto)

- PacBio RS

Oxford Nanopore Technologies → lunghezza: 5-10 Kb

- GridION System
- MinION

Tecnologie Next-NGS

> 1000 bp

Pacific Biosciences → 10-15% di errore

- PacBio RS

Oxford Nanopore Technologies → 10-30% di errore

- GridION System
- MinION

Small but mighty

Watch full video

< 450 g

Sequence anywhere, pocket-sized devices

From $1,000

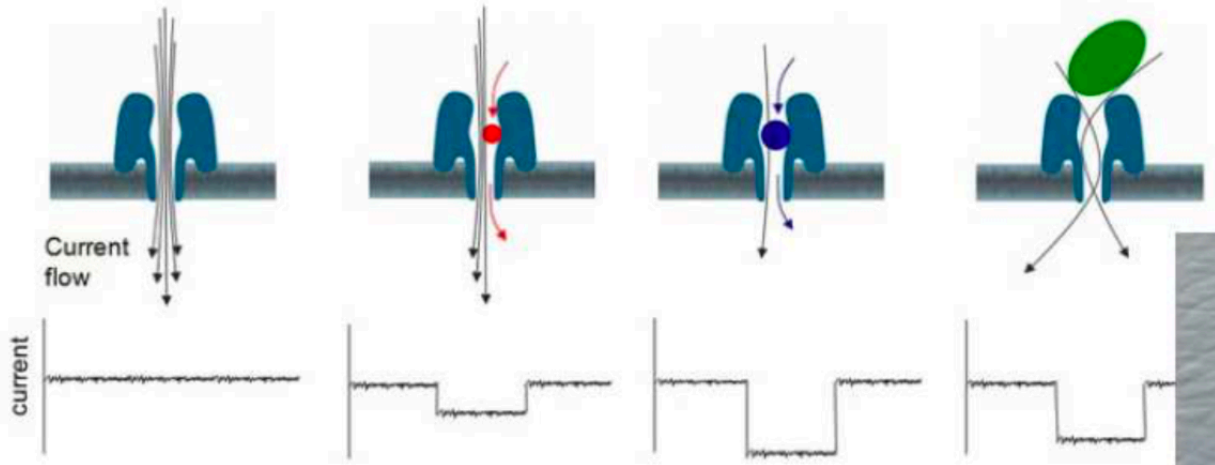
Low-cost sequencing for every lab

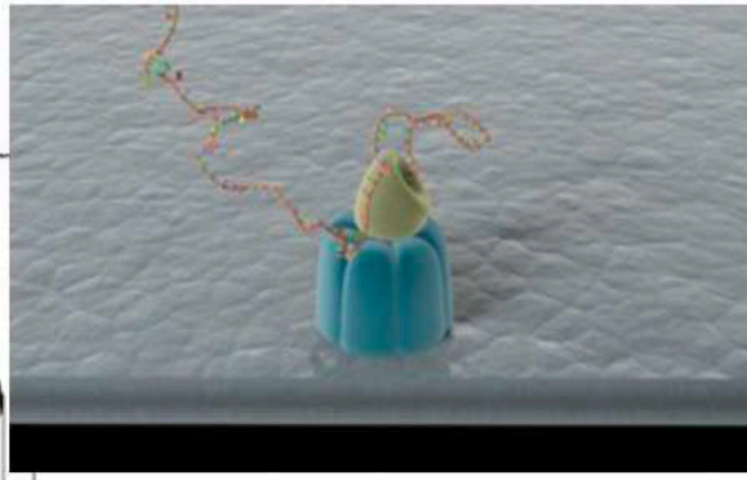
Up to 50 Gb*

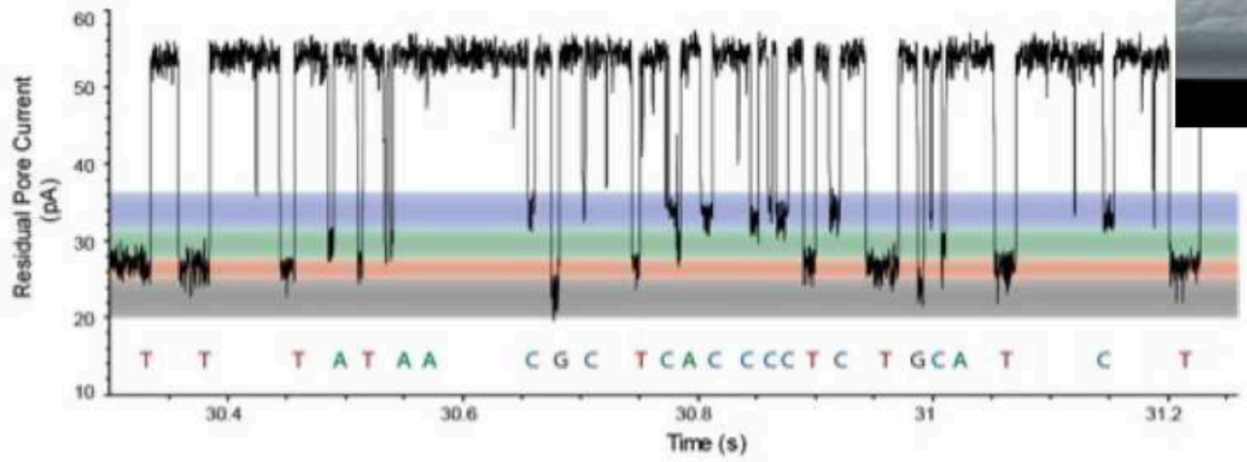
Real-time sequencing data

* Theoretical max output when system is run for 72 hours at 420 bases / second. Outputs may vary according to library type, run conditions, etc.

Oxford Nanopore



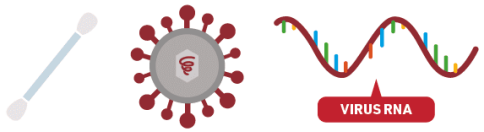




HOW DO THE TESTS FOR CORONAVIRUS WORK?

HOW CURRENT TESTS WORK

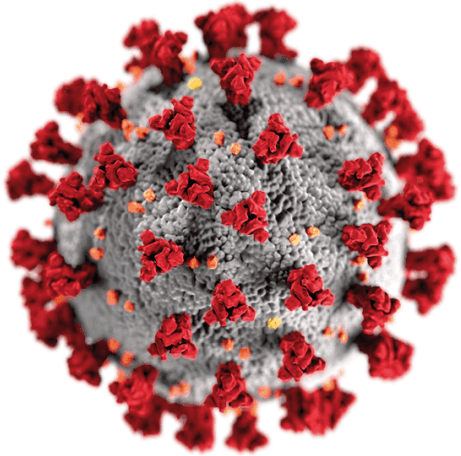
1. A swab is taken of the inside of a patient's nose or the back of their throat. This sample is then sent to a lab to test.



2. The RNA of the virus is extracted and purified. An enzyme, reverse transcriptase, converts the RNA to DNA.

3. The DNA is mixed with primers, sections of DNA designed to bind to characteristic parts of the virus DNA. Repeatedly heating then cooling DNA with these primers and a DNA-building enzyme makes millions of copies of virus DNA.

4. Fluorescent dye molecules bind to the virus DNA as it is copied. Binding makes them give off more light, which is used to confirm the presence of the virus in the sample.



ISSUES WITH TESTING

REAGENT ISSUES

High demand and issues with reagents have delayed testing in some countries.

TIME-CONSUMING

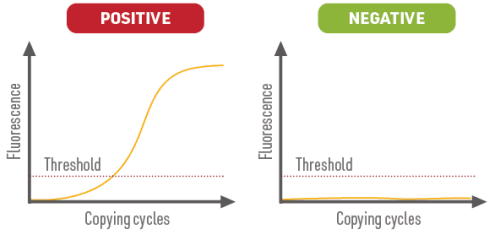
It takes a few hours to get results from the test, limiting how many tests can be done.

FALSE POSITIVES AND NEGATIVES

In some cases sample degradation or contamination can affect the results.

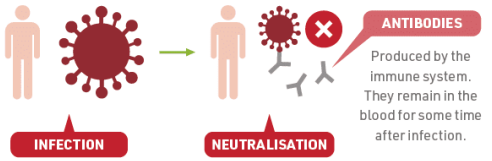
POSITIVE AND NEGATIVE TESTS

The fluorescence increases as more copies of the virus DNA are produced. If it crosses a certain threshold, the test is positive. If the virus isn't present, no DNA copies are made and the threshold isn't reached. In this case, the test is negative.

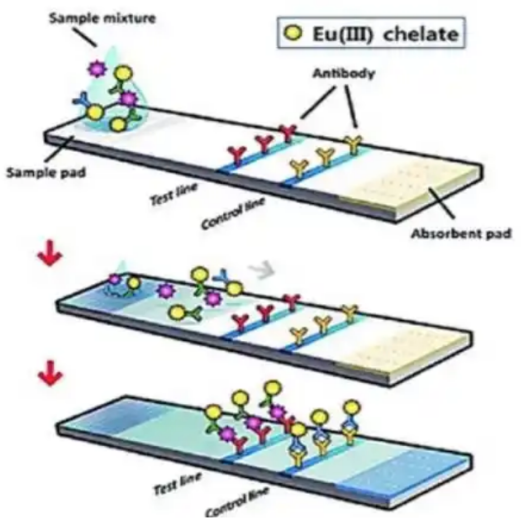


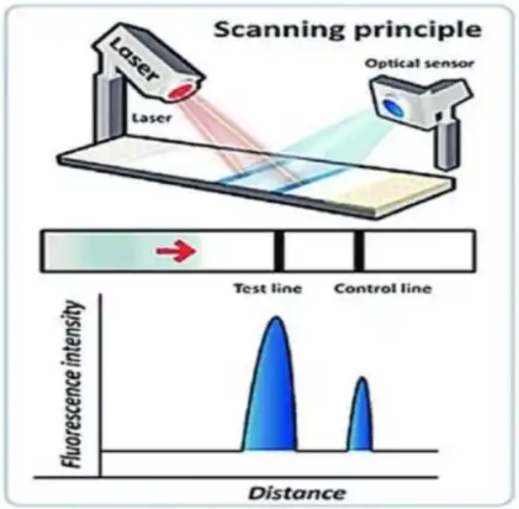
FUTURE TESTS

The current tests are good for diagnosing an infection – but they can't tell us if someone has had it and then recovered. Tests that look for antibodies against the virus can do this.



Tests that look for proteins on the surface of the virus are also in development. These tests are faster, but less accurate.





Il test è basato sulla **tecnologia di immunocromatografia a fluorescenza**. Ogni test è costituito da una banda rivelatrice dell'anticorpo monoclonale anti SARS-CoV-2 (banda T) e una seconda banda rivelatrice di un anticorpo di controllo (banda C). Il campione estratto viene caricato nell'apposito pozzetto dove reagirà con l'anticorpo marcato a fluorescenza per formare un complesso, la miscela quindi migra attraverso la membrana per azione capillare e interagisce con l'anticorpo monoclonale rivestito anti-SARS-CoV-2 sulla linea di rilevamento. Se il campione contiene l'antigene SARS-CoV-2, il complesso a fluorescenza che fluisce lungo la membrana viene catturato dall'anticorpo SARS-CoV-2 e l'intensità del segnale a fluorescenza riflette il quantitativo di antigene SARSCoV-2 catturato ed è individuato da uno strumento analizzatore che dà esito positivo. Altrimenti il risultato sarà negativo.

frontiers in Virology

ORIGINAL RESEARCH
published: 28 February 2022
doi: 10.3389/fviro.2022.814114

BioGem (Ariano Irpino)

Genomic Characterization of the Emerging SARS-CoV-2 Lineage in Two Districts of Campania (Italy) Using Next-Generation Sequencing

OPEN ACCESS

Edited by:
Shuofeng Yuan,
The University of Hong Kong, Hong Kong SAR, China

Marianna Scrima[1,2†], *Alessia Maria Cossu*[1,2,3†], *Egildo Luca D'Andrea*[1,4], *Marco Bocchetti*[2,3], *Ylenia Abruzzese*[1], *Clara Iannarone*[1], *Cinzia Miarelli*[1], *Piera Grisolia*[2], *Federica Melisi*[2,3], *Lucia Genua*[5,6], *Felice Di Perna*[7], *Paolo Maggi*[8,9], *Giovanbattista Capasso*[10], *Teresa Maria Rosaria Noviello*[11,12], *Michele Ceccarelli*[11,12], *Alessandra Fucci*[1,4] *and Michele Caraglia*[1,2,3*]

Tecnologie NGS e Next-NGS

Conseguenze importanti:

✓ esplosione dei dati → Sequence Read Archive (SRA)
✓ nuovi problemi computazionali:

- quantificazione dei trascritti
- predizione di nuovi eventi di splicing
- filogenesi tumorale
- aplotipizzazione
- variant calling
- rappresentazione di più genomi → pan-genoma

→ nuovi algoritmi e tools

La qualità del dato NGS

- ✓ *Reads* relativamente corti → la conoscenza della qualità fondamentale
- ✓ La qualiltà del read deve essere tenuta in considerazione
- ✓ Indice di qualità per ognuna delle basi di un reads NGS
- ✓ Sequenza primaria del *read* + sequenza delle qualità delle basi del *read*

Phred Quality Score

Indice di qualità sviluppato all'interno del software Phred

- ✓ *Base calling* software
- ✓ Phil Green's Lab della Washington University (primi anni '90)

Phred Quality Score

Indice di qualità sviluppato all'interno del software Phred

- ✓ *Base calling* software
- ✓ Phil Green's Lab della Washington University (primi anni '90)

Associa a ogni base del read
la probabilità $\mathbf{p_e}$ ($0 \leq p_e \leq 1$) che la base sia errata

Phred Quality Score

Indice di qualità sviluppato all'interno del software Phred

- ✓ *Base calling* software
- ✓ Phil Green's Lab della Washington University (primi anni '90)

Associa a ogni base del read
la probabilità $\mathbf{p_e}$ ($0 \leq p_e \leq 1$) che la base sia errata

Phred Quality Score → $q = -10 \log_{10} p_e$

Phred Quality Score

Phred q di una base che ha una probabilità dell'1% di essere errata

$\mathbf{p_e} = 0{,}01$

$$q = -10 \log_{10} 0{,}01 = 20$$

Phred Quality Score

Phred q di una base che ha una probabilità dell'100% di essere errata

$\mathbf{p_e} = 1$

$$q = -10 \log_{10} 1 = 0$$

Phred Quality Score

Phred q di una base che ha una probabilità dell'0% di essere errata

$\mathbf{p_e} = 0$

$$q = -10 \log_{10} 0 = +\infty$$

Phred Quality Score

Phred q di una base che ha una probabilità dell'0% di essere errata

$\mathbf{p_e} = 0$

$$q = -10 \log_{10} 0 = +\infty$$

NB: a fini pratici una base con q pari ad almeno 50 è considerata corretta

DALLA LEZIONE 0

FORMATO FASTA

Formato standard per sequenze nucleotidiche

- ✓ plain text
- ✓ sequenza primaria + informazioni addizionali
- ✓ nato come formato di input del software FASTA
- ✓ estensione → `fa` oppure `fasta`

cromosoma

id del genoma di riferimento e la sua localizzazione

catena diretta

```
>X dna:chromosome chromosome:GRCh38:X:154428200:154438516:1
GGATTGGCCGGGTGCGGCGATGCGCGCCTGTAGTCCCACTTTTCCGGAGGCTGAGGCGGG
AGGATCGCTTGAGCCCGGGGACTCGAGACCAGCCTGGCCAACATAGCTAGACCCCGTCTC
TAAAGAAAAAAAAAAGACAA                                    TAAAAATAAA
AATAAAAACGGACGGGCGAA                                    CCTCCCTGTC
CCTCTTGTCTCGTCCTAGTC                                    TCGGAGCGTG
AGCGCCTGGAGACACGTACA                                    GGCATGCACT
TGGTCCAATTACCTGCGGCC                                    AGGTGCGAGG
CGGGGCTCGGGCGGGGGCAA                                    CTGAGGCAGT
GGAGGCTGAGGCTATGATGG                                    GGCCGCGGTG
CGCCCAGGCGCTCTGGCGCATGCCGTGGCTGCCGGTGTTTTTGTCGTTGGCGGCGGCGGC
GGCGGCGGCAGCGGCGGAGCAGCAGGTCCCGCTGGTGCTGTGGTCGAGTGACCGGTGAGC
GGGCCGGGGTGGGATGCGCTGTGGCGGCTGAGGCGCCCTCGCCCGACTCCGGCGCTGTCC
TAGGCGAGGGGTGGTGAGGCCCGGAGGTGGACTGTTCCTTGCTCGGGGGCTCGCAGCGAA
TCTGCCGGCGACAGAGCTCCAGTCCACATGCGCCCCCGTCTGACAGCACCTCTTCTGTGC
CCTGCCAGGGACTTGTGGGCTCCTGCGGCCGACACTCATGAAGGCCACATCACCAGCGAC
TTGCAGCTCTCTACCTACTTAGATCCCGCCCTGGAGCTGGGTCCCAGGAATGTGCTGCTG
TTCCTGCAGGACAAGGTGCGCCCGCCCCAGCCCACTCTCCCCCGGTCATCGGGAGGCAGC
[…]
```

FASTA header:

- ✓ primo simbolo → **>**
- ✓ unica riga

DALLA LEZIONE 0

FORMATO FASTA

Formato standard per sequenze nucleotidiche

- ✓ plain text
- ✓ sequenza primaria + informazioni addizionali
- ✓ nato come formato di input del software FASTA
- ✓ estensione → `fa` oppure `fasta`

```
>X dna:chromosome chromosome:GRCh38:X:154428200:154438516:1
GGATTGGCCGGGTGCGGCGATGCGCGCCTGTAGTCCCACTTTTCCGGAGGCTGAGGCGGG
AGGATCGCTTGAGCCCGGGGACTCGAGACCAGCCTGGCCAACATAGCTAGACCCCGTCTC
TAAAGAAAAAAAAAAGACAAAAGAAGAGAACAACAACAAAAAAACCAAAATAAAAATAAA
AATAAAAACGGACGGGCGAACGCGTCGGGTGGGCGTAGGGTTCCGCGAAGCCTCCCTGTC
CCTCTTGTCTCGTCCTAGTCGGTCTAGCTGGGCCCCATCCCCTTTCCCACTCGGAGCGTG
AGCGCCTGGAGACACGTACAGCCAACCAGTGAGAAGGAGTGGCCGCGAGTGGCATGCACT
TGGTCCAATTACCTGCGGCCCTGCCGGTCGGCCCGCGCTGGGGCCAATGGAGGTGCGAGG
CGGGGCTCGGGCGGGGGCAACGGTCACCTGATCTGCGGCTGTCGAGGCCGCTGAGGCAGT
GGAGGCTGAGGCTATGATGGCGGCCATGGCGACGGCTCGAGTGCGGATGGGGCCGCGGTG
CGCCCAGGCGCTCTGGCGCATGCCGTGGCTGCCGGTGTTTTTGTCGTTGGCGGCGGCGGC
GGCGGCGGCAGCGGCGGAGCAGCAGGTCCCGCTGGTGCTGTGGTCGAGTGACCGGTGAGC
GGGCCGGGGTGGGATGCGCTGTGGCGGCTGAGGCGCCCTCGCCCGACTCCGGCGCTGTCC
TAGGCGAGGGGTGGTGAGGCCCGGAGGTGGACTGTTCCTTGCTCGGGGGCTCGCAGCGAA
TCTGCCGGCGACAGAGCTCCAGTCCACATGCGCCCCCGTCTGACAGCACCTCTTCTGTGC
CCTGCCAGGGACTTGTGGGCTCCTGCGGCCGACACTCATGAAGGCCACATCACCAGCGAC
TTGCAGCTCTCTACCTA
TTCCTGCAGGACAAGGT
[…]
```

Sequenza separata in righe di 60/80bp simboli

Il formato FASTQ

- ✓ formato di puro testo
- ✓ disegnato da Wellcome Trust Sanger Institute per associare ad una sequenza la qualità di ogni sua singola base
- ✓ formato standard di output di strumenti di sequenziamento NGS
- ✓ estensione: `*.fq` oppure `*.fastq`

FASTA si usa per le genomiche di riferimento: si è lavorato anni per arrivare ad una versione definitiva di riferimento.

Il formato FASTQ (esempio)

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
aaaa`aa`aa`]__`aa`_U[_a`^\\UTWZ`X^QX
```

Il formato FASTQ (esempio)

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
aaaa`aa`aa`]__`aa`_U[_a`^\\UTWZ`X^QX
```

Header del *read* (simbolo iniziale: @)

Header di Illumina

Il formato FASTQ (esempio)

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
aaaa`aa`aa`]__`aa`_U[_a`^\\UTWZ`X^QX
```

Sequenza di basi del *read*

Il formato FASTQ (esempio)

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
aaaa`aa`aa`]__`aa`_U[_a`^\\UTWZ`X^QX
```

Header della sequenza dei *phred values* (simbolo iniziale: +)

Il formato FASTQ (esempio)

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
aaaa`aa`aa`]__`aa`_U[_a`^\\UTWZ`X^QX
```

Header della sequenza dei *phred values* (simbolo iniziale: +)

NB: l'ID del *read* è opzionale

Il formato FASTQ (esempio)

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
aaaa`aa`aa`]__`aa`_U[_a`^\\UTWZ`X^QX
```

Sequenza dei Phred scores codificati in caratteri

Il formato FASTQ (esempio)

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
aaaa`aa`aa`]__`aa`_U[_a`^\\UTWZ`X^QX
```

Sequenza dei Phred scores codificati in caratteri
Il carattere i-esimo codifica il Phred q della i-esima base del *read*

Conversione di q in carattere

Ogni Phred q deve essere convertito in un carattere stampabile

Conversione di q in carattere

Ogni Phred q deve essere convertito in un carattere stampabile

Ogni sequenziatore NGS ha una propria funzione di conversione da Phred q a carattere c:

$$c = f(q)$$

Conversione di q in carattere

Una tipica conversione è la seguente:

c = ASCII(min(q, 93) + 33)

cioé:

1. se q è maggiore di 93 ➔ q = 93
2. si aggiunge a q il valore 33
3. si converte in carattere ASCII l'intero ottenuto

Conversione di q in carattere

Una tipica conversione è la seguente:

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

Esempio

il carattere che codifica il Phred q = 31

Conversione di q in carattere

Una tipica conversione è la seguente:

$$c = ASCII(min(q, 93) + 33)$$

Esempio

il carattere che codifica il Phred q = 31

$$c = ASCII(min(31, 93) + 33) = ASCII(64) \rightarrow \text{‘@’}$$

Conversione di q in carattere

Una tipica conversione è la seguente:

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

Esempio

il Phred q codificato dal carattere ‘a’

Conversione di q in carattere

Una tipica conversione è la seguente:

c = ASCII(min(q, 93) + 33)

Esempio

il Phred q codificato dal carattere ‘a’

‘a’ = ASCII(97)
q = 97 - 33 = 64

Conversione di q in carattere

Una tipica conversione è la seguente:

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

Esempio

la probabilità $\mathbf{p_c}$ che un base con qualità codificata dal carattere ‘]’ sia corretta

Conversione di q in carattere

Una tipica conversione è la seguente:

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

Esempio

la probabilità $\mathbf{p_c}$ che un base con qualità codificata dal carattere ‘]’ sia corretta

‘]’ = ASCII(93)

Conversione di q in carattere

Una tipica conversione è la seguente:

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

Esempio

la probabilità $\mathbf{p_c}$ che un base con qualità codificata dal carattere ‘]’ sia corretta

‘]’ = ASCII(93)
q = 93-33 = 60

Conversione di q in carattere

Una tipica conversione è la seguente:

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

Esempio

la probabilità $\mathbf{p_c}$ che un base con qualità codificata dal carattere ‘]’ sia corretta

‘]’ = ASCII(93)
q = 93-33 = 60
$\mathbf{p_e} = 10^{(-60/10)} = 0{,}000001$

Conversione di q in carattere

Una tipica conversione è la seguente:

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

Esempio

la probabilità $\mathbf{p_c}$ che un base con qualità codificata dal carattere ‘]’ sia corretta

‘]’ = ASCII(93)
q = 93-33 = 60
$\mathbf{p_e} = 10^{(-60/10)} = 0{,}000001$
$\mathbf{p_c} = 1 - \mathbf{p_e} = 0{,}999999$

Pre-processing dei reads

I reads NGS prima di essere processati devono essere sottoposti a pre-processing:

- ✓ correzione degli errori

difficile

- ✓ eliminazione dei reads di bassa qualità
- ✓ *trimming* dei reads

pulire le parti pessime della read perchè poco affidabili…

Trimming del read

La conoscenza della qualità di ogni singola base di un *read* consente di effettuare l'operazione di trimming.

1. Si fissa una soglia minima di qualità q*
2. Si individua la più lunga sottostringa del *read* tale che ognuna delle sue basi abbia una qualità almeno pari a q*
3. Si sostituisce al *read* tale sottostringa, se questa è sufficientemente lunga (altrimenti eliminare il *read*)

Trimming del read

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
U[aaaa`aa`aa`]__`aa`__a`^U\\TWZ`X^QX
```

ESEMPIO: q* = 58.

Trimming del read

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
U[aaaa`aa`aa`]__`aa`__a`^U\\TWZ`X^QX
```

Si considerano i caratteri ASCII che codificano una qualità inferiore a 58

Trimming del read

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
U[aaaa`aa`aa`]__`aa`__a`^U\\TWZ`X^QX
```

Si considerano i **caratteri** ASCII che codificano una qualità inferiore a 58

Trimming del read

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
U[aaaa`aa`aa`]__`aa`__a`^U\\TWZ`X^QX
```

Si considerano i **caratteri** ASCII che codificano una qualità inferiore a 58.
La più lunga **sottostringa** composta di soli caratteri che codificano una qualità almeno pari a 58 e:

[aaaa`aa`aa`]__`aa`__a`^

Trimming del read

```
@HWUSI-EAS522:8:5:662:692#0/1
TATGGAGGCCCAACTTCTTGTATTCACAGGTTCTGC
+HWUSI-EAS522:8:5:662:692#0/1
U[aaaa`aa`aa`]__`aa`__a`^U\\TWZ`X^QX
```

Si considerano i **caratteri** ASCII che codificano una qualità inferiore a 58.
La più lunga **sottostringa** composta di soli caratteri che codificano una qualità almeno pari a 58 e:

[aaaa`aa`aa`]__`aa`__a`^

Trimming del read

```
@HWUSI-EAS522:8:5:662:692#0/1
 ATGGAGGCCCAACTTCTTGTATTC
+HWUSI-EAS522:8:5:662:692#0/1
 [aaaa`aa`aa`]__`aa`__a`^
```

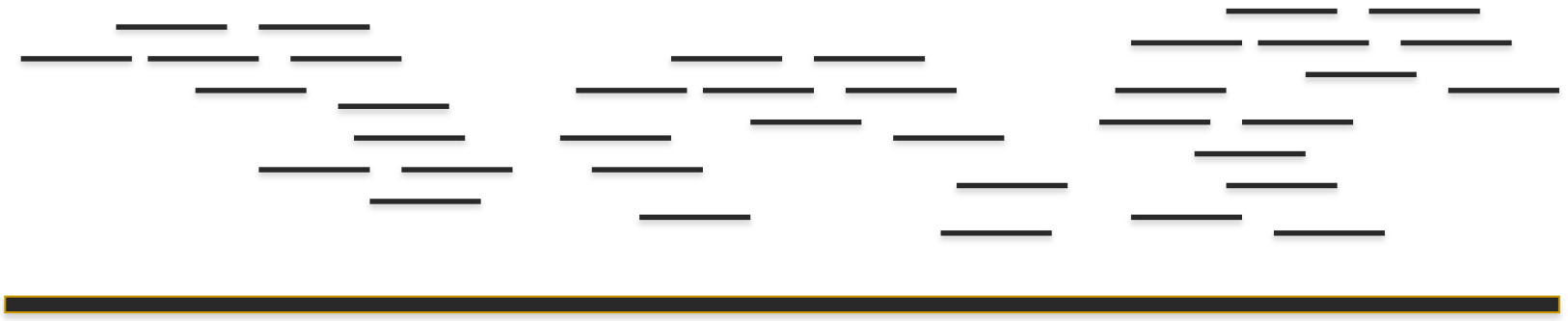
Si considerano i **caratteri** ASCII che codificano una qualità inferiore a 58.
La più lunga **sottostringa** composta di soli caratteri che codificano una qualità almeno pari a 58 e:

[aaaa`aa`aa`]__`aa`__a`^

Sequenziamento (DNA)

SEQUENZIARE (DNA): determinare la sequenza primaria di una molecola di DNA

1. Shotgun sequencing
2. Fragment assembly



DNA

Complicazioni pratiche del sequenziamento

1. Le reads hanno **copertura imperfetta** del genoma da cui derivano– ci sono molte regioni che non sono coperte da nessun read.

2. I sequenziatori sono **error-prone**.

3. Il DNA è **double-stranded**.

Faremo queste assunzioni

1. Le reads hanno **copertura perfetta** del genoma da cui derivano– ogni possibile posizione è stata sequenziata.

2. I sequenziatori sono **error-free**.

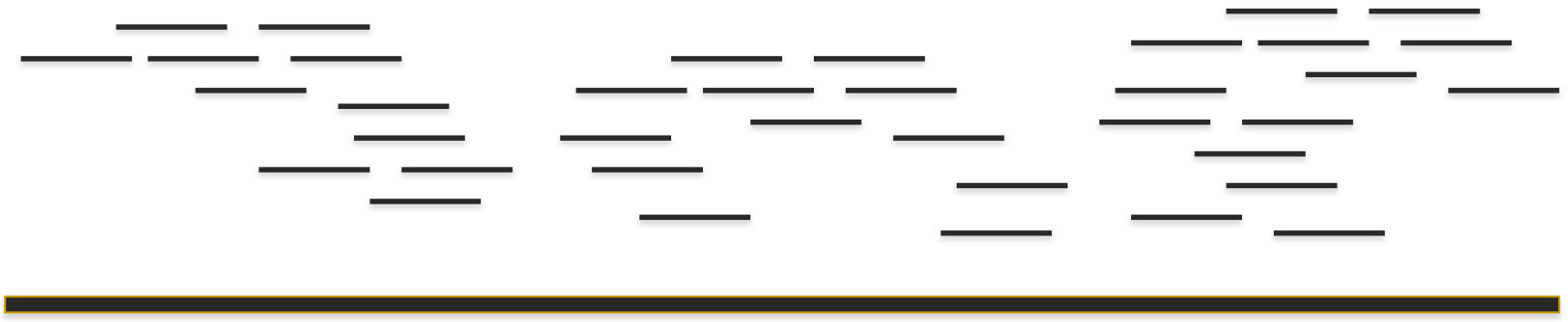
3. Il DNA è **single-stranded**.

Fragment Assembly

INPUT: un set R di *reads* (ottenuti dal sequenziamento di un DNA)

OUTPUT: sequenza primaria della molecola originaria

(in realtà più di una…)



GenBank “creato” così… https://www.ncbi.nlm.nih.gov/assembly/basics/

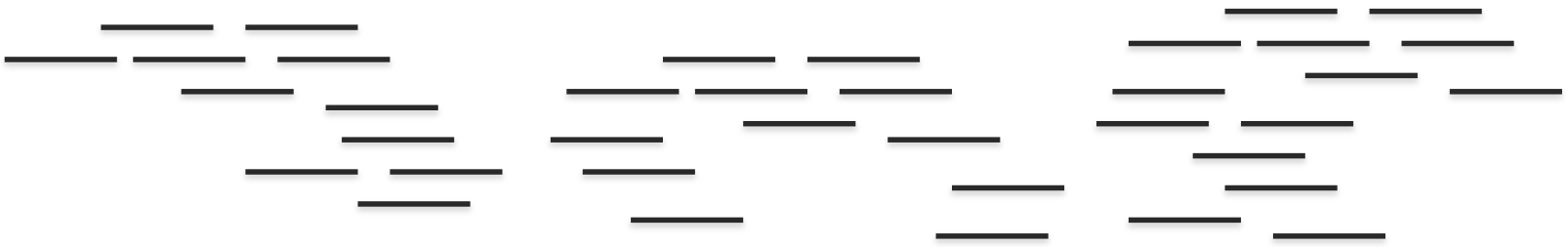
Fragment Assembly

Due tipi di assemblaggio:

ho solo le reads

reads+seq. di riferimento

1. ***De-novo* assembly**
2. *Reference-based* assembly (*resequencing*)

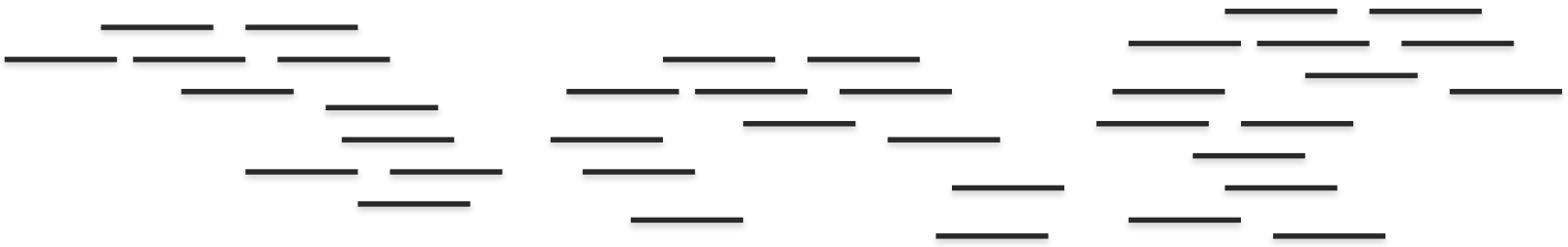


DNA

Fragment Assembly

De-novo fragment assembly:

1. Determinazione dei *contigs*
2. *Scaffolding*



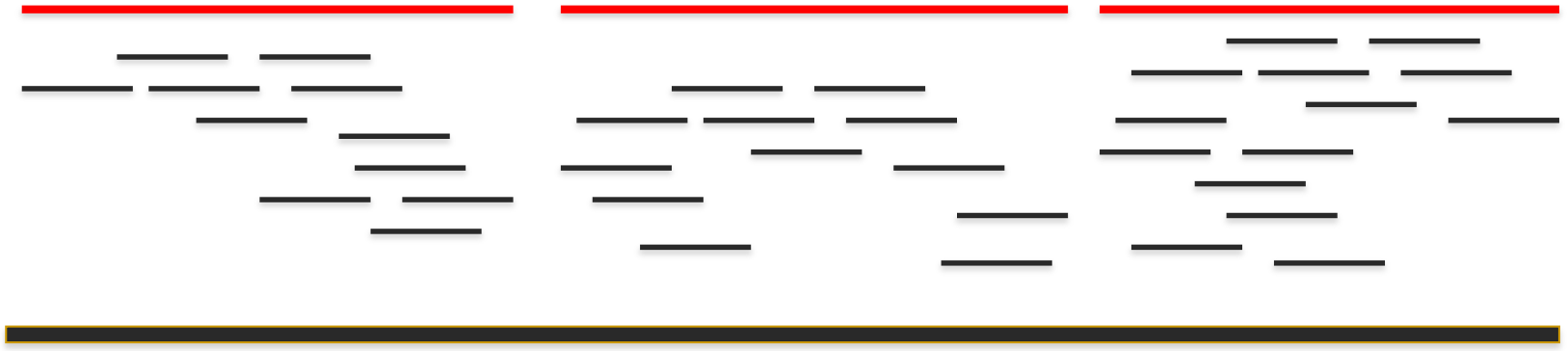
DNA

Fragment Assembly

De-novo fragment assembly:

assemblo in pezzi quanto più lunghi è possibile…

1. Determinazione dei *contigs*
2. *Scaffolding*

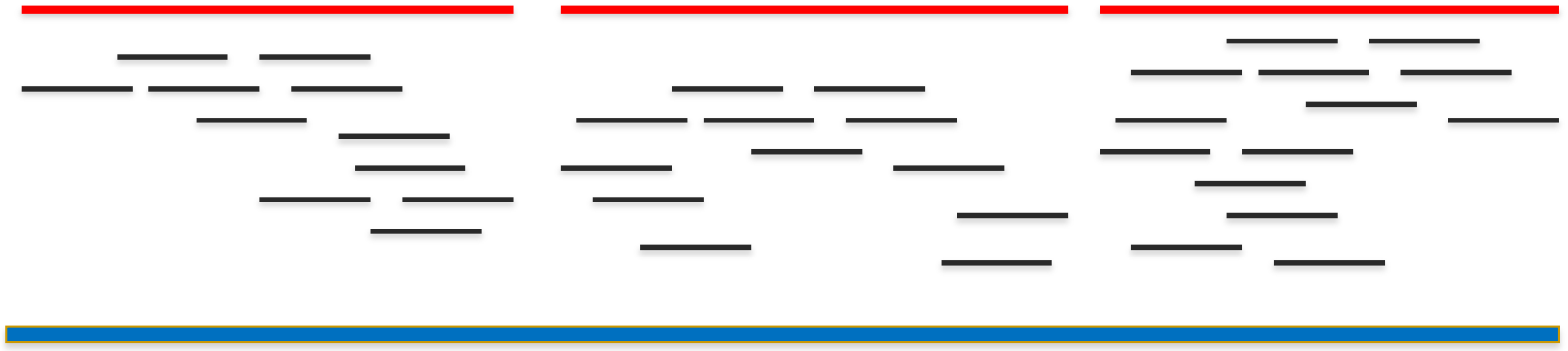


DNA

Fragment Assembly

De-novo fragment assembly:

1. Determinazione dei *contigs*
2. *Scaffolding*



DNA

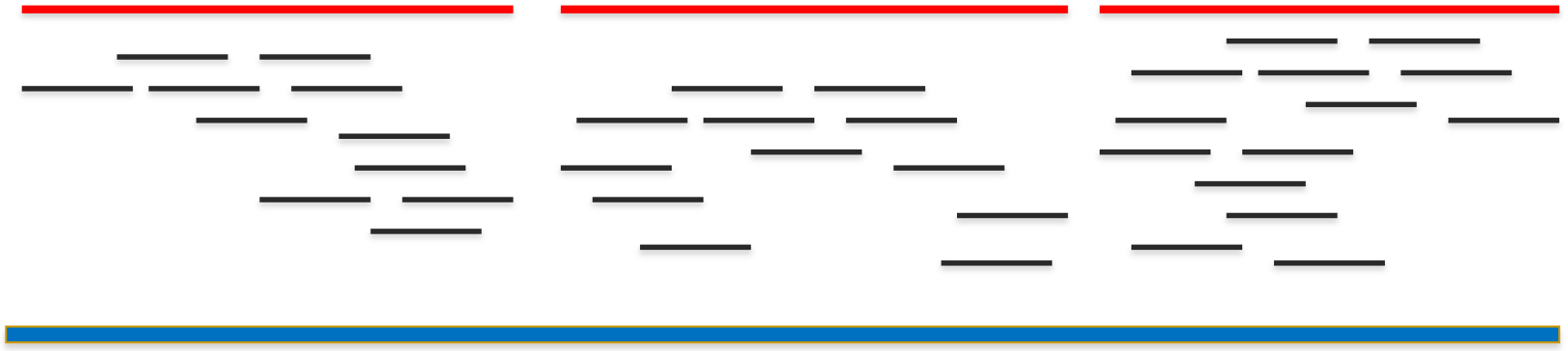
Fragment Assembly

De-novo fragment assembly:

1. Determinazione dei *contigs*
2. Scaffolding

Compito difficile:
- errori
- ripetizioni sul genoma

uso overlap "non reale"



DNA

Fragment Assembly

Determinazione dei *contigs*:

uso overlap per assemblare

- ✓ Overlap Layout Consensus (OLC) → Overlap
- ✓ Grafo di de Bruijn (dBG) → k-mer

scompongo le reads per assemblare

Entrambi gli approcci sono *graph-based*

Fragment Assembly

Approccio OLC:

processo lungo e costoso..

1. Calcolo degli *overlap* tra i *reads*
2. Costruzione dell'Overlap Graph (OG) → *Layout*
3. Visita del grafo → *Consensus*

Ricostruzione del primo genoma umano (Sanger *reads*) (HGP, Gene Myers, Celera)

poche/lunghe

ha definito il problema computazione

OLC: usato anche per analizzare "differenze" tra frammenti di genomi senza assemblare tutta la sequenza

Fragment Assembly

Approccio dBG:

1. Estrazione dei k-mers dai *reads*

facile

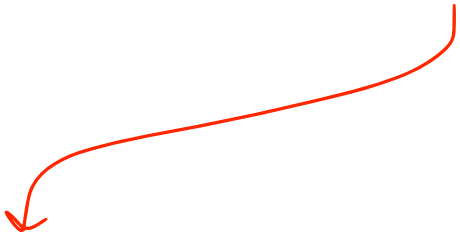
2. Costruzione del grafo di de Bruijn (dBG)

3. Visita del grafo

Ricostruzione di genomi con NGS *reads*

tante ma corte

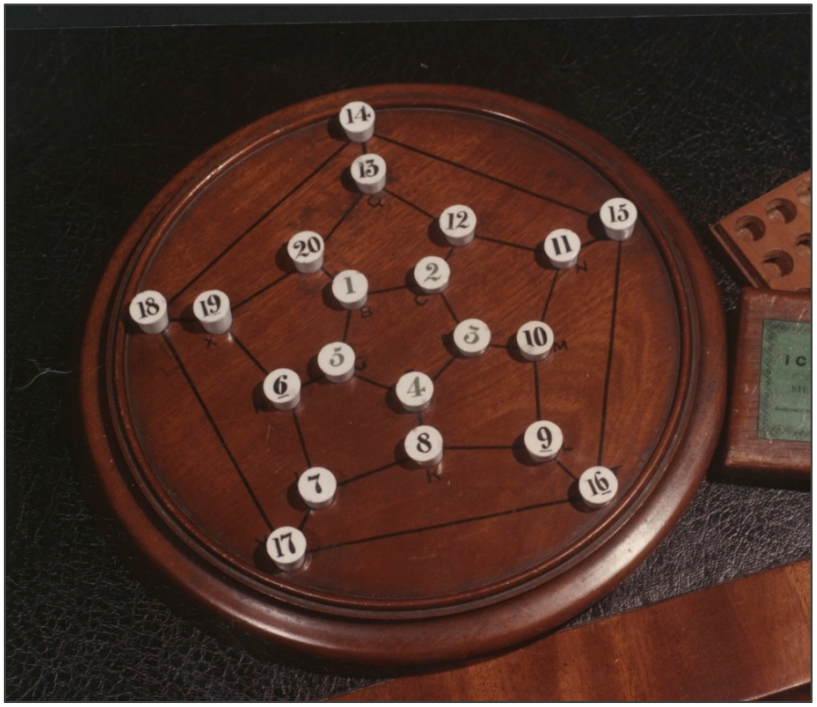
Graph traversal and…

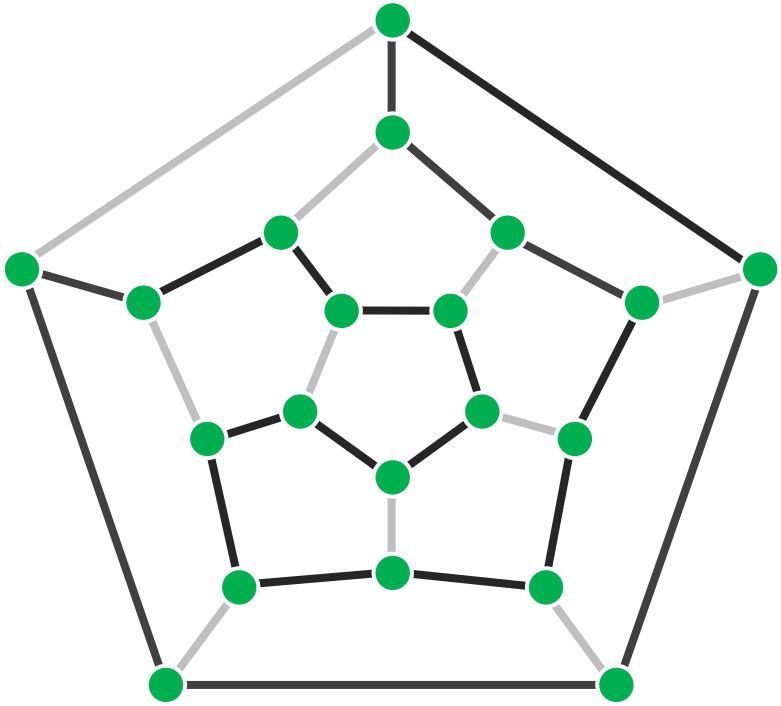


The Icosian Game and the Bridges of Konigsberg

The Origin of "Hamiltonian" Path/Cycle

Icosian game: William *Hamilton,* 1857. Objective is to place pegs 1-20 one at a time in adjacent holes.

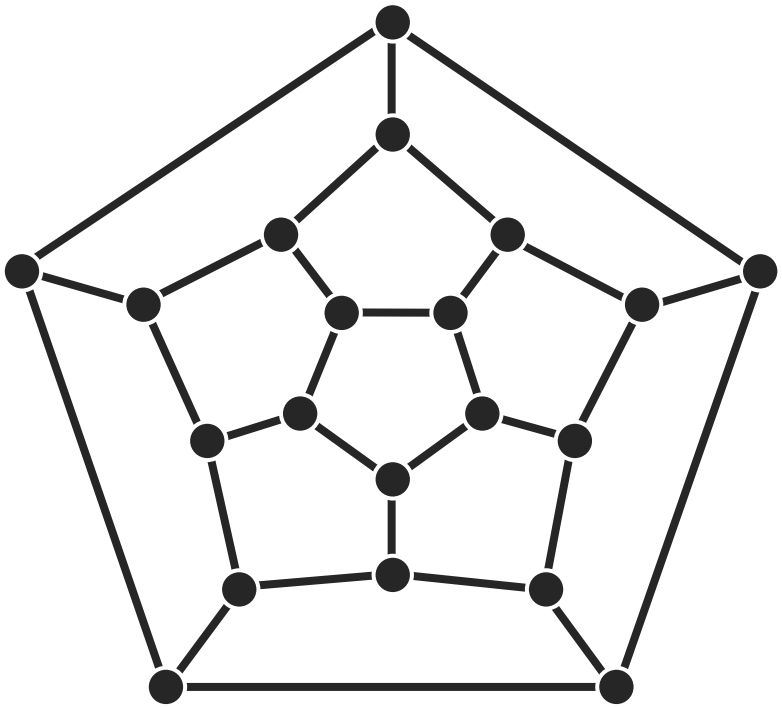




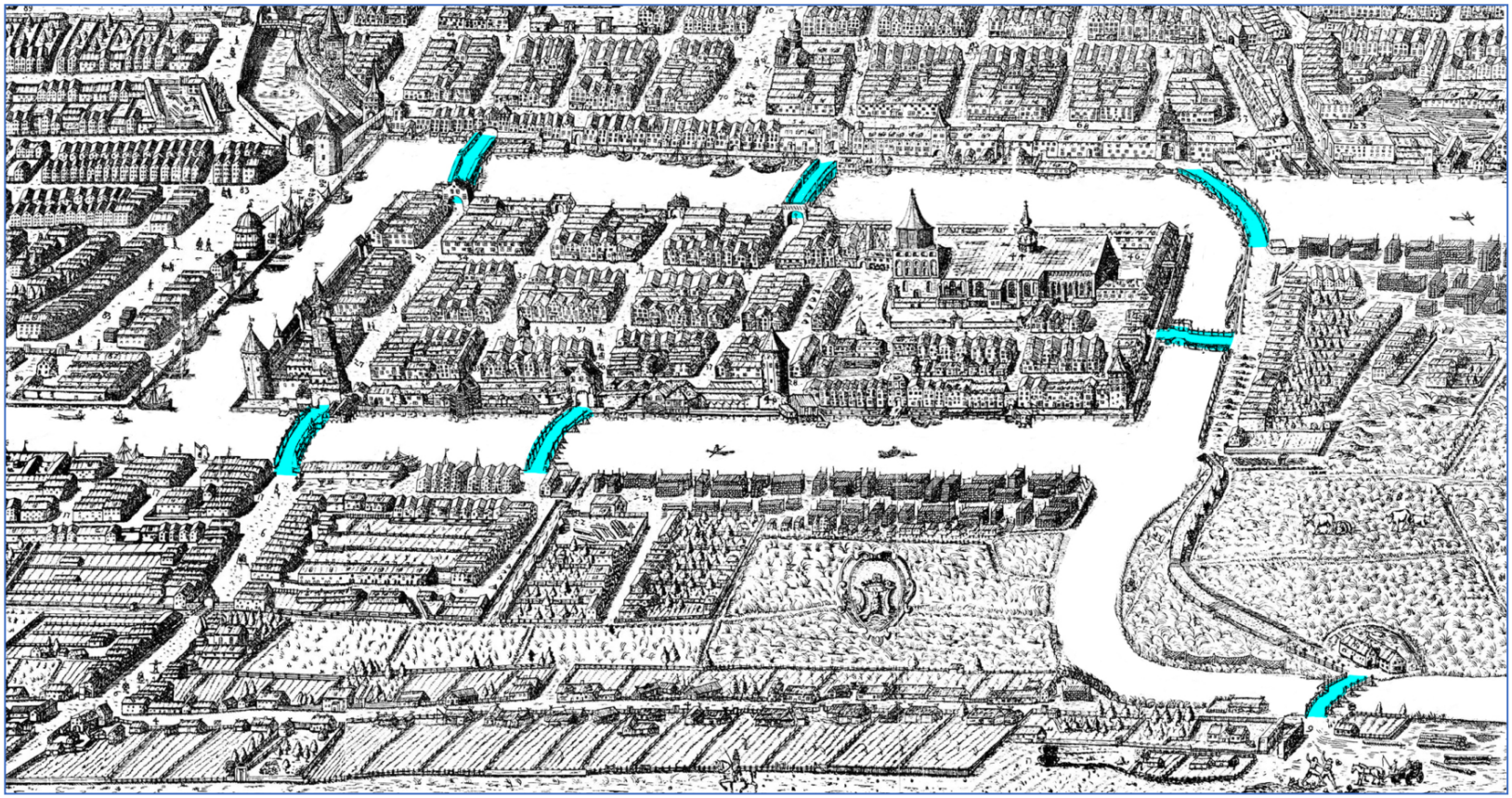
The Origin of "Hamiltonian" Path/Cycle

Hamiltonian cycle: A Hamiltonian path that returns to its starting node.

Exercise: Can you find a Hamiltonian cycle in this graph? (What algorithm did you use?)



The Bridges of Königsberg



STOP: Is it possible to walk across each bridge *exactly once* and return to the starting point?

Leonhard Euler's Insight (1735)

Even better, Euler would *prove* how to quickly determine whether a graph has an Eulerian cycle.

Intractable Problems

Even better, Euler would *prove* how to quickly determine whether a graph has an Eulerian cycle.

Key Point: And yet no one has ever found a polynomial-time algorithm to find a Hamiltonian cycle in a graph!

Similar Problems with Different Fates

Hamiltonian Cycle Problem

NP-Complete

Input: a network with *n* nodes.

Output: "Yes" if there is a cycle visiting every ***node*** in the network; "No" otherwise.

Eulerian Cycle Problem

Polynomial

Input: a network with *n* nodes.

Output: "Yes" if there is a cycle visiting every ***edge*** in the network; "No" otherwise.

Shortest Superstring Problem (Gene Myers)

SSP

INPUT: $r_1, r_2, \ldots, r_n$, stringhe su $\Sigma = \{A,C,G,T\}$

OUTPUT: stringa G di minima lunghezza tale che ogni stringa r_i sia sottostringa di G

Ipotesi: assenza di errori di sequenziamento

Shortest Superstring Problem (Gene Myers)

Esempio per $\Sigma = \{0,1\}$

000, 001, 010, 100, 101, 110, 111

Due soluzioni:

1000101110

0001110100

Shortest Superstring Problem (Gene Myers)

Esempio per $\Sigma = \{0,1\}$

000, 001, 010, 100, 101, 110, 111

Due soluzioni:

1000101110

0001110100

Shortest Superstring Problem (Gene Myers)

Esempio per $\Sigma = \{0,1\}$

000, 001, 010, 100, 101, 110, 111

Due soluzioni:

10001011110

0001110100

Shortest Superstring Problem (Gene Myers)

Esempio per $\Sigma = \{0,1\}$

000, 001, 010, 100, 101, 110, 111

Due soluzioni:

10001011110

00011101000

Shortest Superstring Problem (Gene Myers)

Esempio per $\Sigma = \{0,1\}$

000, 001, 010, 100, 101, 110, 111

Due soluzioni:

10001011110

00011101100

Overlap Layout Consensus

OVERLAP tra r_1 e r_2

→ più lungo suffisso di r_1 che si ripete in un prefisso di r_2 (o viceversa)

```
ATGGAGAGAGA
          AGAGAGAAGT
```

Overlap Layout Consensus

OVERLAP tra r_1 e r_2

→ più lungo suffisso di r_1 che si ripete in un prefisso di r_2 (o viceversa)

```
ATGGAGAGAGA
    AGAGAGAAGT
```

OVERLAP

Overlap Layout Consensus

OVERLAP tra r_1 e r_2

→ più lungo suffisso di r_1 che si ripete in un prefisso di r_2 (o viceversa)

```
ATGGAGAGAGA
     AGAGAGAAGT
```

EXTENSION

Overlap Layout Consensus

OVERLAP tra r_1 e r_2

→ più lungo suffisso di r_1 che si ripete in un prefisso di r_2 (o viceversa)

```
ATGGAGAGAGA
      AGAGAGAAGT
```

EXTENSION

```
ATGGAGAGAGAAGT
```

ASSEMBLY di r_1 e r_2

Overlap Layout Consensus

ASSEMBLY tra tre reads r_1, r_2 e r_3

```
ATGGAGAGAGA GTGTCCGT
     AGAGAGAAGT
```

```
ATGGAGAGAGAAGT
```

ASSEMBLY?

Overlap Layout Consensus

ASSEMBLY tra tre reads r_1, r_2 e r_3

```
ATGGAGAGAGA  GTGTCCGT
     AGAGAGAAGT

ATGGAGAGAGAAGT
```

ASSEMBLY?

Overlap Layout Consensus

ASSEMBLY tra tre reads r_1, r_2 e r_3

ATGGAGAGAGA GTGTCCGT

AGAGAGAAGT

ATGGAGAGAGAAGT

ASSEMBLY?

Overlap Layout Consensus

ASSEMBLY tra tre reads r_1, r_2 e r_3

ATGGAGAGAGA GTGTCCGT

AGAGAGAAGT

ATGGAGAGAGAAGTGTCCGT

ASSEMBLY

Overlap Layout Consensus

ASSEMBLY tra quattro reads r_1, r_2, r_3 e r_4

```
ATGGAGAGAGA  GTGTCCGT
     AGAGAGAAGT     CGTAAATCC

ATGGAGAGAGAAGTGTCCGT
```

ASSEMBLY

Overlap Layout Consensus

ASSEMBLY tra quattro reads r_1, r_2, r_3 e r_4

ATGGAGAGAGA GTGTCCGT

AGAGAGAAGT CGTAAATCC

ATGGAGAGAGAAGTGTCCGT

ASSEMBLY

Overlap Layout Consensus

ASSEMBLY tra quattro reads r_1, r_2, r_3 e r_4

ATGGAGAGAGA GTGTCCGT

AGAGAGAAGT CGTAAATCC

ATGGAGAGAGAAGTGTCCGTAAATCC

ASSEMBLY

Questo sarà fatto sul grafo…

Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

Grafo completo dei reads

Esempio da Pevzner

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

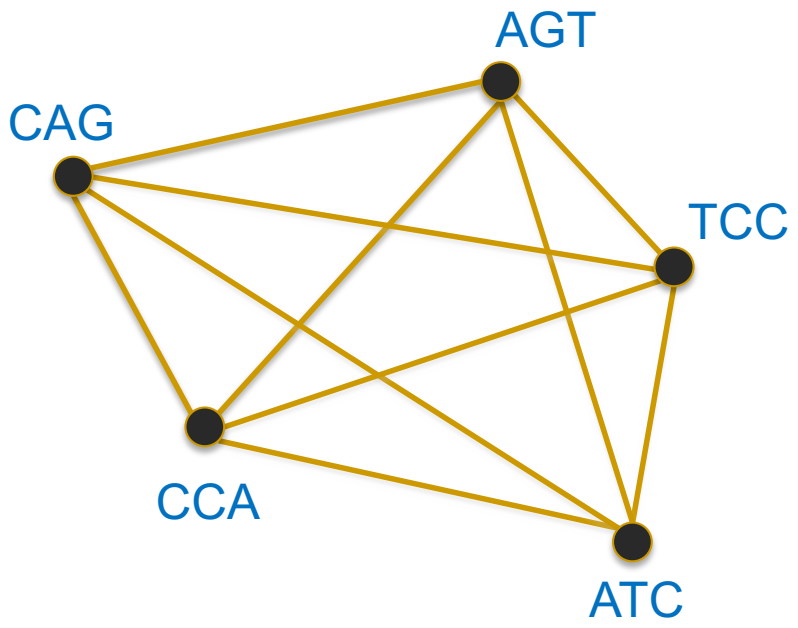
$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$



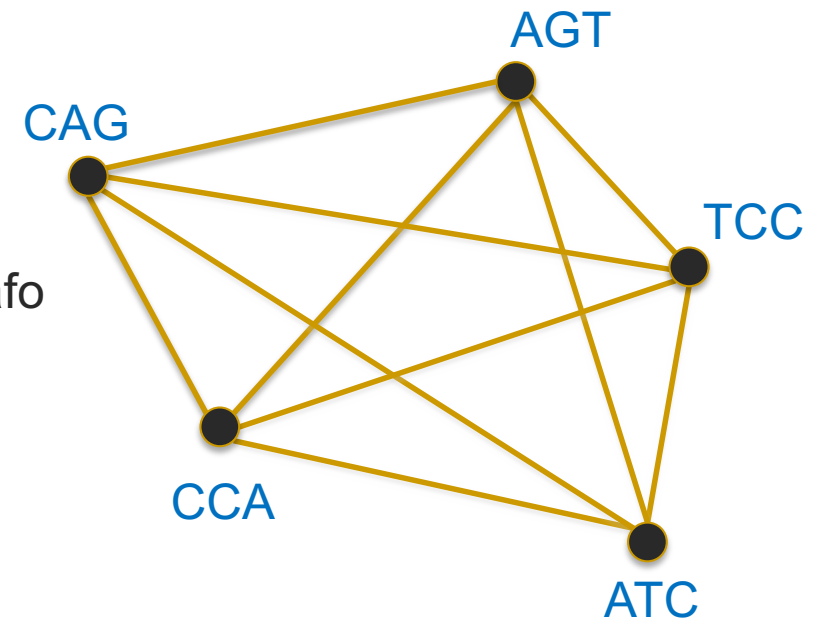
Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CAMMINO HAMILTONIANO (CH):
cammino che tocca tutti i vertici del grafo una e una sola volta



Grafo completo dei reads

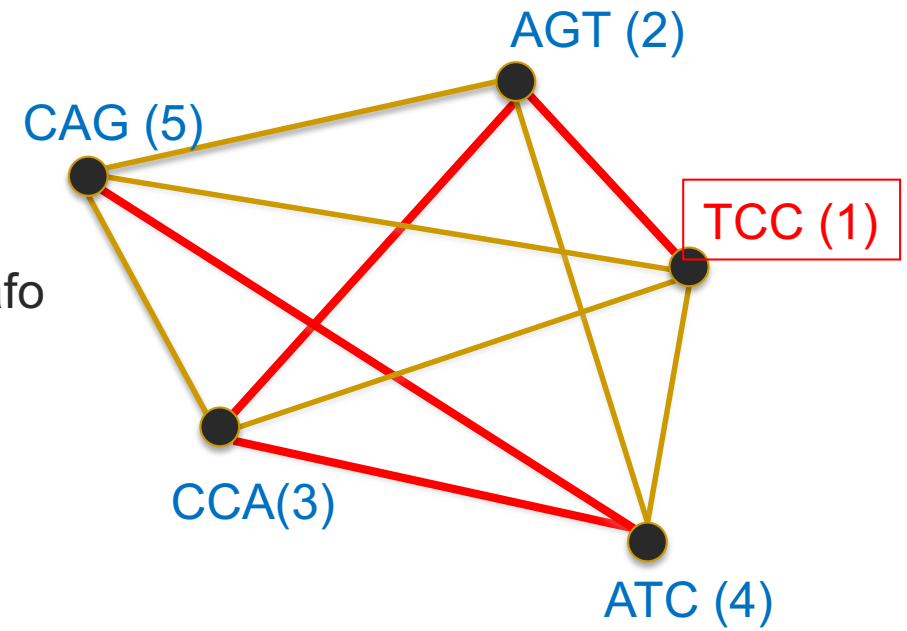
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CAMMINO HAMILTONIANO (CH):
cammino che tocca tutti i vertici del grafo una e una sola volta

CH = < TCC, AGT, CCA, ATC, CAG >

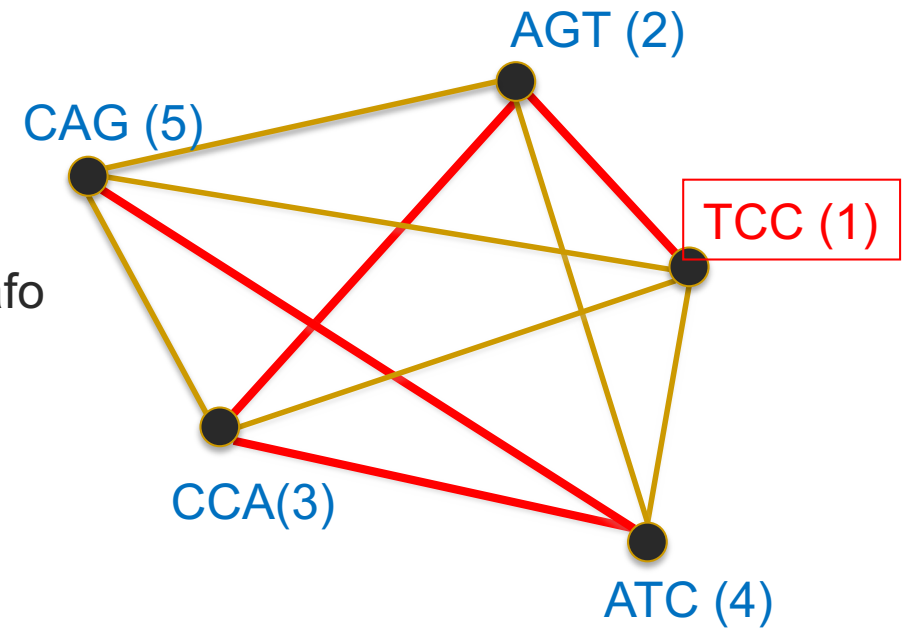


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$



CAMMINO HAMILTONIANO (CH):
cammino che tocca tutti i vertici del grafo una e una sola volta

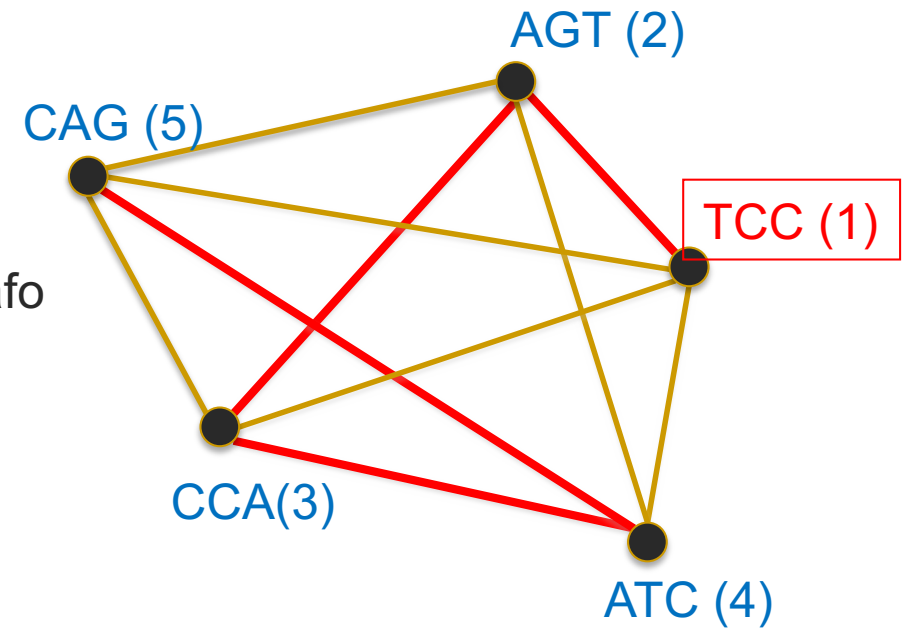
CH = < TCC, AGT, CCA, ATC, CAG >
Superstringa → TCC

Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$



CAMMINO HAMILTONIANO (CH):
cammino che tocca tutti i vertici del grafo una e una sola volta

CH = < TCC, AGT, CCA, ATC, CAG >
Superstringa → TCCAGT

Grafo completo dei reads

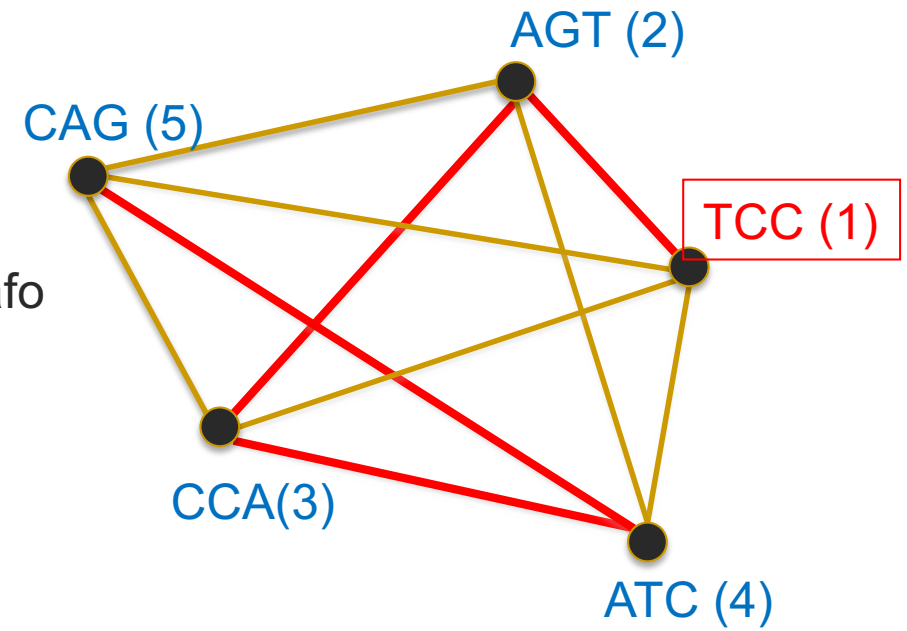
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CAMMINO HAMILTONIANO (CH):
cammino che tocca tutti i vertici del grafo una e una sola volta

CH = < TCC, AGT, CCA, ATC, CAG >
Superstringa → TCCAGTCCA



Grafo completo dei reads

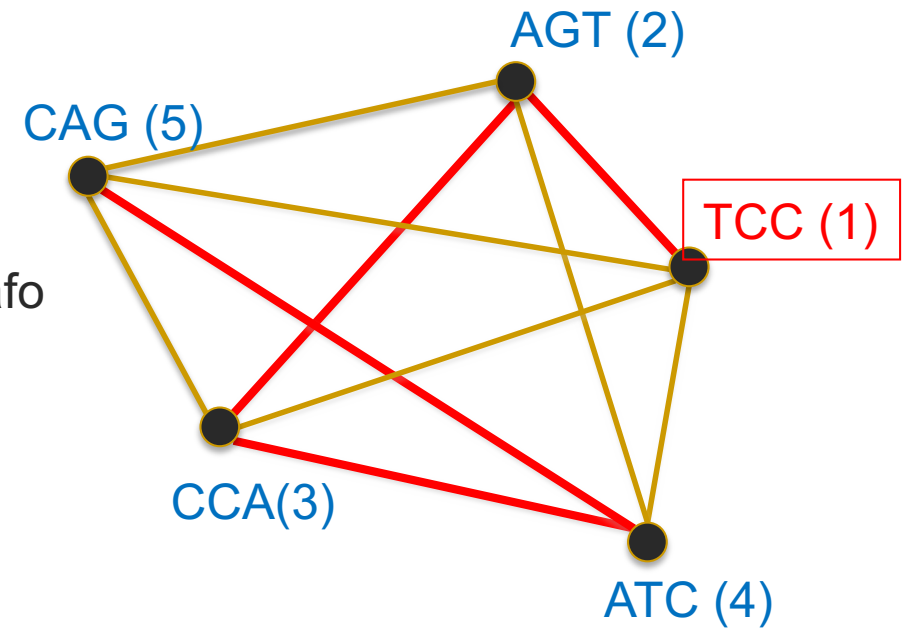
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CAMMINO HAMILTONIANO (CH):
cammino che tocca tutti i vertici del grafo una e una sola volta

CH = < TCC, AGT, CCA, ATC, CAG >
Superstringa → TCCAGTCCAATC

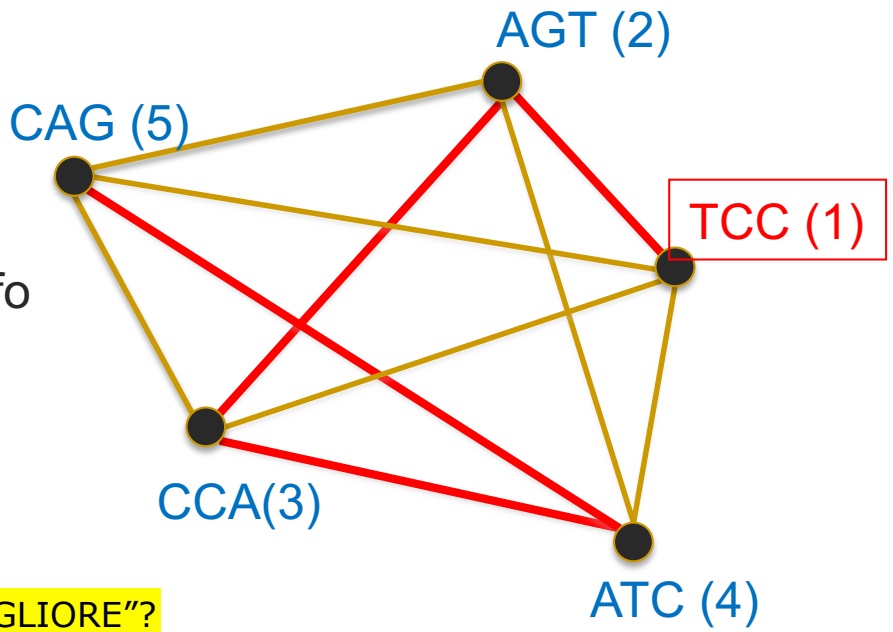


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$



CAMMINO HAMILTONIANO (CH):
cammino che tocca tutti i vertici del grafo una e una sola volta

CH = < TCC, AGT, CCA, ATC, CAG >
Superstringa → TCCAGTCCAATCCAG

HO CONCATENATO! POSSO COSTRUIRNE UNA "MIGLIORE"?

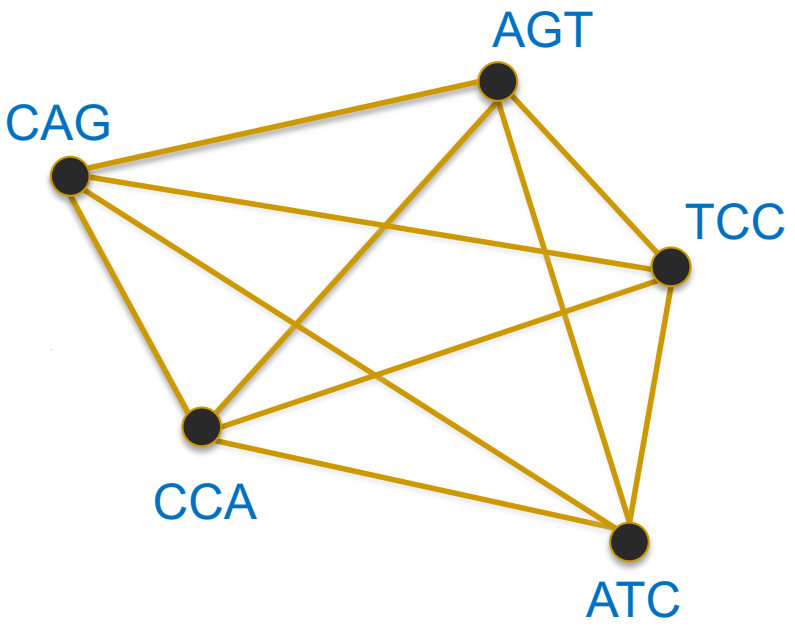
Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

Aggiungo un peso: l'overlap tra le due sequenze

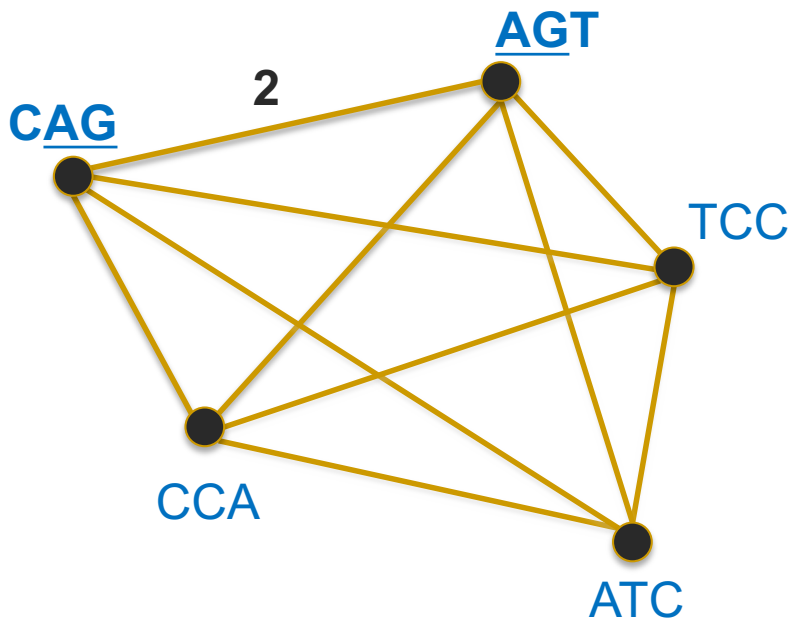


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

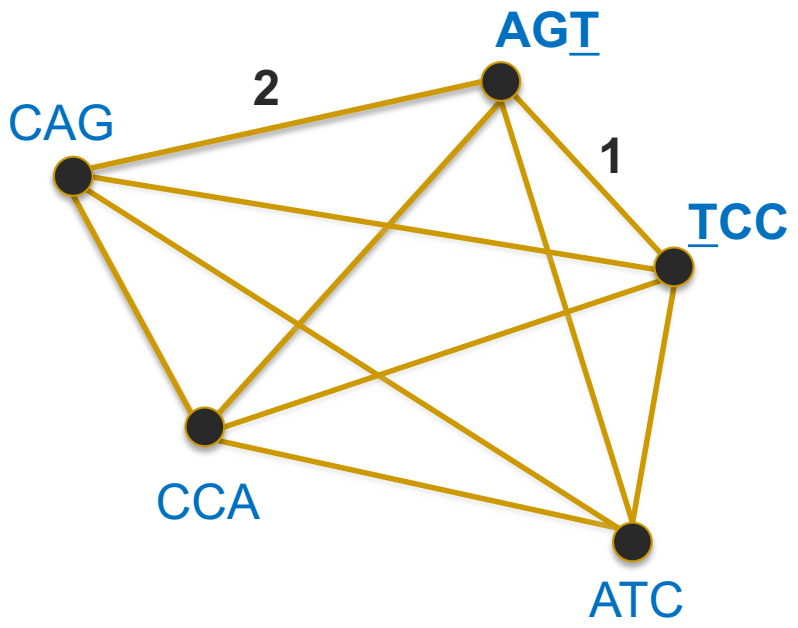


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

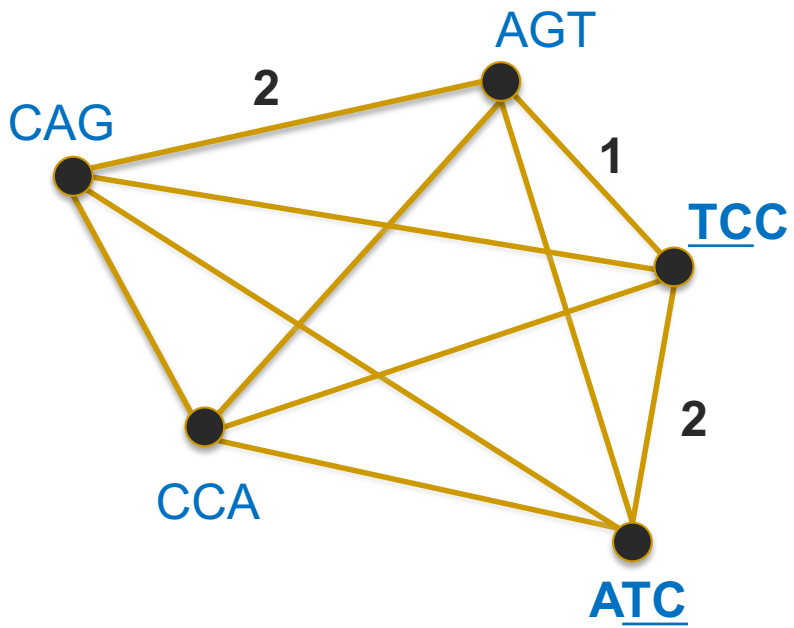


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

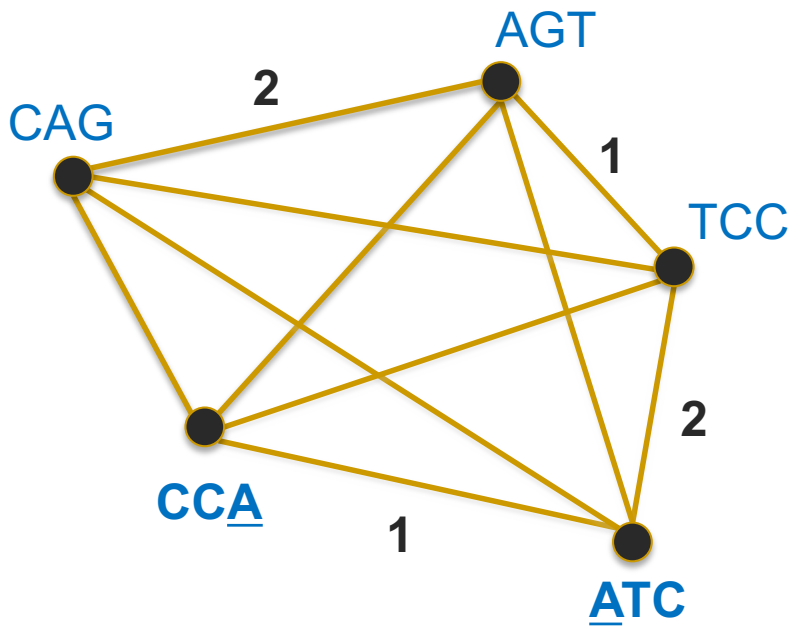


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

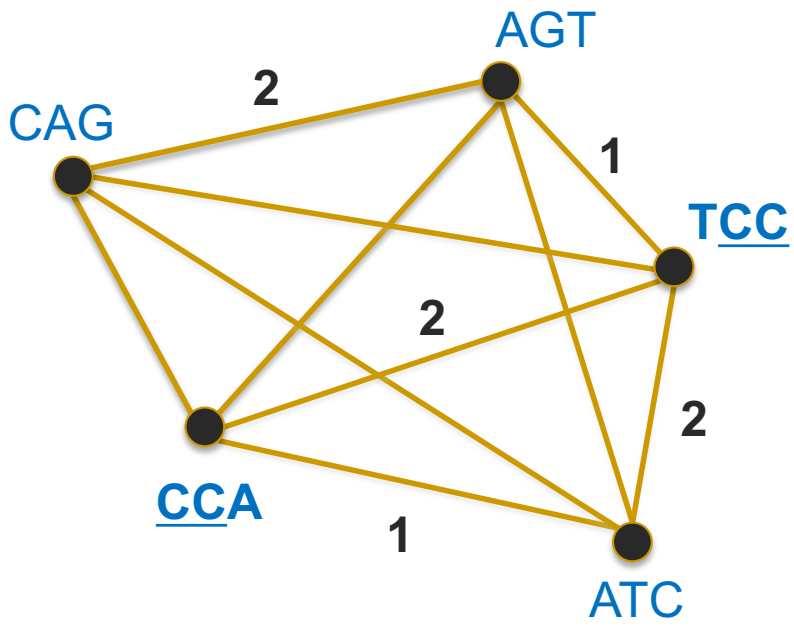


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

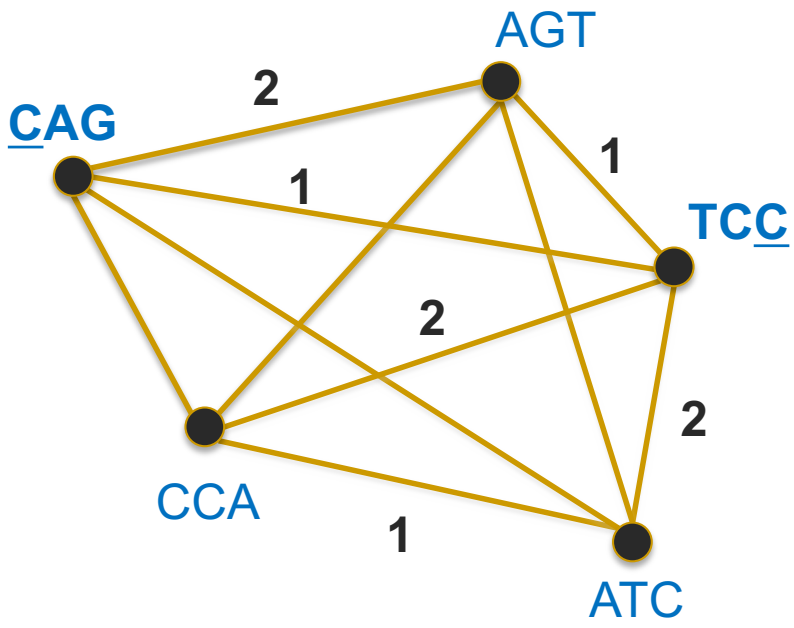


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

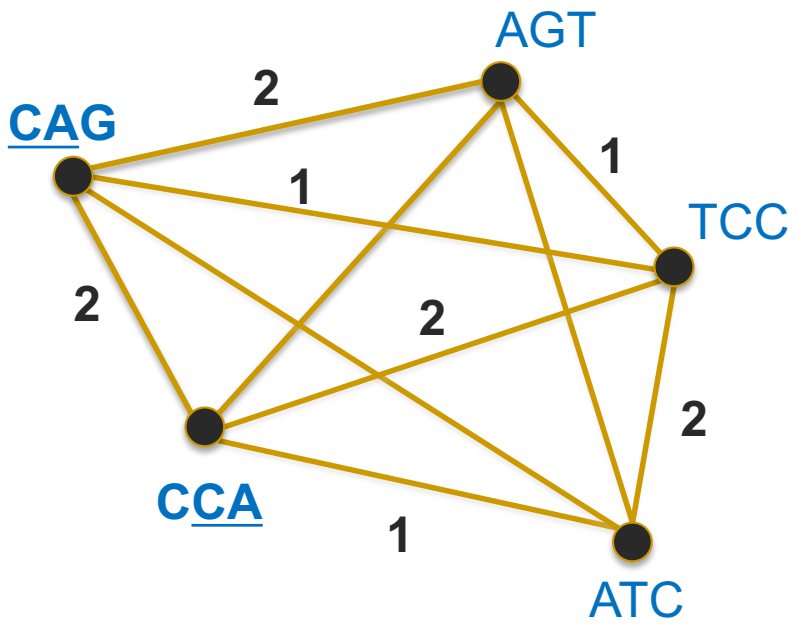


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

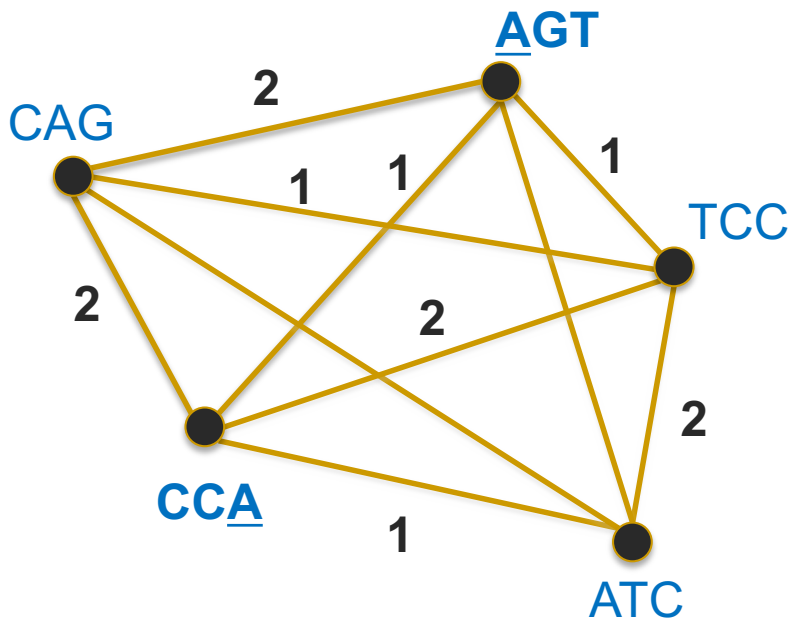


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

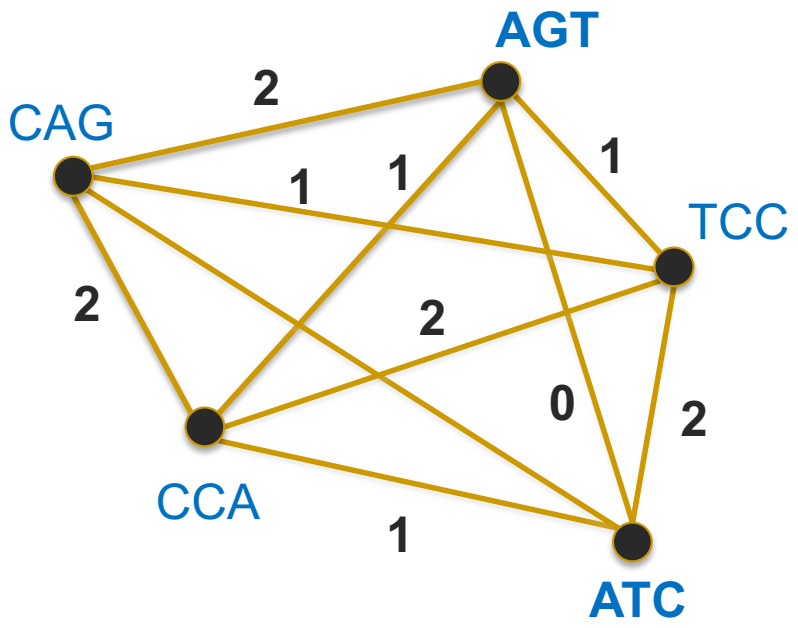


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

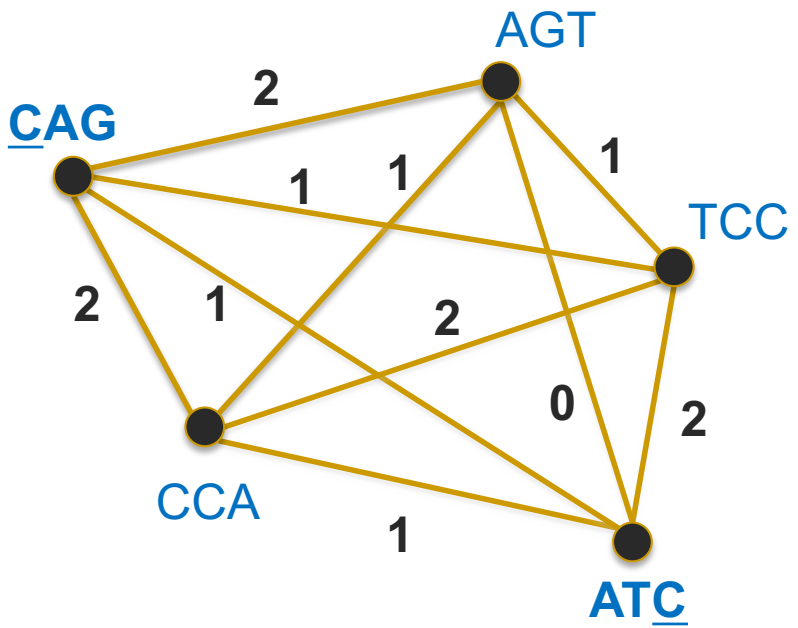


Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$



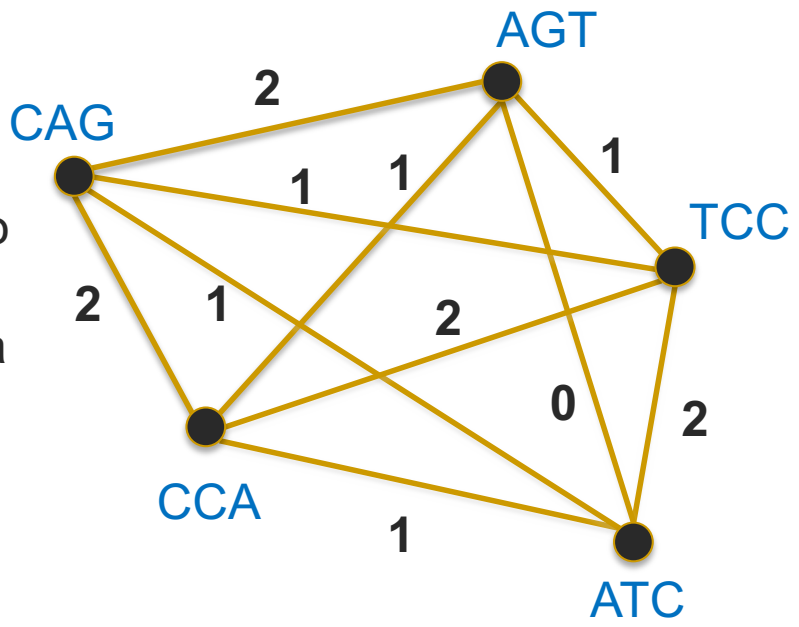
Grafo completo dei reads

$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima



Grafo completo dei reads

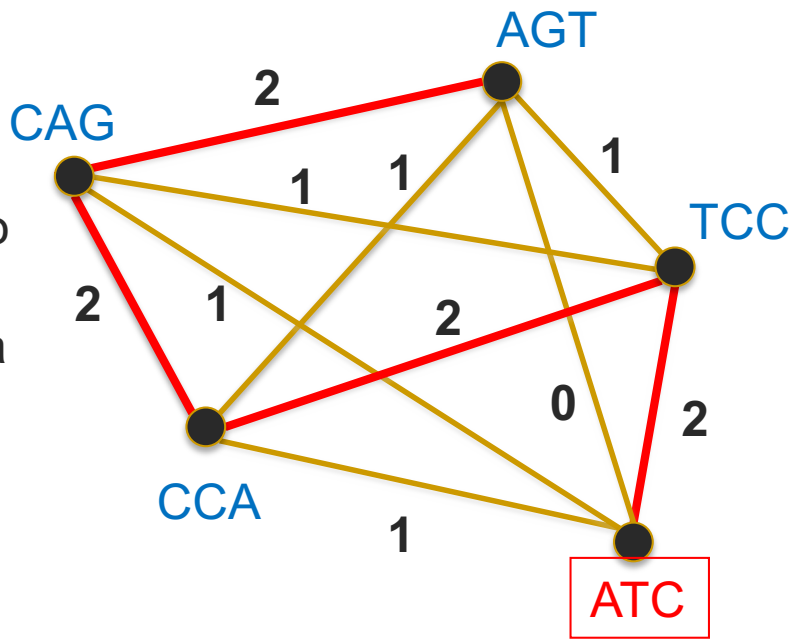
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima

CH = < ATC, TCC, CCA, CAG, AGT >



Grafo completo dei reads

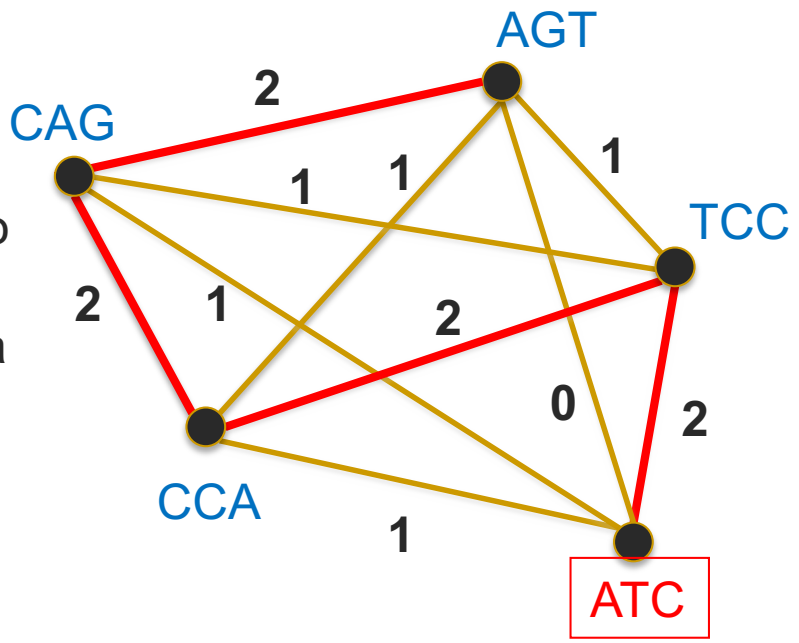
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima

CH = < ATC, TCC, CCA, CAG, AGT >
Superstringa →



Grafo completo dei reads

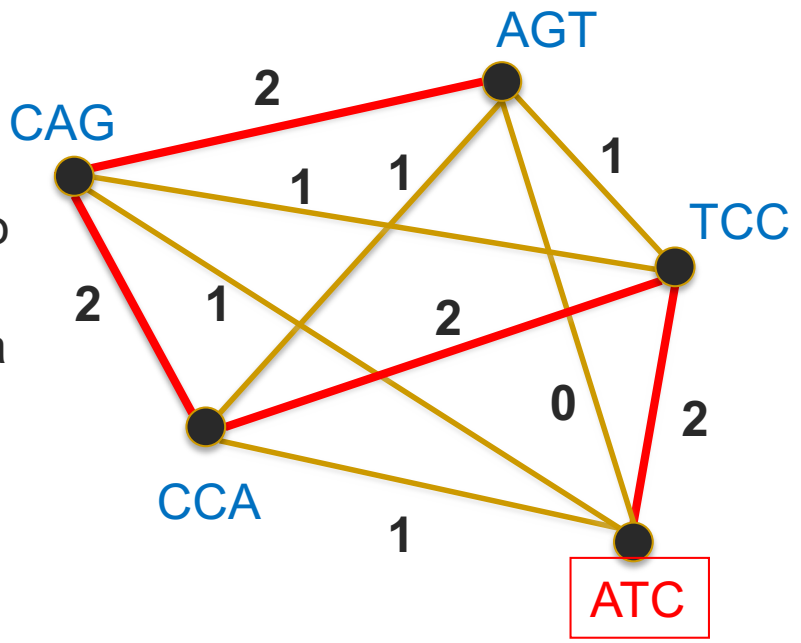
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima

CH = < ATC, TCC, CCA, CAG, AGT >
Superstringa → ATCC



Grafo completo dei reads

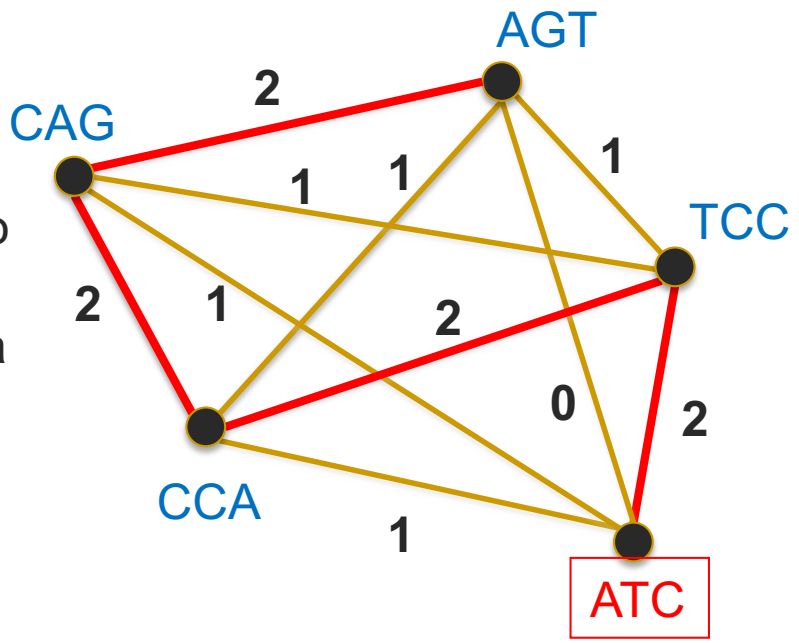
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima

CH = < ATC, TCC, CCA, CAG, AGT >
Superstringa → ATCCA



Grafo completo dei reads

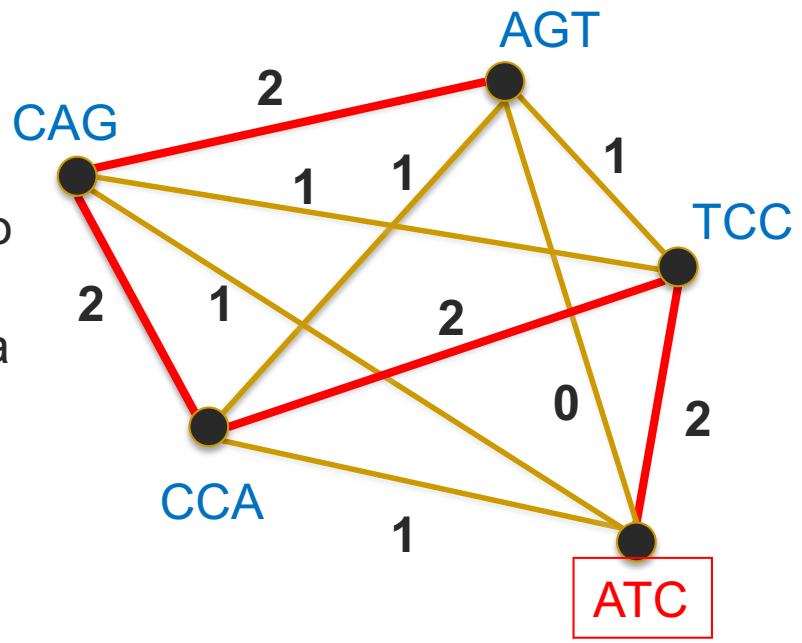
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima

CH = < ATC, TCC, CCA, CAG, AGT >
Superstringa → ATCCAG



Grafo completo dei reads

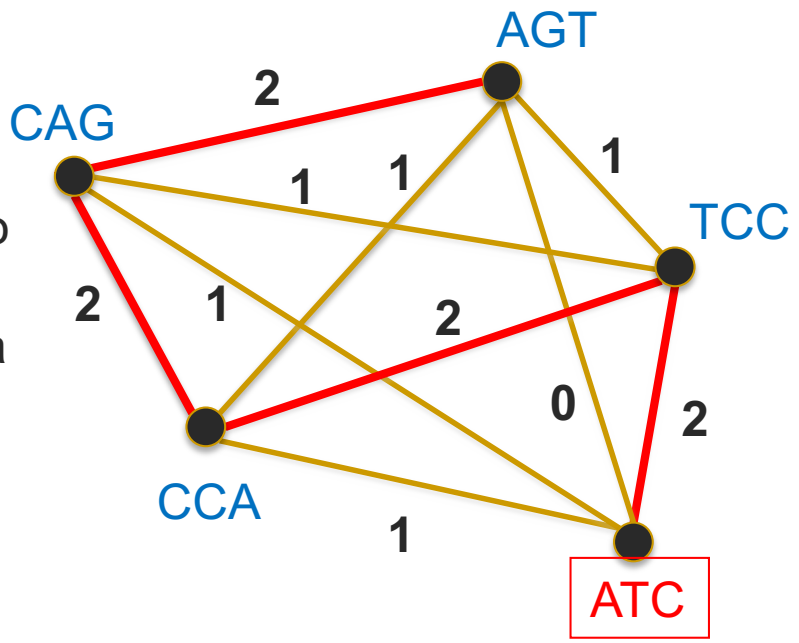
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima

CH = < ATC, TCC, CCA, CAG, AGT >
Superstringa → ATCCAGT



Grafo completo dei reads

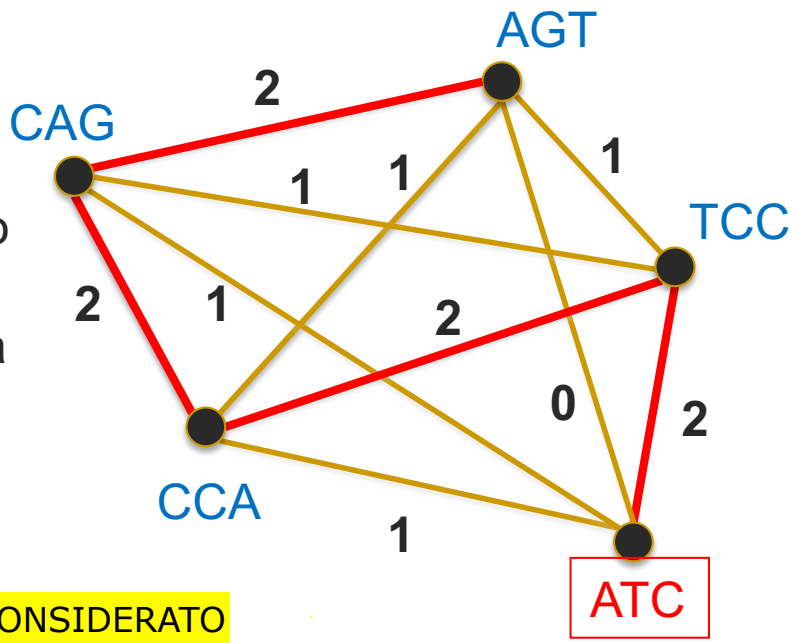
$G = (V, E)$

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima

CH = < ATC, TCC, CCA, CAG, AGT >
Superstringa → **ATCCAGT**



PIU' CORTA PERCHE' E' MASSIMO L'OVERLAP CONSIDERATO

Grafo completo dei reads

$G = (V, E)$

Problema NP-completo
(e quindi si sono trovate euristiche)

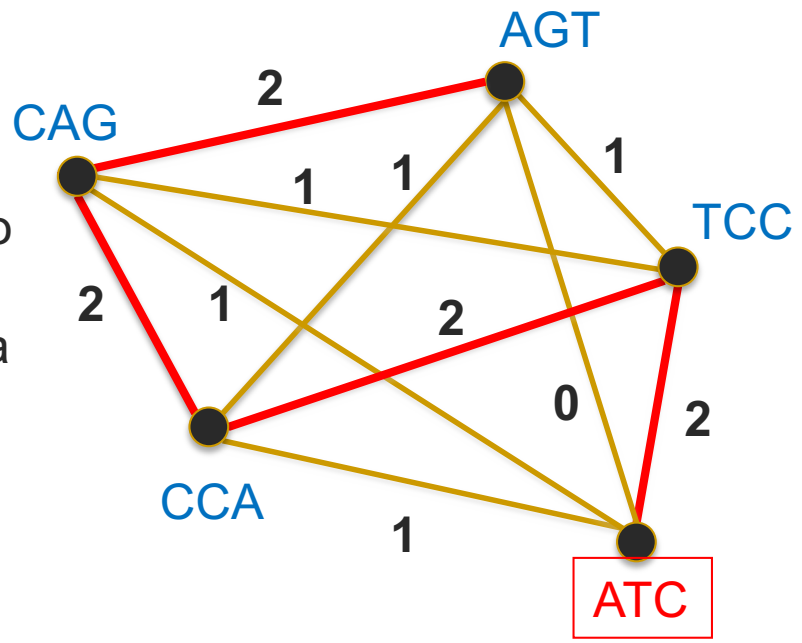
HGP 13 anni…
(e c'è lo scaffolding…)

$V = \{r_1, r_2, \ldots, r_n\} = \{AGT, TCC, ATC, CAG, CCA\}$

$E = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V\}$

CH DI PESO MASSIMO:
cammino che tocca tutti i vertici del grafo una e una sola volta tale che la somma dei pesi degli archi coinvolti sia massima

CH = < ATC, TCC, CCA, CAG, AGT >
Superstringa → **ATCCAGT**



DEF: Overlap

Overlap ov(r_1, r_2)

→ più lungo suffisso di r_1 che si ripete in un prefisso di r_2

r_1 ATGGAGAGAGA
r_2 AGAGAGAAGT

DEF: Overlap

Overlap ov(r_1, r_2)

→ più lungo suffisso di r_1 che si ripete in un prefisso di r_2

r_1 ATGGAGAGAGA
r_2 AGAGAGAAGT

AGAGAGA → ov(r_1, r_2)
AGT → ex(r_1, r_2)

Overlap Graph (OG)

$\mathbf{OG = (V, A)}$

$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in \mathbf{V}, ov(r_i, r_j) \neq \varepsilon\}$

Overlap Graph (OG)

$OG = (V, A)$

$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

A: insieme degli archi (orientati)

Overlap Graph (OG)

$OG = (V, A)$

$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

A: insieme degli archi (orientati)

L'arco (r_i, r_j) viene etichettato da $ex(r_i, r_j)$

Overlap Graph (OG)

$OG = (V, A)$

$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

non necessariamente completo

A: insieme degli archi (orientati)

L'arco (r_i, r_j) viene etichettato da $ex(r_i, r_j)$

NB: di solito si richiede che $|ov(r_i, r_j)|$ superi una certa soglia

Overlap Graph (OG)

$OG = (V, A)$

$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

ATGGAGAGATG

AGATGGAAGT

Overlap Graph (OG)

$OG = (V, A)$

$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

ATGGAGAGATG

AGATGGAAGT

ASSEMBLY ➔ **ATGGAGAGATGGAAGT**

Overlap Graph (OG)

$OG = (V, A)$

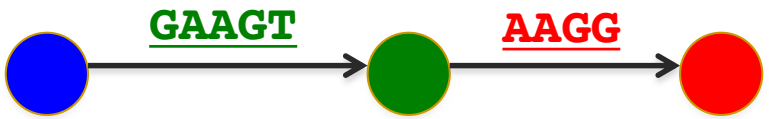
$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

ATGGAGAGATG

AGATGGAAGT

AGTAAGG



ASSEMBLY ➔ **ATGGAGAGATGGAAGT**

Overlap Graph (OG)

$OG = (V, A)$

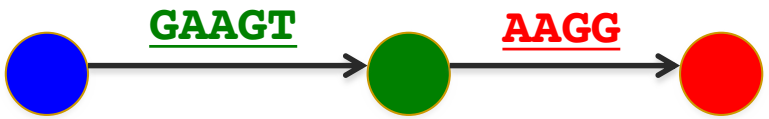
$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

ATGGAGAGATG

AGATGGAAGT

AGTAAGG



ASSEMBLY ➔ **ATGGAGAGATGGAAGTAAGG**

Overlap Graph (OG)

OG = (V, A)

$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j)$ t.c. $r_i, r_j \in \mathbf{V}$, $ov(r_i, r_j) \neq \varepsilon\}$

ASSEMBLY lungo il percorso $< r_{i1}, r_{i2}, \ldots, r_{ip}>$

→ concatenazione r_{i1} $ex(r_{i1}, r_{i2})$ $ex(r_{i2}, r_{i3})$ … $ex(r_{i(p-1)}, r_{ip})$

Overlap Graph (OG)

$OG = (V, A)$

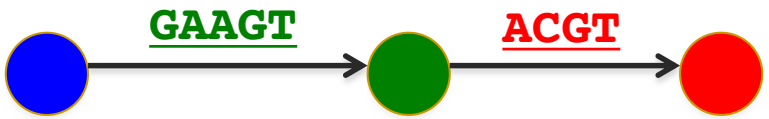
$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

ATGGAGAGATG

AGATGGAAGT

ATGGAAGTACGT



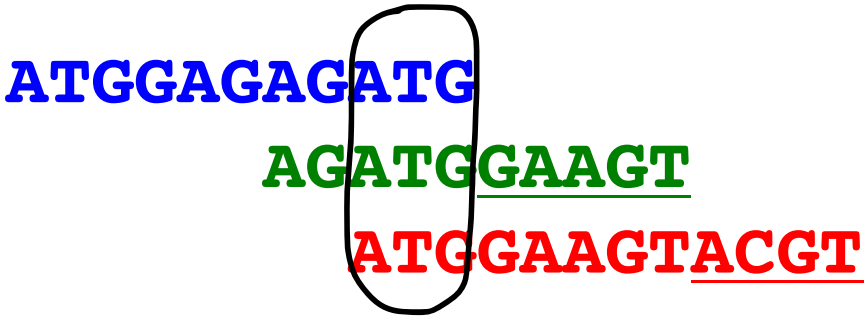
ASSEMBLY ➔ **ATGGAGAGATGGAAGTACGT**

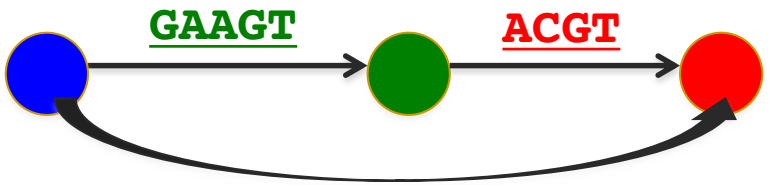
Overlap Graph (OG)

OG = (V, A)

$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in \mathbf{V}, ov(r_i, r_j) \neq \varepsilon\}$





ASSEMBLY ➔ ATGGAGAGATGGAAGTACGT

Overlap Graph (OG)

OG = (V, A)

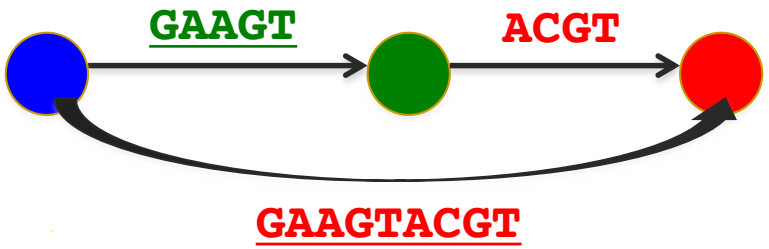
$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in \mathbf{V}, ov(r_i, r_j) \neq \varepsilon\}$

ATGGAGAGATG

AGATGGAAGT

ATGGAAGTACGT



ASSEMBLY ➔ **ATGGAGAGATGGAAGTACGT**

ho 2 assemblaggi uguali… E' inutile…

Overlap Graph (OG)

$OG = (V, A)$

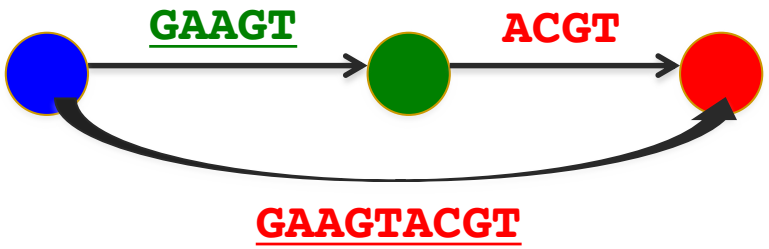
$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j) \text{ t.c. } r_i, r_j \in V, ov(r_i, r_j) \neq \varepsilon\}$

ATGGAGAGATG

AGATGGAAGT

ATGGAAGTACGT



ASSEMBLY ➔ **ATGGAGAGATGGAAGTACGT**

potrei non toccarlo…
Non sarebbe HP…

Overlap Graph (OG)

OG = (V, A)

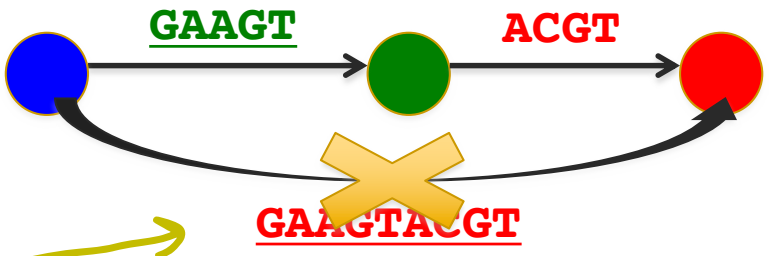
$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j)$ t.c. $r_i, r_j \in \mathbf{V}$, $ov(r_i, r_j) \neq \varepsilon\}$

ATGGAGAGATG

AGATGGAAGT

ATGGAAGTACGT



Arco transitivo (riducibile): arco (r_i, r_j) tale per cui esiste un percorso da r_i a r_j che ha almeno un vertice intermedio

String Graph (SG)

SG = **(V, A)**

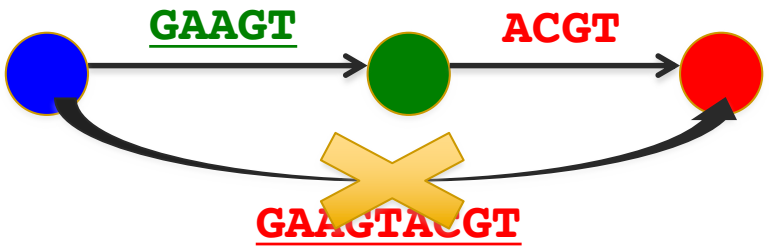
$V = \{r_1, r_2, \ldots, r_n\}$

$A = \{(r_i, r_j)$ t.c. $r_i, r_j \in \mathbf{V}$, $ov(r_i, r_j) \neq \varepsilon$, (r_i, r_j) non transitivo$\}$

ATGGAGAGATG

AGATGGAAGT

ATGGAAGTACGT



Arco transitivo (riducibile): arco (r_i, r_j) tale per cui esiste un percorso da r_i a r_j che ha almeno un vertice intermedio

Overlap Layout Consensus

1. Calcolo degli *overlap* tra i *reads*
2. Costruzione dell'Overlap Graph
3. Riduzione a String Graph Myers, 2005
4. Visita dello String Graph → *contigs*

(cancello archi: non posso cancellare nodi perchè potrei perdere altri percorsi)

assemblaggio che riesco ad ottenere percorrendo percorso "unici" tra nodi. Può essere lento. Ma con opportune strutture dati… (vedremo qualche articolo)

Calcolo dell'overlap

1. Calcolo degli *overlap* tra i *reads*
2. Costruzione dell'Overlap Graph
3. Riduzione a String Graph
4. Visita dello String Graph → *contigs*

Calcolo dell'overlap

1. Calcolo degli *overlap* tra i *reads*

   - ✓ Programmazione dinamica
   - ✓ Approccio *seed-extend*
   - ✓ Burrows-Wheeler Transform e FM-index
   - ✓ Strategie *alignment-free* (MinHash, MinSketch, etc.)

     Uso distanze e somiglianze tra read… (fingerprint basata sulla fattorizzazione di Lyndon)

     (non c'è confronto base-base)

Assemblatori OLC

De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer

2008

David Hernandez,[1,3] Patrice François,[1] Laurent Farinelli,[2] Magne Østerås,[2] and Jacques Schrenzel[1]

[1]Genomic Research Laboratory, Infectious Diseases Service, Geneva University Hospitals and the University of Geneva, CH-1211 Geneva 4, Switzerland; [2]Fasteris SA, CH-1228 Plan-les-Ouates, Switzerland

Novel high-throughput DNA sequencing technologies allow researchers to characterize a bacterial genome during a single experiment and at a moderate cost. However, the increase in sequencing throughput that is allowed by using such platforms is obtained at the expense of individual sequence read length, which must be assembled into longer contigs to be exploitable. This study focuses on the Illumina sequencing platform that produces millions of very short sequences that are 35 bases in length. We propose a de novo assembler software that is dedicated to process such data. Based on a classical overlap graph representation and on the detection of potentially spurious reads, our software generates a set of accurate contigs of several kilobases that cover most of the bacterial genome. The assembly results were validated by comparing data sets that were obtained experimentally for *Staphylococcus aureus* strain MW2 and *Helicobacter acinonychis* strain Sheeba with that of their published genomes acquired by conventional sequencing of 1.5- to 3.0-kb fragments. We also provide indications that the broad coverage achieved by high-throughput sequencing might allow for the detection of clonal polymorphisms in the set of DNA molecules being sequenced.

[Supplemental material is available online at www.genome.org. Edena is freely available for academic users at http://www.genomic.ch/edena.]

Assemblatori OLC

Efficient de novo assembly of large genomes using compressed data structures

SGA
2012

Jared T. Simpson and Richard Durbin[1]

Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SA, United Kingdom

De novo genome sequence assembly is important both to generate new sequence assemblies for previously uncharacterized genomes and to identify the genome sequence of individuals in a reference-unbiased way. We present memory efficient data structures and algorithms for assembly using the FM-index derived from the compressed Burrows-Wheeler transform, and a new assembler based on these called SGA (String Graph Assembler). We describe algorithms to error-correct, assemble, and scaffold large sets of sequence data. SGA uses the overlap-based string graph model of assembly, unlike most de novo assemblers that rely on de Bruijn graphs, and is simply parallelizable. We demonstrate the error correction and assembly performance of SGA on 1.2 billion sequence reads from a human genome, which we are able to assemble using 54 GB of memory. The resulting contigs are highly accurate and contiguous, while covering 95% of the reference genome (excluding contigs <200 bp in length). Because of the low memory requirements and parallelization without requiring inter-process communication, SGA provides the first practical assembler to our knowledge for a mammalian-sized genome on a low-end computing cluster.

Non costruisce prima OL e poi SG, ma usando BWT e FM-index inserisce solo archi non transitivi

Assemblatori OLC

Canu: scalable and accurate long-read assembly via adaptive *k*-mer weighting and repeat separation

2017

Sergey Koren,[1,5] Brian P. Walenz,[1,5] Konstantin Berlin,[2] Jason R. Miller,[3] Nicholas H. Bergman,[4] and Adam M. Phillippy[1]

[1]Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, Maryland 20892, USA; [2]Invincea Incorporated, Fairfax, Virginia 22030, USA; [3]J. Craig Venter Institute, Rockville, Maryland 20850, USA; [4]National Biodefense Analysis and Countermeasures Center, Frederick, Maryland 21702, USA

Long-read single-molecule sequencing has revolutionized de novo genome assembly and enabled the automated reconstruction of reference-quality genomes. However, given the relatively high error rates of such technologies, efficient and accurate assembly of large repeats and closely related haplotypes remains challenging. We address these issues with Canu, a successor of Celera Assembler that is specifically designed for noisy single-molecule sequences. Canu introduces support for nanopore sequencing, halves depth-of-coverage requirements, and improves assembly continuity while simultaneously reducing runtime by an order of magnitude on large genomes versus Celera Assembler 8.2. These advances result from new overlapping and assembly algorithms, including an adaptive overlapping strategy based on *tf–idf* weighted MinHash and a sparse assembly graph construction that avoids collapsing diverged repeats and haplotypes. We demonstrate that Canu can reliably assemble complete microbial genomes and near-complete eukaryotic chromosomes using either Pacific Biosciences (PacBio) or Oxford Nanopore technologies and achieves a contig NG50 of >21 Mbp on both human and *Drosophila melanogaster* PacBio data sets. For assembly structures that cannot be linearly represented, Canu provides graph-based assembly outputs in graphical fragment assembly (GFA) format for analysis or integration with complementary phasing and scaffolding techniques. The combination of such highly resolved assembly graphs with long-range scaffolding information promises the complete and automated assembly of complex genomes.

NG50=lunghezza del più corto contig per cui quelli più lunghi coprono almeno il 50% dell'assemblaggio

nature communications

Article

https://doi.org/10.1038/s41467-023-38716-x

Linear time complexity de novo long read genome assembly with GoldRush

Received: 10 November 2022

Accepted: 11 May 2023

Published online: 22 May 2023

Check for updates

Johnathan Wong[1,2] ✉, Lauren Coombe[1,2], Vladimir Nikolić[1], Emily Zhang[1], Ka Ming Nip[1], Puneet Sidhu[1], René L. Warren[1,3] & Inanç Birol[1,3] ✉

Current state-of-the-art de novo long read genome assemblers follow the Overlap-Layout-Consensus paradigm. While read-to-read overlap – its most costly step – was improved in modern long read genome assemblers, these tools still often require excessive RAM when assembling a typical human dataset. Our work departs from this paradigm, foregoing all-vs-all sequence alignments in favor of a dynamic data structure implemented in GoldRush, a de novo long read genome assembly algorithm with linear time complexity. We tested GoldRush on Oxford Nanopore Technologies long sequencing read datasets with different base error profiles sourced from three human cell lines, rice, and tomato. Here, we show that GoldRush achieves assembly scaffold NGA50 lengths of 18.3-22.2, 0.3 and 2.6 Mbp, for the genomes of human, rice, and tomato, respectively, and assembles each genome within a day, using at most 54.5 GB of random-access memory, demonstrating the scalability of our genome assembly paradigm and its implementation.

QUALITY ASSESSMENT OF DNA-SEQ DE NOVO ASSEMBLIES USING QUAST

Marta Benegas Coll | March 7, 2023

Genetic Variation studies often involve analyzing samples from a previously studied species. For instance, it is of interest to examine genomes of various cultivars, strains, or populations of the same species. In such cases, it may be necessary to perform de novo DNA-Seq assembly to obtain the genome of the sequenced organisms, even if a reference genome is available. To ensure the accuracy and reliability of the analysis, it is recommended to employ different assembly algorithms with varying configurations, as different datasets may require distinct analytical approaches.

In this regard, the QUAST tool proves useful, as it allows for easy comparison of different de novo assemblies. To illustrate its functionality, we will present several case studies utilizing a Caenorhabditis elegans WGS dataset. This dataset comprises short paired-end and mate-pair reads of the C. elegans N2 strain, enabling a thorough examination of QUAST's features.