Elementi di Bioinformatica

Gianluca Della Vedova

Univ. Milano–Bicocca
https://gianluca.dellavedova.org

20 ottobre 2022

Grafi di assemblaggio

Assemblaggio di genomi

Tecnologie

Assemblaggio di genomi

Tecnologie

- Porzioni di genoma chiamate read

Assemblaggio di genomi

Tecnologie

- Porzioni di genoma chiamate read
- 50–10000bp (base pairs)

Assemblaggio di genomi

Tecnologie

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

Assemblaggio di genomi

Tecnologie

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

Assemblaggio di genomi

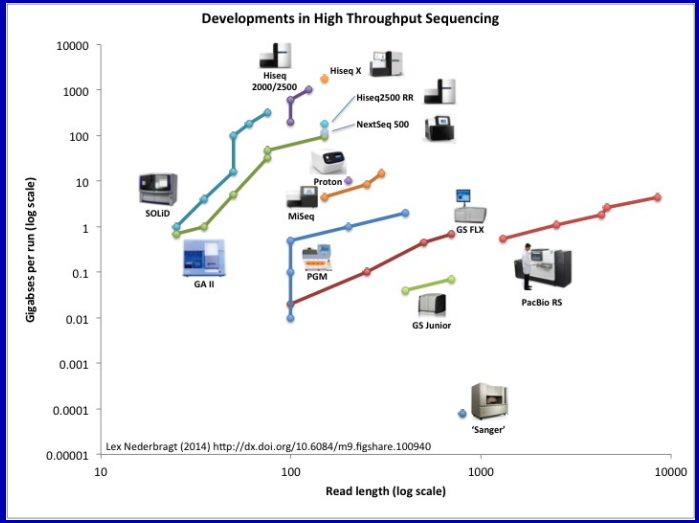
Tecnologie

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

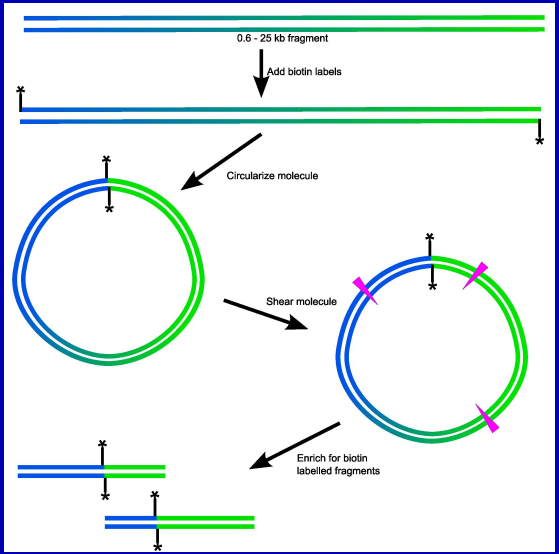
Obiettivo

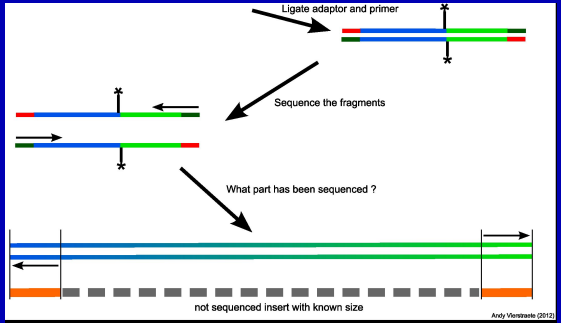
Ricostruire il genoma: circa 3 miliardi bp

Evoluzione tecnologica



Mate pairs


0.6 - 25 kb fragment
Add biotin labels
Circularize molecule
Shear molecule
Enrich for biotin
labelled fragments


Ligate adaptor and primer
Sequence the fragments
What part has been sequenced ?
not sequenced insert with known size
Andy Vierstraete (2012)

Regola 1

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

Overlap — sovrapposizione

```
TCTATATCTCGGCTCTAGG          Read 1
    ||||||| |||||||
    TATCTCGACTCTAGGCCC       Read 2
```

Errore oppure organismi diploidi

Regola 1

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

Overlap — sovrapposizione

```
TCTATATCTCGGCTCTAGG
    ||||||| |||||||
    TATCTCGACTCTAGGCCC
```

Read 1

Read 2

Probabile motivo

```
    TCTATATCTCGGCTCTAGG
GGCGTCTATATCTCGGCTCTAGGCCCTCATTTTTT
        TATCTCGACTCTAGGCCC
```

Read 1

True genome

Read 2

Errore oppure organismi diploidi

Grafo di overlap

Read

ACGTGTG CGTGTGC GTGCCA CCACG

Arco fra tutte le coppie di read con overlap abbastanza lungo

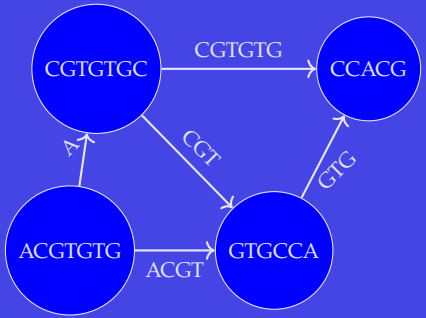
Grafo di overlap

Read

ACGTGTG CGTGTGC GTGCCA CCACG

Arco fra tutte le coppie di read con overlap abbastanza lungo

Grafo



String Graph

Read

ACGTGTG CGTGTGC GTGCCA CCACG

Si rimuovono gli archi transitivi dal grafo di overlap

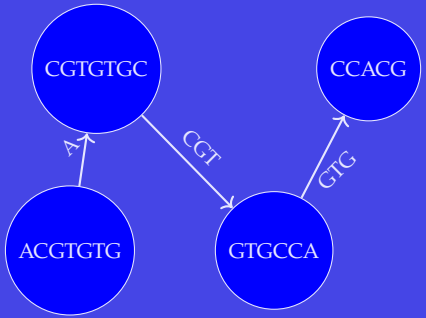
String Graph

Read

ACGTGTG CGTGTGC GTGCCA CCACG

Si rimuovono gli archi transitivi dal grafo di overlap

Grafo



Shortest superstring

Istanza

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

T è il genoma assemblato, $\mathcal{S}$ le read

Shortest superstring

Istanza

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

Soluzioni ammissibili

Superstring T di $\mathcal{S}$. Ogni s_i è sottostringa di T

T è il genoma assemblato, $\mathcal{S}$ le read

Shortest superstring

Istanza

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

Soluzioni ammissibili

Superstring T di $\mathcal{S}$. Ogni s_i è sottostringa di T

Funzione obiettivo

|T|

T è il genoma assemblato, $\mathcal{S}$ le read

Shortest superstring

Istanza

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

Soluzioni ammissibili

Superstring T di $\mathcal{S}$. Ogni s_i è sottostringa di T

Funzione obiettivo

$|T|$

T è il genoma assemblato, $\mathcal{S}$ le read

Problema

Regioni ripetute

Algoritmo ingordo

Algoritmo

Algoritmo ingordo

Algoritmo

1. Fondere le due stringhe con massimo overlap

Algoritmo ingordo

Algoritmo

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

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
3. ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
3. ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t
4. ng_time long_ti g_long_ ng_lon a_long long_l ong_lo

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
3. ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t
4. ng_time long_ti g_long_ ng_lon a_long long_l ong_lo
5. ng_time ong_lon long_ti g_long_ a_long long_l

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
3. ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t
4. ng_time long_ti g_long_ ng_lon a_long long_l ong_lo
5. ng_time ong_lon long_ti g_long_ a_long long_l
6. ong_lon long_time g_long_ a_long long_l

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
3. ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t
4. ng_time long_ti g_long_ ng_lon a_long long_l ong_lo
5. ng_time ong_lon long_ti g_long_ a_long long_l
6. ong_lon long_time g_long_ a_long long_l
7. long_lon long_time g_long_ a_long

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
3. ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t
4. ng_time long_ti g_long_ ng_lon a_long long_l ong_lo
5. ng_time ong_lon long_ti g_long_ a_long long_l
6. ong_lon long_time g_long_ a_long long_l
7. long_lon long_time g_long_ a_long
8. long_lon g_long_time a_long

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
3. ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t
4. ng_time long_ti g_long_ ng_lon a_long long_l ong_lo
5. ng_time ong_lon long_ti g_long_ a_long long_l
6. ong_lon long_time g_long_ a_long long_l
7. long_lon long_time g_long_ a_long
8. long_lon g_long_time a_long
9. long_long_time a_long

Esempio: a_long_long_long_time

1. ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
2. ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
3. ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t
4. ng_time long_ti g_long_ ng_lon a_long long_l ong_lo
5. ng_time ong_lon long_ti g_long_ a_long long_l
6. ong_lon long_time g_long_ a_long long_l
7. long_lon long_time g_long_ a_long
8. long_lon g_long_time a_long
9. long_long_time a_long
10. a_long_long_time

Problema del commesso viaggiatore (TSP)

Istanza

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

Problema del commesso viaggiatore (TSP)

Istanza

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

Soluzioni ammissibili

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

Problema del commesso viaggiatore (TSP)

Istanza

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

Soluzioni ammissibili

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

Funzione obiettivo

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

Problema del commesso viaggiatore (TSP)

Istanza

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

Soluzioni ammissibili

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

Funzione obiettivo

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

- Una soluzione è un percorso che tocca ogni città esattamente una volta e torna al punto di partenza

Problema del commesso viaggiatore (TSP)

Istanza

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

Soluzioni ammissibili

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

Funzione obiettivo

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

- Una soluzione è un percorso che tocca ogni città esattamente una volta e torna al punto di partenza
- Il costo è il peso totale di tutti gli archi attraversati

Problema del commesso viaggiatore (TSP)

Istanza

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

Soluzioni ammissibili

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

Funzione obiettivo

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

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

Superstringa più corta e TSP

Similarità

1 read = 1 città

Superstringa più corta e TSP

Similarità

1 read = 1 città

Differenze

Superstringa più corta e TSP

Similarità

1 read = 1 città

Differenze

- assemblaggio ≠ ciclo

Superstringa più corta e TSP

Similarità

1 read = 1 città

Differenze

- assemblaggio ≠ ciclo
- lunghezza stringa ≠ costo percorso TSP

Superstringa più corta e TSP

Similarità

1 read = 1 città

Differenze

- assemblaggio ≠ ciclo
- lunghezza stringa ≠ costo percorso TSP

Grafo di overlap — TSP

Read

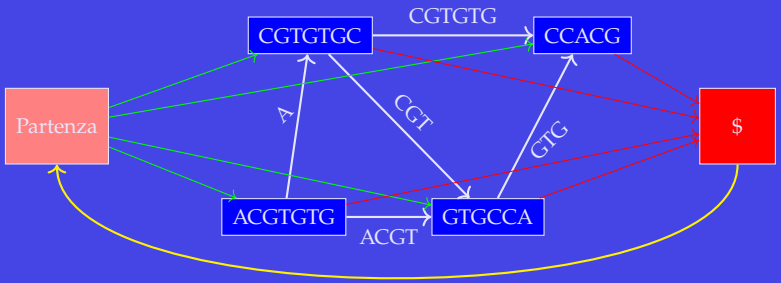
ACGTGTG CGTGTGC GTGCCA CCACG

Grafo di overlap — TSP

Read

ACGTGTG CGTGTGC GTGCCA CCACG

Grafo



Overlay — Layout — Consensus

Passi

Overlay — Layout — Consensus

Passi

1. Overlap: calcolare le sovrapposizioni e costruire il grafo. Usare suffix array (esatto) o programmazione dinamica (errori).

Overlay — Layout — Consensus

Passi

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

Overlay — Layout — Consensus

Passi

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

Reverse and complement

- Non si conosce lo strand

Reverse and complement

- Non si conosce lo strand
- Versione canonica (minima fra x e revcomp(x)

Reverse and complement

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

SBH

DNA array

SBH

DNA array

- Tecnologia vecchia

SBH

DNA array

- Tecnologia vecchia
- Per ogni k-mero, si conosce se appare nel genoma

SBH

DNA array

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

SBH

DNA array

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

Procedura

SBH

DNA array

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

Procedura

1. Ogni k-mero viene diviso in $(k-1)$-meri

SBH

DNA array

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

Procedura

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

SBH

DNA array

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

Procedura

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

SBH

DNA array

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

Procedura

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

Adesso

Stessa procedura, a partire dai read

Grafo di de Bruijn

Grafo di de Bruijn

Grafo di de Bruijn

Read

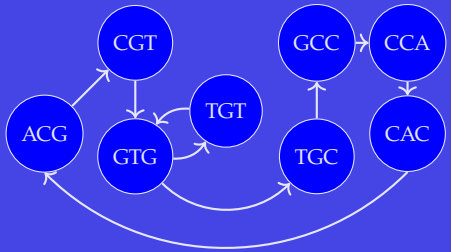
ACGTGTG CGTGTGC GTGCCA CCACG

Grafo di de Bruijn

Read

ACGTGTG CGTGTGC GTGCCA CCACG

de Bruijn graphs

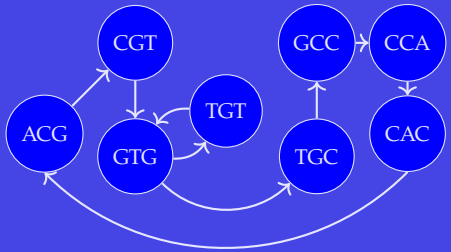


Grafo di de Bruijn

Read

ACGTGTG CGTGTGC GTGCCA CCACG

de Bruijn graphs

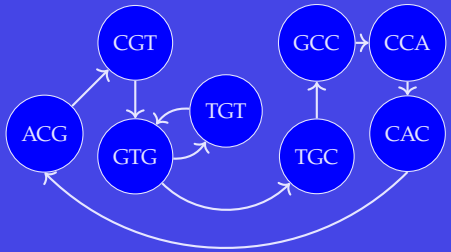


Grafo di de Bruijn

Read

ACGTGTG	CGTGTGC	GTGCCA	CCACG

de Bruijn graphs



4-meri — 3-meri — distinti

ACGT	ACG	CGT	ACG	
CACG	CAC	ACG	CAC	
CCAC	CCA	CAC	CCA	
CGTG	CGT	GTG	CGT	
GCCA	GCC	CCA	GCC	
GTGC	GTG	TGC	GTG	
GTGT	GTG	TGT		
TGCC	TGC	GCC	TGC	
TGTG	TGT	GTG	TGT	

Problemi su grafi

Ciclo Euleriano

Problemi su grafi

Ciclo Euleriano

1. Un assemblaggio valido è un cammino che attraversa ogni arco esattamente una volta

Problemi su grafi

Ciclo Euleriano

1. Un assemblaggio valido è un cammino che attraversa ogni arco esattamente una volta
2. Cammino Euleriano

Problemi su grafi

Ciclo Euleriano

1. Un assemblaggio valido è un cammino che attraversa ogni arco esattamente una volta
2. Cammino Euleriano

Ciclo Hamiltoniano

Problemi su grafi

Ciclo Euleriano

1. Un assemblaggio valido è un cammino che attraversa ogni arco esattamente una volta
2. Cammino Euleriano

Ciclo Hamiltoniano

1. É un cammino che attraversa ogni vertice esattamente una volta

Problemi su grafi

Ciclo Euleriano

1. Un assemblaggio valido è un cammino che attraversa ogni arco esattamente una volta
2. Cammino Euleriano

Ciclo Hamiltoniano

1. É un cammino che attraversa ogni vertice esattamente una volta
2. Caso particolare di TSP

Problemi su grafi

Ciclo Euleriano

1. Un assemblaggio valido è un cammino che attraversa ogni arco esattamente una volta
2. Cammino Euleriano

Ciclo Hamiltoniano

1. É un cammino che attraversa ogni vertice esattamente una volta
2. Caso particolare di TSP

Confronto

Qual è più difficile da risolvere?

Grafi Euleriani

Definizione

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

Grafi Euleriani

Definizione

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

Definizione

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

Grafi Euleriani

Definizione

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

Definizione

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

Teorema

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

Grafi Euleriani 2

Teorema

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

Grafi Euleriani 2

Teorema

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

Teorema

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

Ridurre il grafo di overlap

- Caso senza errori

Ridurre il grafo di overlap

- Caso senza errori

Osservazione 1

G grafo di overlap con $(a \to b_1)$ unico arco irriducibile uscente da a, e $(a, b_1), \ldots, (a, b_n)$ archi uscenti da a. Allora $(b_i \to b_{i+1})$ con $1 \leqslant i \leqslant n-1$ sono archi di G.

Ridurre il grafo di overlap

- Caso senza errori

Osservazione 1

G grafo di overlap con $(a \longrightarrow b_1)$ unico arco irriducibile uscente da a, e $(a, b_1), \ldots, (a, b_n)$ archi uscenti da a. Allora $(b_i \longrightarrow b_{i+1})$ con $1 \leqslant i \leqslant n-1$ sono archi di G.

Osservazione 2

G grafo di overlap con $(a \longrightarrow b_1)$ unico arco irriducibile uscente da a, e $(a, b_1), \ldots, (a, b_n)$ archi uscenti da a. Allora $(b_1 \longrightarrow b_i)$ con $2 \leqslant i \leqslant n-1$ sono archi di G.

Ridurre il grafo di overlap — algoritmo

1. b_i ordinati per lunghezza dell'arco

Ridurre il grafo di overlap — algoritmo

1. b_i ordinati per lunghezza dell'arco
2. Marcare "da eliminare" i vertici b_j tale che $(b_i \longrightarrow b_j)$ con $i < j$

Ridurre il grafo di overlap — algoritmo

1. b_i ordinati per lunghezza dell'arco
2. Marcare "da eliminare" i vertici b_j tale che $(b_i \rightarrow b_j)$ con $i < j$
3. Rimuovere gli archi che terminano in vertici da eliminare

Licenza d'uso

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

- Attribuzione — Devi attribuire la paternità dell'opera nei modi indicati dall'autore o da chi ti ha dato l'opera in licenza e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera.