



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Applicazioni dell'algoritmica alla biologia: alberi evolutivi

Matteo Tortoli

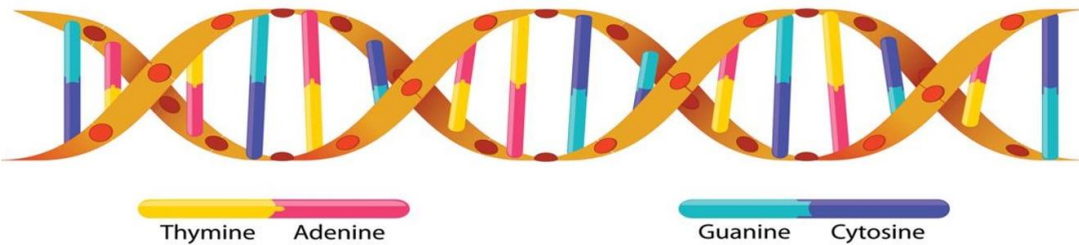
Relatrice Prof.ssa Maria Cecilia Verri

Firenze, 12 luglio 2019

Concetti base di biologia

DNA ed allineamento di sequenze

Il **DNA** o **acido desossiribonucleico** è una macromolecola contenente il patrimonio genetico degli esseri viventi.




- Struttura a doppia elica di lunghezza variabile;
- 4 tipi di basi azotate:
 - Timina (T)
 - Adenina (A)
 - Guanina (G)
 - Citosina (C)

Una successione di basi azotate prende il nome di **sequenza**.

Esempio di una sequenza di DNA → ATGTAAGACT

Che cos'è la bioinformatica?

X-Informatics  il risultato dell'incontro tra l'informatica ed altre scienze di base, quali la biologia, la chimica, l'astronomia ecc.

Bioinformatica

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.

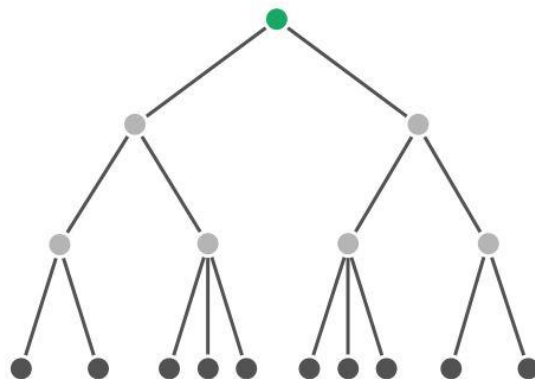
Filogenetica  Area di ricerca che studia le relazioni evolutive tra le entità biologiche attraverso la costruzione di **alberi evolutivi** (chiamati anche **alberi filogenetici**).

Albero Evolutivo

È un diagramma che rappresenta le relazioni evolutive tra le entità biologiche, dove i nodi (o vertici) rappresentano tali entità, mentre gli archi mostrano loro relazioni.

Due tipi di albero

Albero radicato (o con radice)



Ancestor
TIME
Present Day

- Nodo speciale, chiamato radice;
 - Vertici (nodi interni) con grado maggiore di 1 sono gli antenati;
 - Vertici (foglie) con grado 1 sono le specie attualmente esistenti;
- Quindi la radice è l'antenato comune a tutti i vertici.

Albero non radicato (o senza radice)

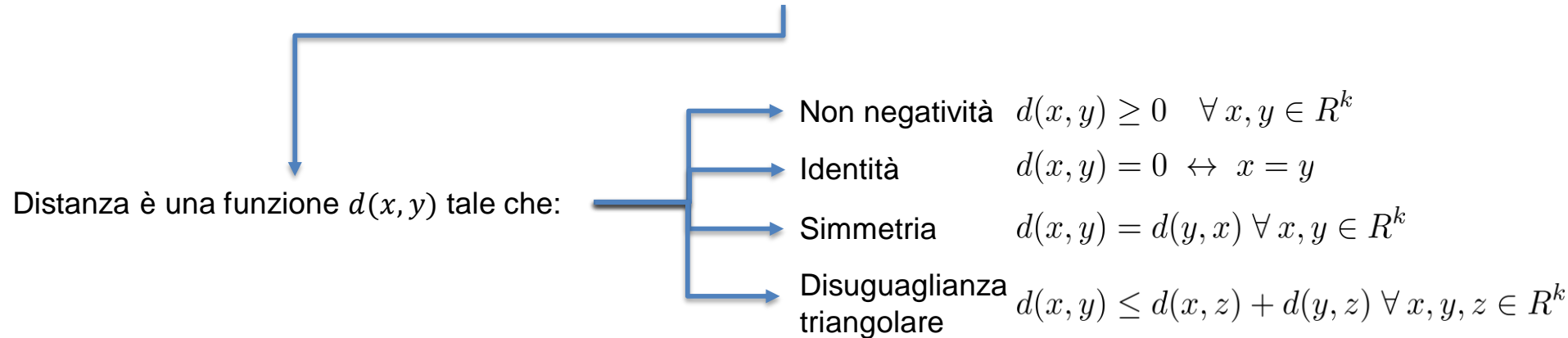


- Alberi senza la radice;
 - Vertici con grado maggiore di 1 sono i nodi interni;
 - Vertici con grado 1 sono dette foglie;
- Usati per mostrare le relazioni tra le entità piuttosto che mostrare l'antenato comune a tutti.

Come si costruiscono gli alberi evolutivi?

Matrice delle distanze

Gli algoritmi utilizzati per la costruzione degli alberi evolutivi prendono il nome di *algoritmi basati sulla distanza*, in quanto prendono in input una *matrice delle distanze*.



Date n unità, calcolando la distanza per ogni coppia di elementi si ottiene una *matrice D delle distanze* di *dimensione $n \times n$* .

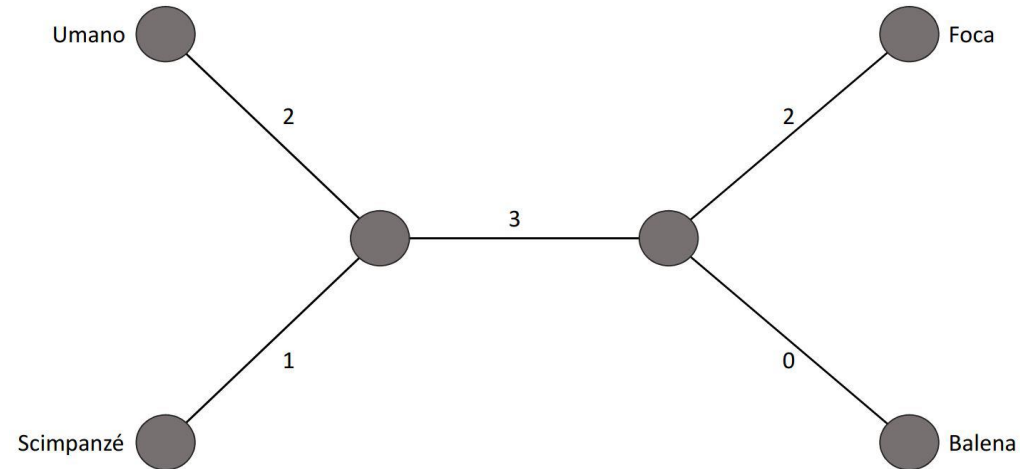
Esempio:

SPECIE	ALLINEAMENTO	MATRICE DELLE DISTANZE			
		Umano	Scimpanzé	Foca	Balena
Umano	ATGTAAGACT	0	3	7	5
Scimpanzé	ACGTAGGCCT	3	0	6	4
Foca	TCGAGAGCAC	7	6	0	2
Balena	TCGAAAGCAT	5	4	2	0

Problema degli alberi basati sulla distanza

Proprietà dell'albero:

- Numero non negativo su ogni arco rappresenta la distanza tra le foglie;
- *Distanza evolutiva* tra due entità biologiche i e $j \rightarrow$ somma del peso degli archi che collegano i e j ;
- Tutti i vertici hanno grado diverso da 2 \rightarrow *Albero semplice*;
- L'albero si *adatta* alla matrice D .



Un albero T si *adatta* ad una matrice delle distanze D se $\rightarrow \forall i, j \in V, D_{ij} = d_{ij}(T)$

Sia T che D si definiscono *additivi*, altrimenti si parla di *non additività*.

Problema degli alberi basati sulla distanza:

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

Obiettivo degli algoritmi basati sulla distanza \rightarrow Risolvere il problema degli alberi basati sulla distanza

Algoritmo per il problema degli alberi basati sulla distanza

Parte 1

Matrice in input

→ $D =$

specie	u	s	f	b
u	0	3	7	5
s	3	0	6	4
f	7	6	0	2
b	5	4	2	0

$\min \rightarrow D_{fb} = 2$

Come trovare d_{fp} e d_{bp} ?
Si aggiunge u ed s a T e si scrivono le distanze in funzione di p .

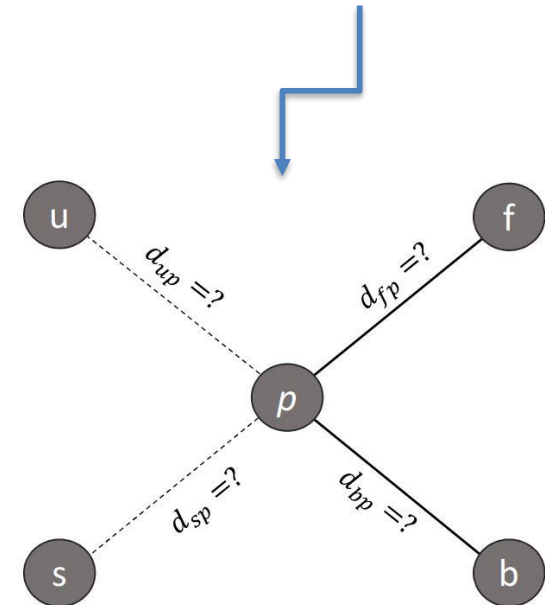
Sostituendo d_{up}
alle due formule
si ottiene che
 $D_{fp} = 2$ e
 $D_{bp} = 0$

$$d_{up} = \frac{d_{fu} + d_{bu} - d_{fb}}{2}$$

Come trovare d_{up} ?

$$d_{fp} = d_{fu} - d_{up}$$

$$d_{bp} = d_{bu} - d_{up}$$

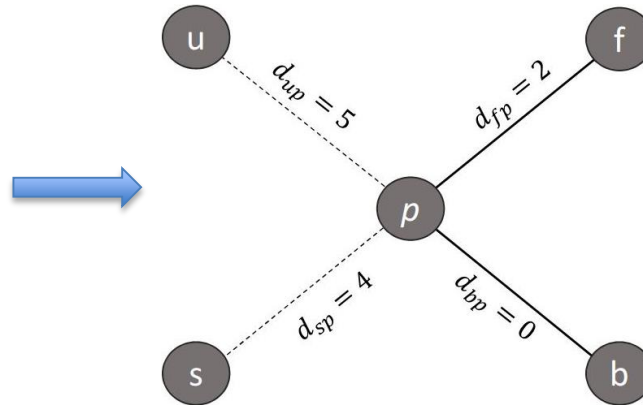


Algoritmo per il problema degli alberi basati sulla distanza

Parte 2

$$d_{up} = d_{fu} - d_{fp} = 5$$

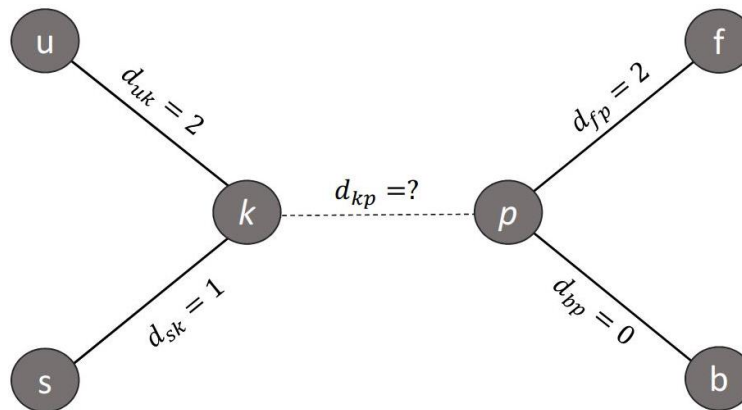
$$d_{sp} = d_{bs} - d_{bp} = 4$$



Aggiorniamo D : si elimina f e b ed al loro posto si inserisce p .

$D =$

specie	u	s	p
u	0	3	5
s	3	0	4
p	5	4	0



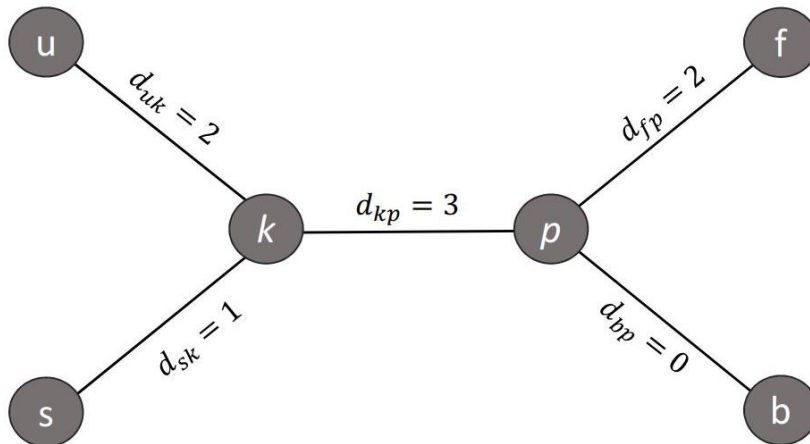
Il genitore di u e s ? Si sceglie un generico nodo interno k e si applicano ricorsivamente gli step precedenti

Algoritmo per il problema degli alberi basati sulla distanza

Parte 3

Infine si calcola d_{kp} : $d_{kp} = d_{up} - d_{uk} = 5 - 2 = 3$

Albero finale



Complessità temporale

3 Step:

- Trovare il minimo in D di dimensione $n \times n$:

$$T(\text{step1}) = O(n) \times O(n) = O(n^2)$$
- Trovare il genitore per ogni coppia di foglie e calcolare la distanza di tutte le n foglie rispetto al genitore stesso

$$T(\text{step2}) = O(n) + O(1) \simeq O(n)$$

- Calcolare la distanza tra le foglie interne (genitori)

$$T(\text{step3}) = O(n/2)$$

$$T(\text{totale}) = T(\text{step1}) + T(\text{step2}) + T(\text{step3}) \simeq O(n^2)$$

L'algoritmo è terminato!

Albero Additivo

Parte 1

Criticità → l'elemento più piccolo della matrice D deve corrispondere a due foglie vicine nell'albero T .



Possibile soluzione → Invece di cercare le foglie vicine in T , aggiungerle all'albero una alla volta.



Nuovo problema → Calcolare il peso degli archi che collegano le foglie con i rispettivi genitori (*arti*).



Teorema del peso degli arti:

Sia $limbweight(j)$ il peso dell'arto di j . Data una matrice delle distanze additiva D ed una foglia j , $limbweight(j)$ è uguale al valore minimo di $\frac{D_{j,i} + D_{j,k} - D_{i,k}}{2}$ tra tutte le foglie i e k .

Algoritmo «albero additivo»

Matrice in input



$D =$

specie	f	b	u	s
f	0	13	21	22
b	13	0	12	13
u	21	12	0	13
s	22	13	13	0

Albero Additivo

Parte 2

Step 1 → Se D contiene due elementi, restituisci l'albero costituito da un arco che collega le due foglie. Altrimenti step 2;

Step 2 → Scegli b e calcola $limbweight(b)$ usando il teorema del peso degli arti: $\longrightarrow limbweight(b) = 2$

Step 3 → Sottrai $limbweight(b)$ nella riga e colonna b (esclusa la diagonale):

$$\longrightarrow D = \begin{array}{c|cccc} & \text{specie} & f & b & u & s \\ \hline f & & 0 & 11 & 21 & 22 \\ b & & 11 & 0 & 10 & 11 \\ u & & 21 & 10 & 0 & 13 \\ s & & 22 & 11 & 13 & 0 \end{array}$$

Step 4 → Applica nuovamente il teorema del peso degli arti, con $limbweight(b) = 0$.

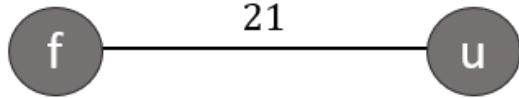
$$limbweight(b) = \frac{D_{f,b} + D_{b,u} - D_{f,u}}{2} \rightarrow limbweight(b) = 0, \text{ quindi } \rightarrow 0 = \frac{D_{f,b} + D_{b,u} - D_{f,u}}{2} \rightarrow$$

$$D_{f,u} = D_{f,b} + D_{b,u} \longrightarrow \text{La foglia } b \text{ è lungo l'arco che collega } f \text{ con } u.$$

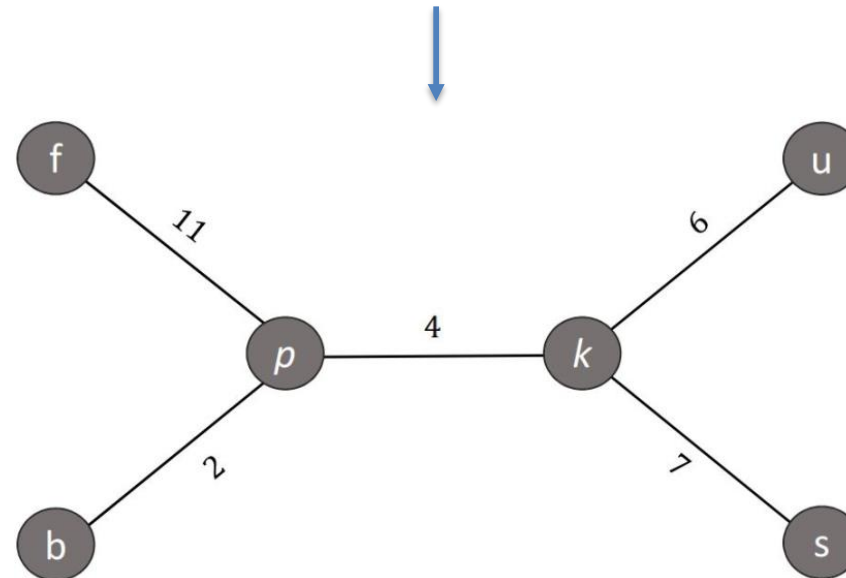
Albero Additivo

Parte 3

Step 5 → Rimuovi b da D e *riesegui tutti gli step* fino a che non si ottiene una matrice 2×2 ;

Step 6 → Costruisci T a partire da D di dimensione 2×2 $\longrightarrow T \rightarrow$ 

Step 7 → Inserisci di volta in volta le foglie in T , dove il peso del loro arto è *limbweight*. Il risultato di questa operazione sarà l'albero evolutivo completo.



L'algoritmo è terminato!

Albero Additivo

Parte 4 – Complessità temporale

3 Step:

- calcola il peso dell'arto dell' n – *esima* foglia;
- aggiorna la riga e la colonna n in D ;
- Individua il punto in cui n va inserita in T

Matrice di dimensione $n \times n$ quindi
 $T(\text{AlberoAdditivo}) = O(n^2)$

Viene eseguito n volte, quindi $T(\text{totale}) = O(n) \times T(\text{AlberoAdditivo}) = O(n) \times O(n^2) = O(n^3)$

Criticità \rightarrow *AlberoAdditivo* non riesce a costruire T se D non è additiva.

Per definizione di non addittività, non c'è modo che un albero si adatti ad una matrice non additiva.

In tal caso possiamo costruire un albero T che approssimi D .

Algoritmo **Neighbor-Joining**.

Neighbor-Joining

Parte 1

Matrice non
additiva in input

→ $D =$

specie	f	b	u	s
f	0	3	4	3
b	3	0	4	5
u	4	4	0	2
s	3	5	2	0

→

Obiettivo: costruire T che approssimi
al meglio le distanze tra le foglie in D .

- Costruisci la matrice $D^* \rightarrow$ Data in input D si definisce D^* la seguente matrice:

$$\forall f, b \in D, D^*(f, b) = (n - 2) \cdot D(f, b) - \sum_{k=1}^n D(f, k) - \sum_{k=1}^n D(b, k)$$

↓

specie	f	b	u	s
f	0	-16	-12	-14
b	-16	0	-14	-12
u	-12	-14	0	-16
s	-14	-12	-16	0

↓

L'elemento più **piccolo** in D^* corrisponde ad una coppia di foglie **vicine** nell'albero T .

Neighbor-Joining

Parte 2

2. Cerca l'elemento minimo in $D^* \rightarrow D_{fb}^* = -16$.
3. Calcola il *delta* tra $totalDistance(D_f)$ e $totalDistance(D_b)$.

$$\Delta_{fb} = \frac{totalDistance(D_f) - totalDistance(D_b)}{n - 2} = \frac{10 - 12}{4 - 2} = -1$$

4. Calcola $limbweight(f)$ e $limbweight(b)$

$$limbweight(f) = \frac{D_{fb} + \Delta_{fb}}{2} = \frac{3 + (-1)}{2} = 1$$

$$limbweight(b) = \frac{D_{fb} - \Delta_{fb}}{2} = \frac{3 - (-1)}{2} = 2$$

5. Aggiorna la matrice $D \rightarrow$ Aggiungi il genitore non noto di f e b in D , ovvero una riga ed una colonna p tale che:

$$\forall u \in D \setminus \{f, b\}, D_{up} = \frac{D_{fu} + D_{bu} - D_{fb}}{2}$$

Infine si eliminano f e b da D .

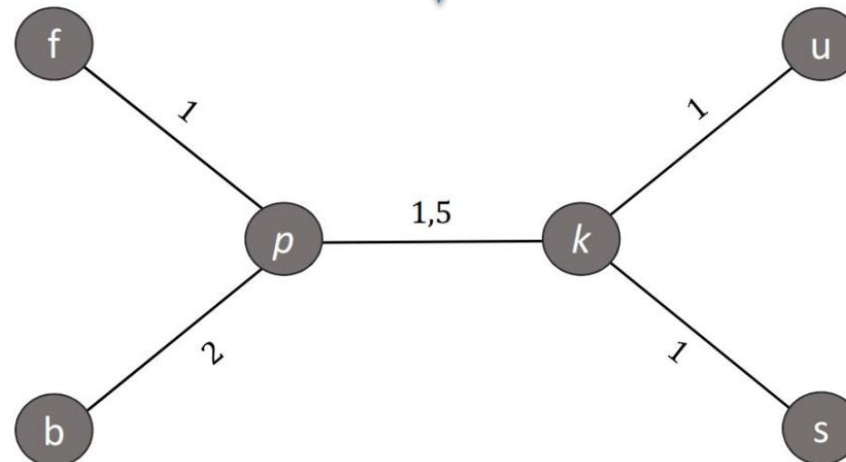
Neighbor-Joining

Parte 3

La matrice risultante è $\rightarrow D = \begin{matrix} & \text{specie} & & \\ & p & u & s \\ p & \begin{pmatrix} 0 & 2,5 & 2,5 \end{pmatrix} \\ u & \begin{pmatrix} 2,5 & 0 & 2 \end{pmatrix} \\ s & \begin{pmatrix} 2,5 & 2 & 0 \end{pmatrix} \end{matrix} \rightarrow \text{Esegui gli step fino a che non ottieni una matrice } 2 \times 2.$

p e k sono nodi interni collegati da un arco di peso 1,5.
Adesso si può costruire l'albero finale T .

$$D = \begin{matrix} & \text{specie} & & \\ & p & k \\ p & \begin{pmatrix} 0 & 1,5 \end{pmatrix} \\ k & \begin{pmatrix} 1,5 & 0 \end{pmatrix} \end{matrix}$$



L'algoritmo è terminato!

Neighbor-Joining

Parte 4

Per capire quanto T approssimi al meglio D si costruisce $D(T)$ e si calcola la discrepanza tra D e $D(T)$. Quindi:

$$D(T) = \begin{matrix} & \begin{matrix} \text{specie} & f & b & u & s \end{matrix} \\ \begin{matrix} f \\ b \\ u \\ s \end{matrix} & \begin{pmatrix} 0 & 3 & 3,5 & 3,5 \\ 3 & 0 & 4,5 & 4,5 \\ 3,5 & 4,5 & 0 & 2 \\ 3,5 & 4,5 & 2 & 0 \end{pmatrix} \end{matrix} \longrightarrow \begin{aligned} \text{Discrepancy}(D(T), D) &= \sum_{i=1}^{j-1} \sum_{j=i+1}^n (D_{ij}(T) - D_{ij})^2 = \\ &= 0 + (3,5 - 4)^2 + (3,5 - 3)^2 + (4,5 - 4)^2 + (4,5 - 5)^2 + 0 = 1 \end{aligned}$$

Il risultato mostra che c'è poca discrepanza tra le due matrici.

Complessità Temporale

2 step:

- Crea D^* e cerca l'elemento minimo $\longrightarrow T(\text{step1}) = O(n) \times O(n) = O(n^2)$
- Calcola il *delta*, il peso degli arti ed infine aggiorna la matrice $D \longrightarrow T(\text{step2}) = O(n)$

$$T(\text{NeighborJoining}) = T(\text{step1}) + T(\text{step2}) = O(n^2) + O(n) = O(n^2)$$

Eseguito tante volte quante sono le foglie in D , quindi n volte

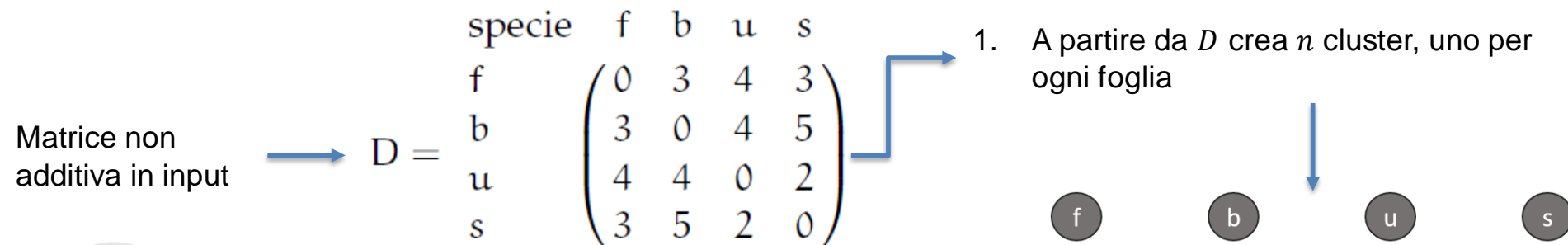
$$T(\text{Totale}) = T(\text{NeighborJoining}) \times O(n) = O(n^3)$$

Unweighted Pair Group Method with Arithmetic Mean

Parte 1

UPGMA (Unweighted Pair Group Method with Arithmetic Mean) → data in input una matrice delle distanze D *additiva* o *non additiva*, restituisce un albero *radicato* T in cui tutte le foglie sono alla stessa distanza dalla radice (*albero ultrametrico*).

- Ogni vertice ha associato un numero non negativo → età del vertice;
- Peso degli archi → differenza tra le età dei nodi;
- Foglie → entità biologiche attualmente esistenti;
- Nodi interni → Speciazioni (processi in cui si formano nuove specie).



Unweighted Pair Group Method with Arithmetic Mean

Parte 2

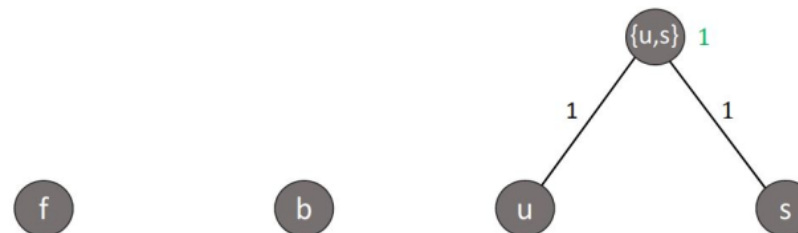
- Scegli i due cluster X e Y più vicini secondo la seguente definizione di distanza: $D_{X,Y} = \frac{1}{|X| \cdot |Y|} \cdot \sum_{i \in X, j \in Y} D_{i,j} \longrightarrow D_{u,s} = 2$
- Crea un cluster Z che è dato dall'unione di X ed $Y \rightarrow \{u, s\} = \{u\} \cup \{s\}$
- Crea in T un nodo interno per il cluster Z , calcola la sua età ($age(Z) = \frac{D_{X,Y}}{2}$) ed il peso degli archi di x ed y

$$age(\{u, s\}) = \frac{D_{u,s}}{2} = 1$$

$$edgeweight(\{u, s\}, s) = age(\{u, s\}) - age(s) = 1$$

$$edgeweight(\{u, s\}, u) = age(\{u, s\}) - age(u) = 1$$

L'albero risultante:



Unweighted Pair Group Method with Arithmetic Mean

Parte 3

5. Aggiorna $D \rightarrow$ calcola la distanza tra Z e gli altri elementi presenti in D usando la formula dello step 2

$$D_{f,\{u,s\}} = \frac{D_{f,u} + D_{f,s}}{2} = 3,5$$

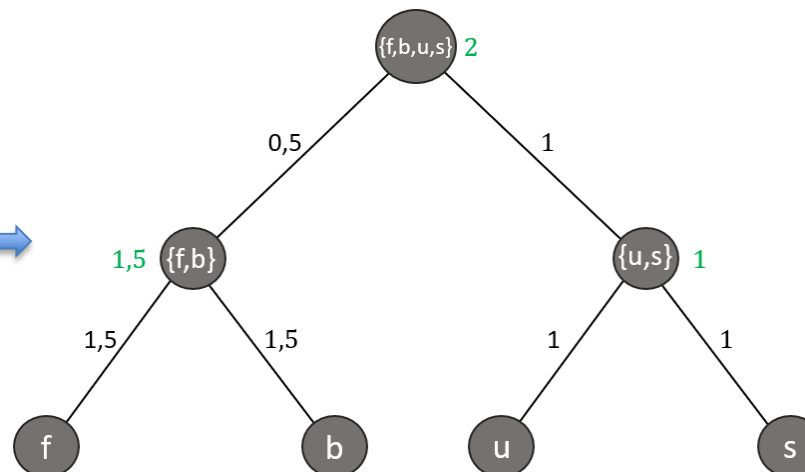
$$D_{b,\{u,s\}} = \frac{D_{b,u} + D_{b,s}}{2} = 4,5$$

$$D = \begin{matrix} & \text{specie} & f & b & \{u,s\} \\ \begin{matrix} f \\ b \\ \{u,s\} \end{matrix} & \begin{pmatrix} 0 & 3 & 3,5 \\ 3 & 0 & 4,5 \\ 3,5 & 4,5 & 0 \end{pmatrix} \end{matrix}$$

Esegui gli step fino a che non ottieni una matrice di dimensione 2×2 .

$$D = \begin{matrix} & \text{specie} & \{f,b\} & \{u,s\} \\ \begin{matrix} \{f,b\} \\ \{u,s\} \end{matrix} & \begin{pmatrix} 0 & 4 \\ 4 & 0 \end{pmatrix} \end{matrix} \rightarrow \text{Il cluster } \{f,b,u,s\} \text{ contiene tutte le specie ed è la radice in } T.$$

Albero finale T



L'algoritmo è terminato!

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 $\rightarrow O(n)$.

Queste iterazioni vengono fatte $n - 2$ volte, ovvero fino a che non si ottiene una matrice $2 \times 2 \rightarrow O(n - 2)$


$$T(Totale) = T(UPGMA) \times O(n - 2) \simeq O(n^2)$$

La discussione è terminata!

