



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Applicazioni dell'algoritmica alla biologia: alberi evolutivi

Matteo Tortoli

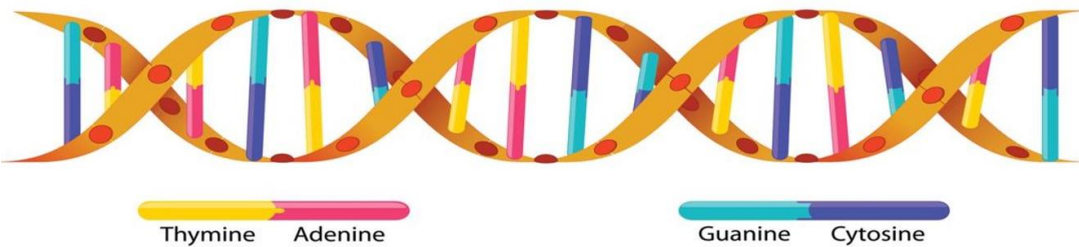
Relatrice Prof.ssa Maria Cecilia Verri

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.

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.

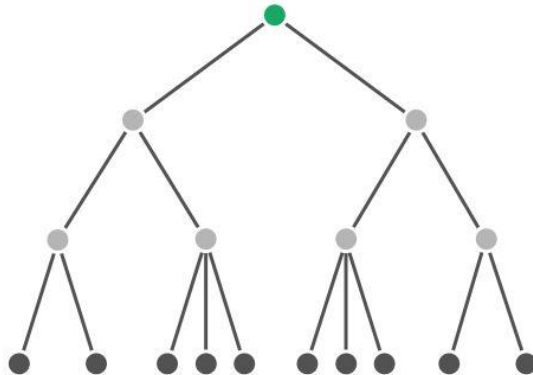
Filogenetica  Studia le relazioni evolutive tra le entità biologiche attraverso la costruzione di **alberi evolutivi** (o **alberi filogenetici**).

Albero Evolutivo

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

Due tipi di albero

Albero radicato



- Nodo speciale chiamato radice

Albero non radicato



- Albero senza la radice



Come si costruiscono gli alberi evolutivi?

Matrice delle distanze

Algoritmi basati sulla distanza: prendono in input una *matrice delle distanze*.

Distanza è una funzione $d(x, y)$ tale che:

Non negatività $d(x, y) \geq 0 \quad \forall x, y \in R^k$

Identità $d(x, y) = 0 \leftrightarrow x = y$

Simmetria $d(x, y) = d(y, x) \quad \forall x, y \in R^k$

Disuguaglianza triangolare $d(x, y) \leq d(x, z) + d(y, z) \quad \forall x, y, z \in R^k$

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

Esempio:

SPECIE

ALLINEAMENTO

MATRICE DELLE DISTANZE

Umano
Scimpanzé
Foca
Balena

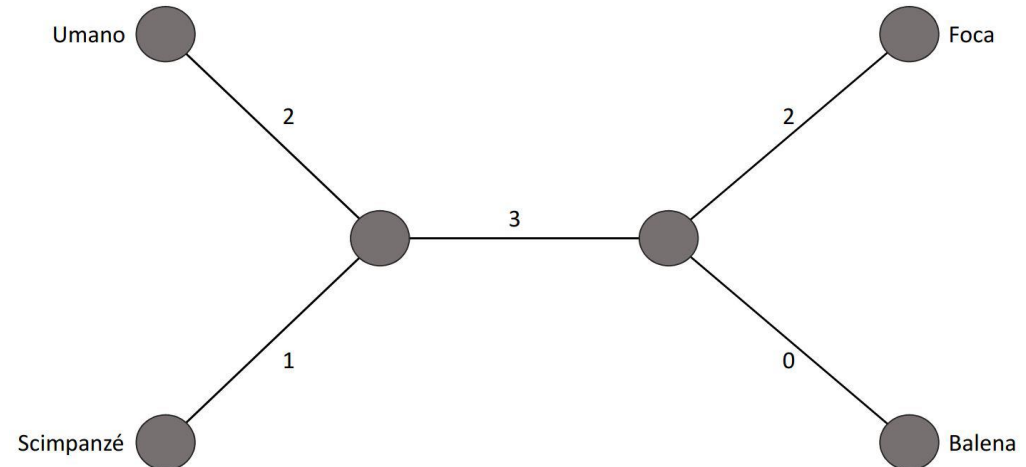
ATGTAAGACT
ACGTAGGCCT
TCGAGAGCAC
TCGAAAGCAT

Umano	Scimpanzé	Foca	Balena
0	3	7	5
3	0	6	4
7	6	0	2
5	4	2	0

Problema degli alberi basati sulla distanza

Proprietà dell'albero:

- Numero non negativo su ogni arco \rightarrow distanza tra i nodi;
- Tutti i vertici hanno grado diverso da 2 \rightarrow *Albero semplice*;
- L'albero si *adatta* alla matrice.



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

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 \longrightarrow Risolvere il problema degli alberi basati sulla distanza

Algoritmo per il problema degli alberi basati sulla distanza

Parte 1

Matrice in input $\rightarrow D =$

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

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

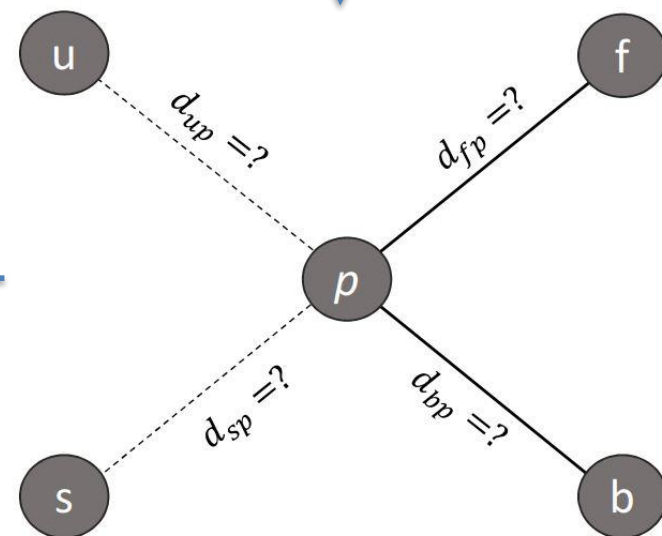
d_{fp} e d_{bp} ? Aggiungi u ed s a T e scrivi le distanze in funzione di p .

$d_{up}?$

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

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

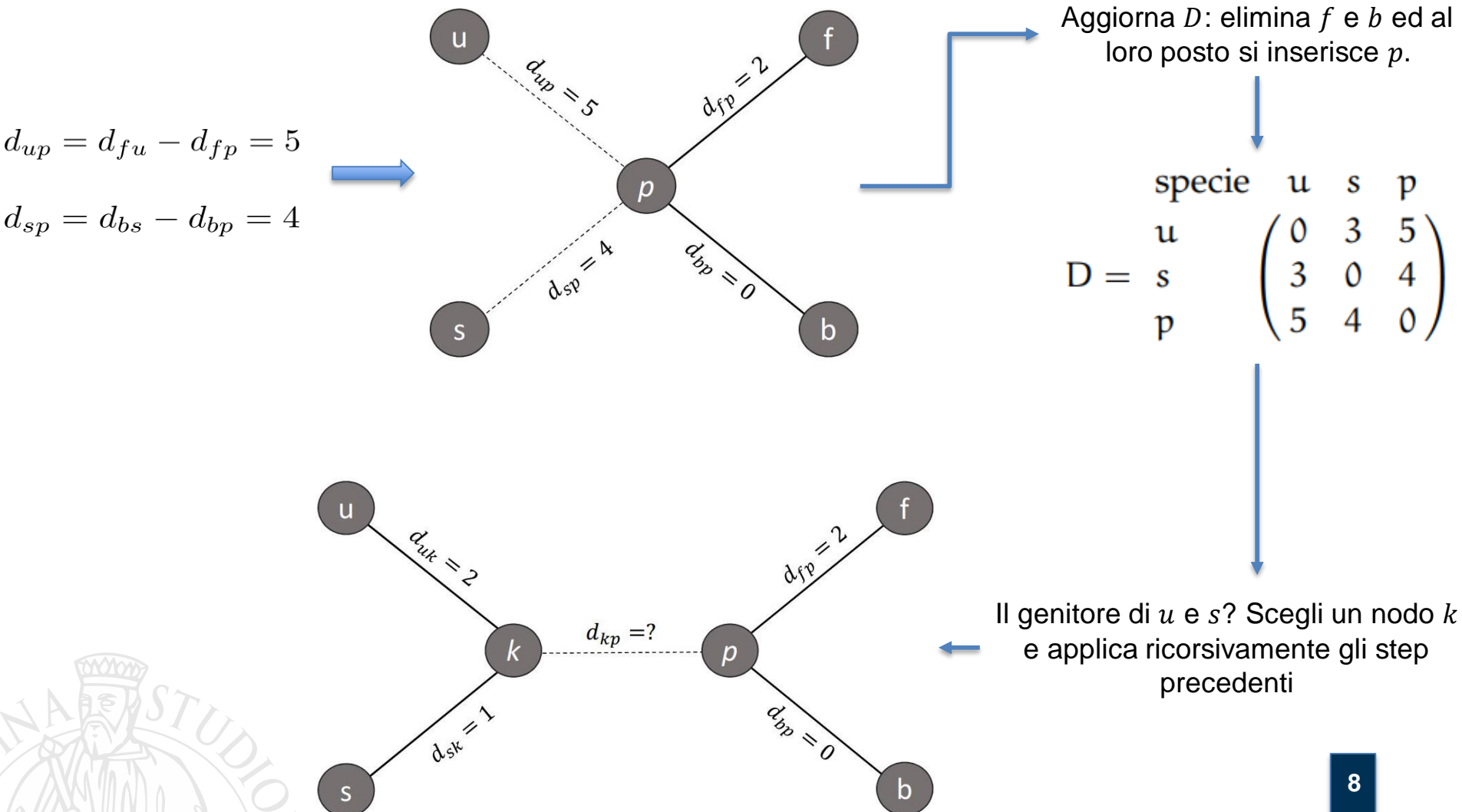
$$d_{bp} = d_{bu} - d_{up}$$



Si ottiene che $d_{fp} = 2$ e $d_{bp} = 0$

Algoritmo per il problema degli alberi basati sulla distanza

Parte 2

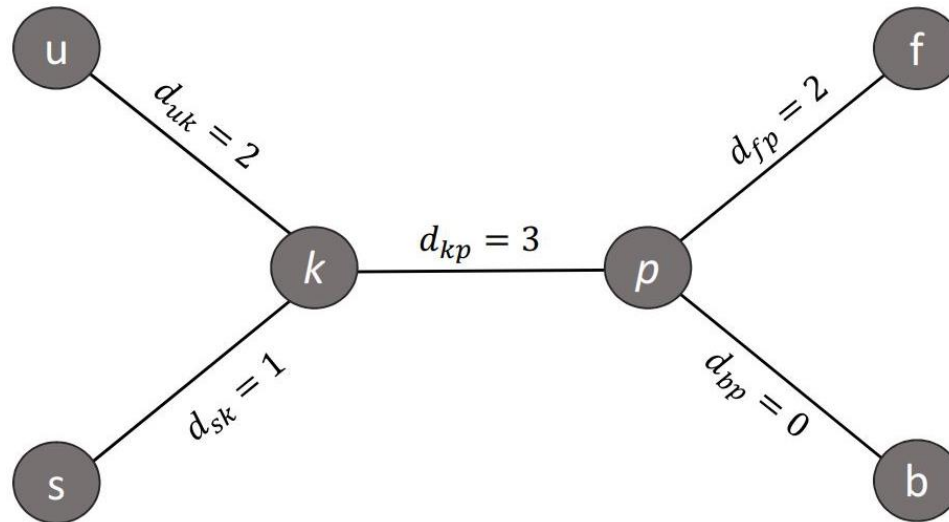


Algoritmo per il problema degli alberi basati sulla distanza

Parte 3

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

Albero finale



Complessità temporale

2 Step:

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

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

- Calcolare la distanza dei vertici genitori ed aggiornare la matrice D .

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

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

L'algoritmo è terminato!

Criticità dell'algoritmo

L'algoritmo costruisce T se e solo se:

- L'elemento più piccolo in D corrisponde a due foglie vicine in T ;
 - D è *additiva*.



Per definizione di non additività, 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 non additiva.



Algoritmo che risolve entrambe le criticità: **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 D .

1. Costruisci la matrice $D^* \rightarrow$ Data in input D si definisce D^* :

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

↓

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 f e b . $\longrightarrow \Delta_{fb} = \frac{\sum_{k=1}^n D(f, k) - \sum_{k=1}^n D(b, k)}{n - 2} = -1$

$$\longrightarrow \text{edgeweight}(f) = \frac{D_{fb} + \Delta_{fb}}{2} = 1$$

4. Calcola $\text{edgeweight}(f)$ e $\text{edgeweight}(b)$

$$\longrightarrow \text{edgeweight}(b) = \frac{D_{fb} - \Delta_{fb}}{2} = 2$$

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

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

Elimina f e b da D .

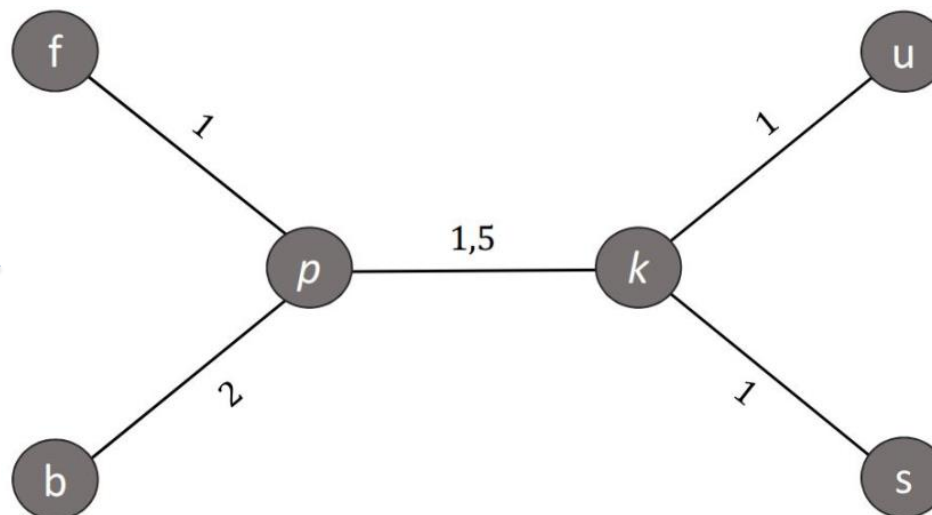
Neighbor-Joining

Parte 3

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

p e k nodi interni \rightarrow arco di peso 1,5.
Costruisci T .

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



L'algoritmo è terminato!

Neighbor-Joining

Parte 4 - Calcolo discrepanza tra D e $D(T)$

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

Poca discrepanza tra D e $D(T)$.

Complessità Temporale

2 step:

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



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

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

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

Unweighted Pair Group Method with Arithmetic Mean

Parte 1

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

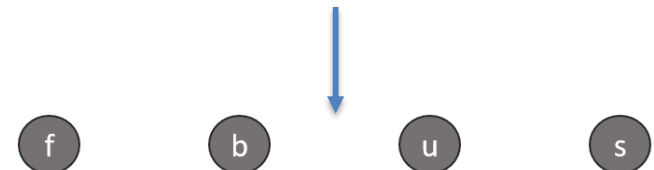
- Foglie → entità biologiche attualmente esistenti;
- Nodi interni → speciazioni;
- Ogni vertice ha associato un numero non negativo → età del vertice;
- Peso degli archi → differenza tra le età dei nodi;

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

1. A partire da D crea un cluster per foglia



Unweighted Pair Group Method with Arithmetic Mean

Parte 2

2. Scegli i due cluster X e Y più vicini secondo la seguente definizione di distanza:

$$D_{X,Y} = \frac{1}{|X| \cdot |Y|} \cdot \sum_{i \in X, j \in Y} D_{i,j} \longrightarrow D_{u,s} = 2$$

3. Crea un cluster $\{u, s\}$ tale che $\rightarrow \{u, s\} = \{u\} \cup \{s\}$

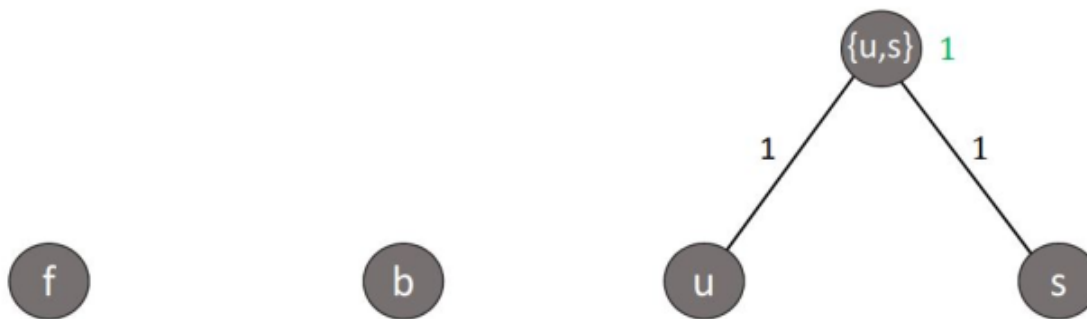
4. Crea in T un nodo interno per $\{u, s\}$, calcola la sua età ed il peso degli archi di u e s

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

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

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

L'albero risultante:



Unweighted Pair Group Method with Arithmetic Mean

Parte 3

5. Aggiorna $D \rightarrow$ elimina u e s ed aggiungi $\{u, s\}$ calcolando la distanza con gli altri elementi

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

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

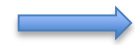


$$D = \begin{matrix} & \text{specie} & & & \\ & f & b & \{u, s\} & \\ f & 0 & 3 & 3,5 & \\ b & 3 & 0 & 4,5 & \\ \{u, s\} & 3,5 & 4,5 & 0 & \end{matrix}$$

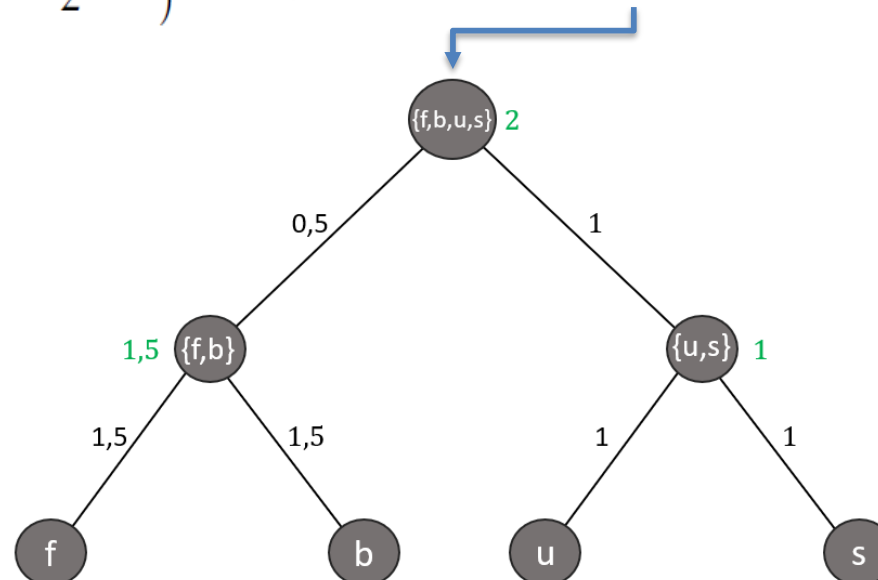


Esegui gli step fino a che non ottieni una matrice 1×1 .

$$\begin{matrix} & \text{specie} & & \\ & \{f, b, u, s\} & & \\ \text{specie} & \{f, b, u, s\} & & \\ D = \{f, b, u, s\} & \left(\begin{matrix} 2 \end{matrix} \right) & & \end{matrix}$$



Il cluster $\{f, b, u, s\}$ è la radice in 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 $\rightarrow T(\text{UPGMA}) = O(n)$.

Queste operazioni vengono iterate $n - 1$ volte, ovvero fino a che non si ottiene una matrice $1 \times 1 \rightarrow O(n - 1)$


$$T(\text{Totale}) = T(\text{UPGMA}) \times O(n - 1) \simeq O(n^2)$$

La discussione è terminata, grazie per l'attenzione!

