# Discussione Slide Tesi

Trascrizione della discussione orale

## Slide 2: «Concetti base di biologia»

Prima di poter definire che cos’è un albero evolutivo, è necessario fornire dei concetti base di biologia, altrimenti non sarebbe possibile capire il senso di questa presentazione:

Il DNA o acido desossiribonucleico è una macromolecola contenente il patrimonio genetico degli esseri viventi, quindi ne detiene tutta l’informazione genetica ed ereditaria. Possiamo considerarla una sorta di archivio di informazioni degli esseri viventi!

La struttura è caratterizzata da una doppia elica di lunghezza variabile, composta da quattro tipi di basi azotate:

* Timina (T)
* Adenina (A)
* Guanina (G)
* Citosina (C)

Una successione di basi azotate prende il nome di sequenza.  
Un esempio di una sequenza di DNA può essere la seguente: ATGTAAGACT

***DURATA: 42 SECONDI***

## Slide 3: «Che cos’è la bioinformatica?»

Negli ultimi decenni, grazie al progresso scientifico e tecnologico, sono nate nuove discipline chiamate genericamente X-Informatics, in quanto sono il risultato dell’incontro tra l’informatica ed altre scienze di base (quali la biologia, la chimica, l’astronomia, la geologia ecc).

Tra tutte queste discipline, Quella che viene affrontata in questa presentazione è la bioinformatica. Ma che cosa è?  
La bioinformatica è un campo multidisciplinare della scienza che coinvolge la genetica, la biologia molecolare, l’informatica, la matematica e la statistica, rivolta a studiare sistemi biologici utilizzando metodi e modelli informatici e computazionali.

Tale scienza occupa numerose aree di ricerca, quella di cui ci occupiamo in questa discussione è la filogenetica, che studia le relazioni evolutive tra le entità biologiche (dagli esseri viventi fino ai virus) attraverso la costruzione di alberi evolutivi.

***DURATA: 55 SECONDI***

## Slide 4: «Albero evolutivo»

Ricordando la definizione di albero, ovvero un grafo non orientato connesso e aciclico, l’albero evolutivo (o filogenetico) è un albero che rappresenta le relazioni evolutive tra le varie entità biologiche, dove i nodi (o vertici) rappresentano tali entità, mentre gli archi mostrano le relazioni tra di loro. Nello specifico i vertici che hanno grado (num di archi incidenti al vertice) maggiore di uno, definiti nodi interni mentre quelli con grado esattamente uguale ad uno sono definite foglie.

Gli alberi evolutivi possono essere suddivisi in due tipi: alberi radicati e non radicati.

L’albero radicato o albero con radice si sviluppa a partire da un nodo speciale, chiamato radice e si estende fino alle foglie. La radice, quindi, è l’antenato comune a tutti i vertici dell’albero.

Gli alberi non radicati sono alberi senza la radice. A differenza di quelli con radice, vengono usati per mostrare le relazioni tra le entità piuttosto che mostrare l’antenato comune a tutti.

Ma come si costruiscono gli alberi evolutivi?...

***DURATA: 1 MINUTO***

## Slide 5: «Matrice delle distanze»

Attraverso i cosiddetti “*algoritmi basati sulla distanza*”, in quanto prendono in input una matrice delle distanze.

Dati due punti x e y, la distanza è una funzione d(x, y) che possiede le seguenti proprietà:

1. Non negatività [mostrare formula]
2. Identità [mostrare formula]
3. Simmetria [mostrare formula]
4. Disuguaglianza triangolare [mostrare formula]

Ma allora date unità, calcolando la distanza per ogni coppia di elementi si ottiene una *matrice delle distanze D di dimensione*

Ci sono vari modi per calcolare la distanza (ad esempi distanza euclidea, di Manhattan, ecc). In questo esempio [mostrare la matrice di esempio] abbiamo 4 ipotetiche sequenze di DNA di alcune specie. In questo caso la distanza è ottenuta contando il numero di simboli (basi azotate) differenti per ogni coppia di elementi (ad esempio la distanza tra umano e lo scimpanzé è di 3, in quanto la loro sequenza differisce di 3 simboli). Il rispettivo albero evolutivo che si ottiene viene mostrato nella slide successiva.

***DURATA: 55 SECONDI***

## Slide 6: «Problema degli alberi basati sulla distanza»

[mostrare albero]

Questo è l’albero evolutivo ottenuto dalla matrice D. Si possono notare delle proprietà importanti su tale albero:

* su ogni arco è presente un numero non negativo che prende il nome di “peso dell’arco” e rappresenta quanto sono distanti i vertici.
* Poiché tutti i vertici hanno grado diverso da 2, quindi si parla di albero semplice.
* *L’albero si adatta alla matrice D.*

Si dice che un albero T si *adatta* ad una matrice delle distanze D se per ogni i e j si ha che vale l’uguaglianza mostrata (mostrare ), ovvero l’elemento nella riga i e colonna j nella matrice D è uguale alla distanza tra le due foglie i e j nel rispettivo albero T, in tal caso sia la matrice che l’albero vengono definiti *additivi*. In caso contrario, si parla di *non additività*.

Dopo tutte queste nozioni, possiamo introdurre il problema degli alberi basati sulla distanza:

*Data in* ***input*** *una matrice delle distanze additiva restituire in* ***output*** *un albero evolutivo semplice.*

Quindi l’obiettivo degli algoritmi basati sulla distanza è quello di trovare una soluzione al problema degli alberi basati sulla distanza.

***DURATA: 1 MINUTO***

## Slide 7: «Algoritmo per il problema degli alberi basati sulla distanza» - Parte 1

Presentiamo un algoritmo per risolvere il suddetto problema!

Si prenda in considerazione la matrice delle distanze D [mostrare matrice], che è la stessa di un paio di slide fa, quindi u=umano, b=balena, ecc.

L’idea di base dell’algoritmo è che all’elemento più piccolo della matrice corrispondano due foglie vicine nel rispettivo albero. Quindi poiché è l’elemento più piccolo della matrice, possiamo supporre che siano vicini e che quindi il loro genitore sia uno generico vertice p. A questo punto bisogna calcolare la distanza tra f e p e tra b e p. Per far ciò, eseguiamo due step: si aggiunge all’albero T le foglie u ed s, tratteggiando il loro arco in quanto ancora non è possibile sapere come sono realmente collocate [mostrare albero] e si riscrivono le distanze in funzione di p [mostrare formule:  e ]. Dalle formule si può notare che l’unica incognita è . Possiamo riscriverlo in funzione degli elementi presenti nella matrice, ovvero in nel seguente modo [mostrare formula ]. Se sostituiamo questa uguaglianza alle due formule precedenti, riusciamo a trovare e .

***DURATA: 1 MINUTO E 15 SECONDI***

## Slide 8: «Algoritmo per il problema degli alberi basati sulla distanza» - Parte 2

Adesso cerchiamo e : il primo si ottiene facendo la differenza tra fu e fp (mostrare formula ], in modo analogo si trova (mostrare formula ]. L’albero risultante sarà: [mostrare albero].

Lo step successivo consiste nell’aggiornare la matrice D, quindi eliminiamo f e b ed al loro posto inseriamo il genitore p [mostrare matrice].

Rimane comunque da capire come u e s sono collocati nell’albero. Per far ciò si sceglie un generico nodo interno 𝑘 come genitore di u ed s e si applicano ricorsivamente gli step precedenti, ottenendo quindi [mostrare albero].

***DURATA: 42 SECONDI***

## Slide 9: «Algoritmo per il problema degli alberi basati sulla distanza» - Parte 3

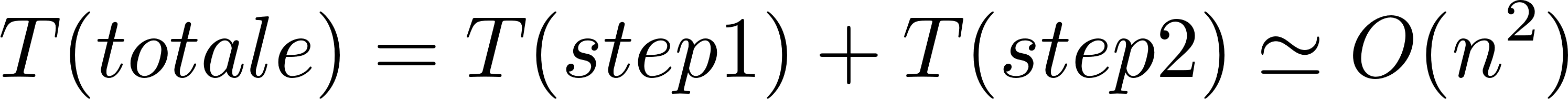
Infine rimane solo da calcolare la distanza tra k e p, che è ottenuta dalla seguente differenza [mostrare ]. L’albero finale è il seguente [mostrare albero finale!]. L’algoritmo è terminato!!

### Complessità temporale

Per calcolare la complessità nel tempo dell’algoritmo possiamo suddividerlo in 2 step:

1. Trovare il minimo in una matrice di dimensione n × n. Se si suppone l’utilizzo di un algoritmo di ricerca lineare, (che nel caso pessimo è O(n)) si ha che T(step1)=O(n^2), in quanto la matrice è di dimensione nxn.
2. Calcolare la distanza dei vertici “genitori” rispetto alle loro foglie, ed aggiornare la matrice D. Poiché vanno aggiornati i valori con tutte le altre n foglie, che sono n, allora, si ha che T(step2)=O(n)

Adesso è sufficiente sommare le due complessità e si trova quella totale dell’algoritmo:

[mostrare formula ].

***DURATA: 1 MINUTO***

## Slide 10: «Criticità»

L’algoritmo appena mostrato presenta due criticità, infatti riesce a costruire l’albero T solo se si verificano due condizioni:

* l’elemento più piccolo della matrice D deve corrispondere a due foglie vicine nel rispettivo albero T (e questo non è necessariamente vero)
* D deve essere additiva.

È necessario però fare una precisazione: non c’è modo che un albero si adatti ad una matrice non additiva, proprio per definizione di non additività, tuttavia è possibile costruire un albero che approssimi al meglio tale matrice.

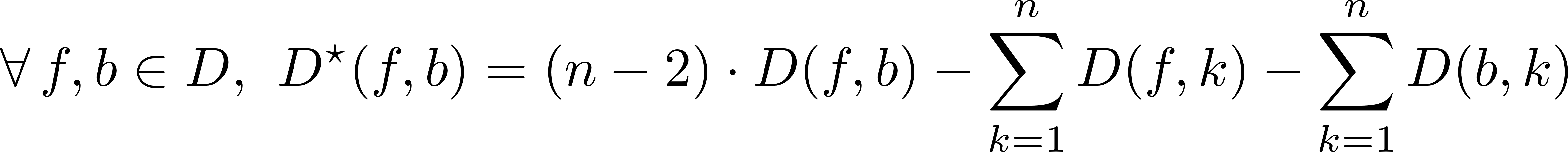
Entrambe le criticità vengono risolte dall’algoritmo **Neighbor-Joining**.

***DURATA: 40 SECONDI***

## Slide 11: «neighbor-joining» - Parte 1

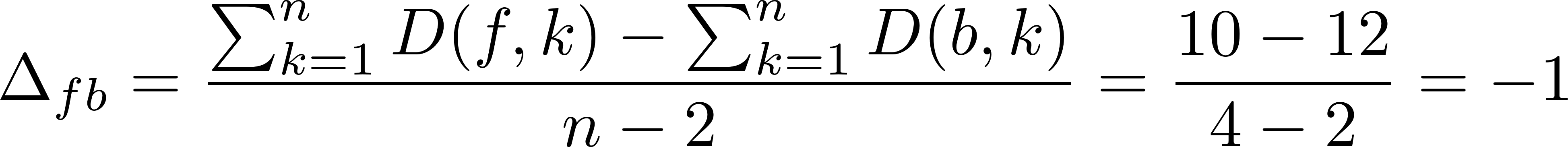
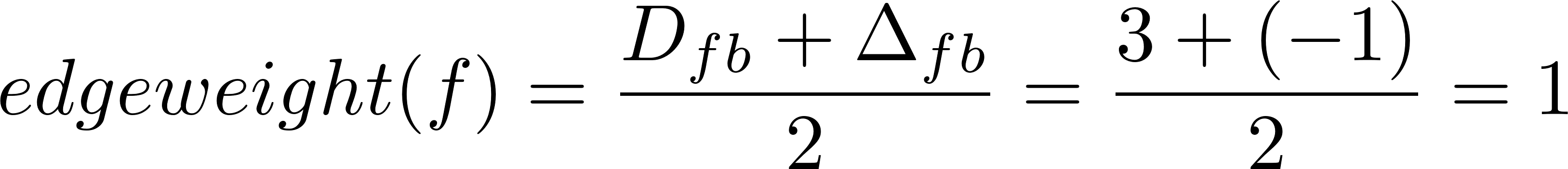
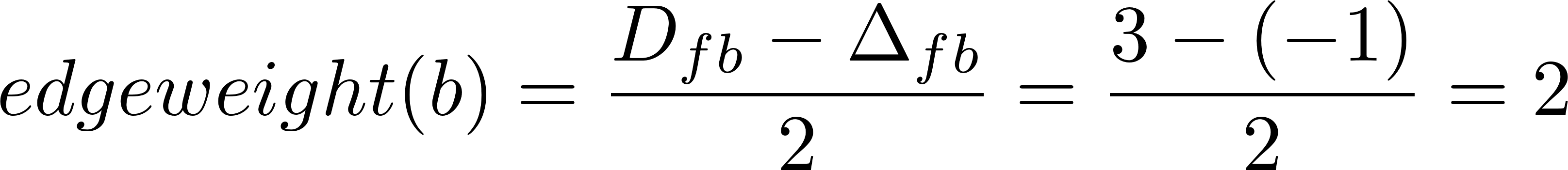
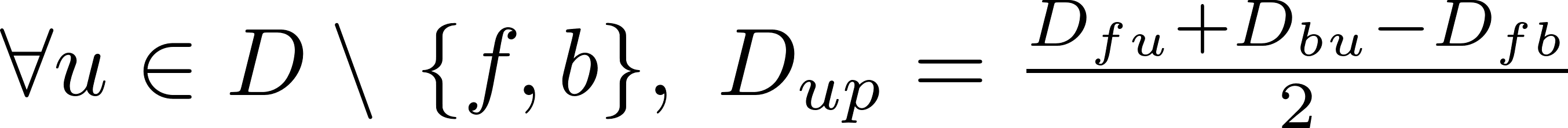
Consideriamo la seguente matrice non additiva: [mostrare matrice]. L’obiettivo è quello di costruire un albero che approssimi al meglio tale matrice!

L’algoritmo è suddiviso in più step:

1. Costruiamo la matrice D★ 🡪 Dato in input D si definisce D★ la matrice ottenuta dalla seguente formula [mostrare formula ]. Quindi, applicandola si ottiene D★ [mostrare la matrice].  
   La sua caratteristica principale è che l’elemento più piccolo *corrisponderà sempre* ad una coppia di foglie vicine nell’albero T.

***DURATA: 33 SECONDI***

## Slide 12: «neighbor-joining» - Parte 2

1. Cerchiamo l’elemento minimo in D★, ovvero D★(f,b). Quindi f e b sono foglie vicine nell’albero T e pertanto avranno un genitore p in comune.
2. calcoliamo il delta tra f e b, quindi: [mostrare formula ]...
3. … Usiamo questo delta per trovare il peso dell’arco che collega f a p e b a p: [mostrare le formule , ]
4. Aggiorniamo la matrice D, aggiungendo il genitore delle foglie, ovvero una riga ed una colonna p tale che vale la seguente uguaglianza [mostrare la formula ]. Infine, eliminiamo f e b.

...

***DURATA: 50 SECONDI***

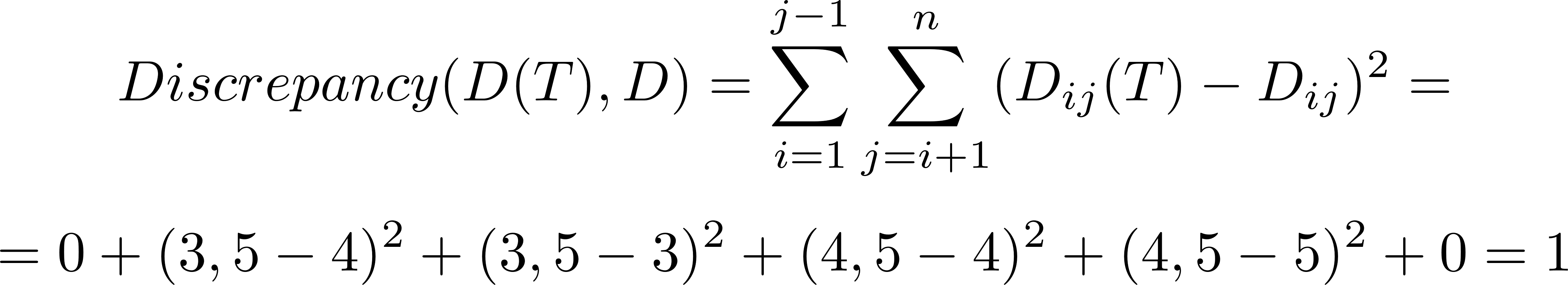
## Slide 13: «neighbor-joining» - Parte 3

… La matrice risultante è la seguente [mostrare la matrice].

Adesso seguiamo i 5 step fino a che non otteniamo una matrice 2 × 2: [mostrare la matrice]. Dalla matrice si ricava che p e k sono dei nodi interni connessi tra di loro tramite un arco di peso 1,5. Poiché abbiamo trovato tutte le altre informazioni necessarie per costruire l’albero finale, lo costruiamo, ottenendo così l’albero finale T: [mostrare albero].  
L’algoritmo è terminato!!

***DURATA: 30 SECONDI***

## Slide 14: «neighbor-joining» - Parte 4

Per capire quanto l’albero approssimi al meglio la matrice D, costruiamo la matrice D(T) che è ottenuta dall’albero [mostrare D(T)] e calcoliamo la discrepanza tra le due matrici [mostrare la formula ], otteniamo il valore 1. Il risultato mostra che non c’è una grande discrepanza tra le due matrici.

Complessità temporale:

Per calcolare la complessità possiamo individuare 2 step:

1. Crea D★ e cerca il suo elemento minimo. Poiché la matrice è nxn, allora la complessità è O(n^2)
2. Calcola il *delta*, il peso degli archi delle foglie ed aggiorna la matrice , quindi O(n).

Quindi la complessità del neighbor joining è data dalla somma dei due step, quindi O(n^2). Poiché queste operazioni vengono eseguite tante volte quante sono le foglie nella matrice (ovvero n), allora [mostrare formula].

***DURATA: 1 MINUTO***

## Slide 15: « Unweighted Pair Group Method with Arithmetic Mean» - Parte 1

Fino ad ora abbiamo visto algoritmi che costruiscono alberi senza radice. L’algoritmo UPGMA invece, prende in input una matrice delle distanze additiva o non additiva e restituisce un albero radicato in cui tutte le foglie sono alla stessa distanza dalla radice, definito anche albero ultrametrico.

Tale albero possiede delle caratteristiche, ovvero:

* Le foglie sono le entità biologiche attualmente esistenti;
* I nodi interni rappresentano le speciazioni, quei processi attraverso le quali si formano nuove specie.
* Ogni vertice ha associato un numero non negativo che rappresenta l’età del vertice…
* … quindi il peso degli archi è dato dalla differenza tra le età dei nodi;

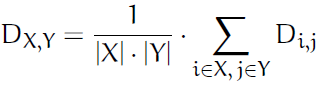
L’algoritmo UPGMA può essere diviso in più fasi.

Si prenda in considerazione la seguente matrice non additiva [mostrare matrice] (da notare che è la stessa usata nel NJ).

1. A partire da creiamo un cluster per foglia. Nel nostro caso ne creiamo 4 [mostrare immagine]

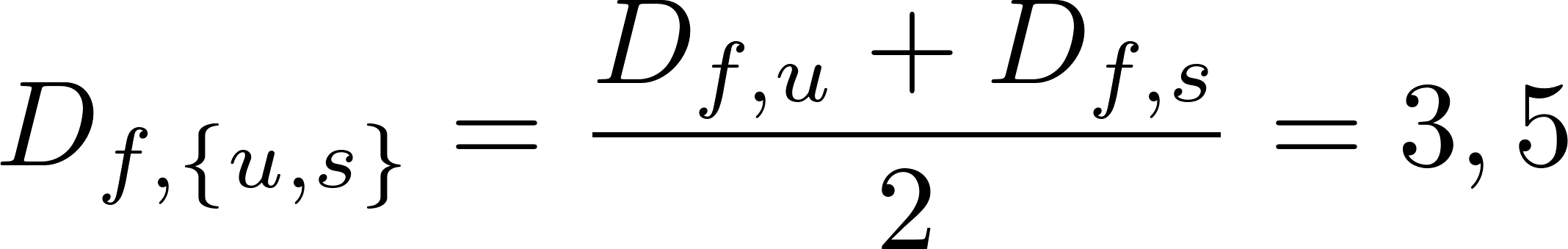
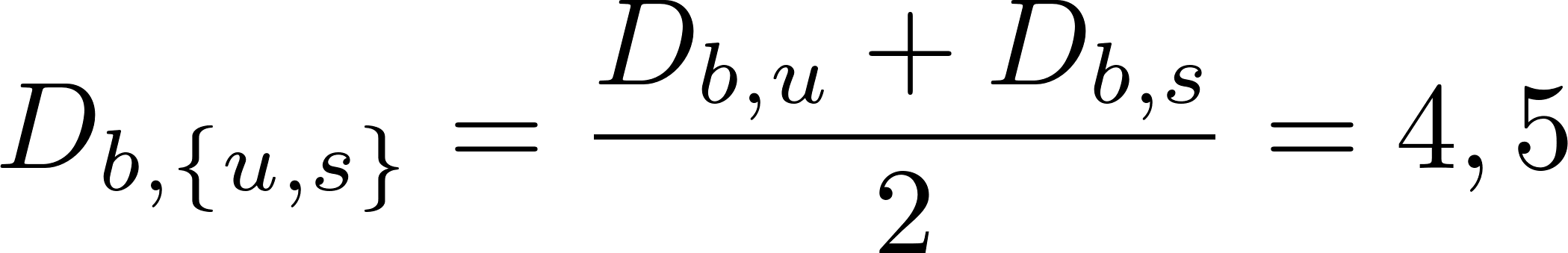
***DURATA: 1 MINUTO***

## Slide 16: « Unweighted Pair Group Method with Arithmetic Mean» - Parte 2

1. Scegliamo i due cluster X e Y più vicini secondo la seguente definizione di distanza [mostrare formula ]. Poiché in questo caso ciascun cluster è formato da un solo elemento, prendere i due cluster più vicini equivale a scegliere l’elemento più piccolo in D, ovvero D(u,s)=2. Quindi U ed S sono vicini.
2. Creiamo un cluster {u,s} che è dato dall’unione tra i cluster u e s.
3. Creiamo in un nodo interno per il cluster {u,s}, calcoliamo la sua età ed il peso degli archi di ed .Quindi l’età del cluster è 1, ed anche il peso degli archi che lo collegano con u ed s. A questo punto l’albero risultante è il seguente [mostrare albero].

***DURATA: 50 SECONDI***

## Slide 17: « Unweighted Pair Group Method with Arithmetic Mean» - Parte 3

1. Aggiorniamo 𝐷, quindi eliminiamo u e s ed al loro posto aggiungiamo il cluster {u,s}, calcolando la distanza tra esso e gli altri elementi presenti nella matrice usando la formula dello step 2. Nel nostro caso: [mostra le formule. Prima , poi , ed infine la matrice]

Adesso eseguiamo gli step mostrati fino a che non otteniamo una matrice di dimensione 2×2: [mostrare la matrice]. L’ultima coppia rimasta, ovvero {f,b,u,s}, è il cluster contenente tutte le specie, quindi è la radice nell’albero. L’albero finale è il seguente: [mostrare albero].

L’algoritmo è terminato!

***DURATA: 45 SECONDI***

## Slide 18: « Unweighted Pair Group Method with Arithmetic Mean» - Parte 4

Complessità temporale:

Ad ogni iterazione vengono effettuate una serie di operazioni, tra cui aggiornare 𝐷 calcolando la distanza tra il cluster appena inserito e gli altri elementi, quindi la complessità è 𝑂(𝑛).

Queste operazioni vengono iterate 𝑛−2 volte, ovvero fino a che non si ottiene una matrice 2×2, quindi 𝑂(𝑛−2)

Allora la complessità totale dell’algoritmo sarà la seguente: .

Con quest’ultima slide la discussione della tesi è terminata!!

***DURATA: 30 SECONDI***

***TOTALE 855 SECONDI!***