

Curs 7:

Gruparea datelor (I)

# Structura

- Gruparea datelor
  - Concepte de bază
  - Evaluarea calității grupării
- Algoritmi partiționali
  - kMeans
  - Fuzzy cMeans
- Algoritmi ierarhici
  - Algoritmi aglomerativi
  - Algoritmi divizivi
- Algoritmi bazați pe estimarea densității datelor – Curs 8
- Algoritmi bazați pe modele probabiliste – Curs 8

# Scopul grupării datelor (reminder)

## Ce se cunoaște?

- un **set de date** (nu neapărat structurate)
- o măsură de **similaritate/disimilaritate** între date (măsura e specifică problemei de rezolvat) pe baza căreia se construiește o **matrice de similaritate/disimilaritate**

## Ce se dorește?

- un **model** care descrie modul în care se **grupează datele** în clustere (grupuri) astfel încât datele care aparțin aceluiași cluster să fie mai similare între ele decât cele care aparțin unor clustere diferite

## Care este scopul final?

- Să se poată verifica dacă două date aparțin aceluiași cluster
- Să se determine clusterul de care aparține o dată
- Să se identifice date atipice (care nu aparțin nici unui cluster)

# Scopul grupării datelor (reminder)

## Exemple:

- **Customer segmentation** = identificarea grupurilor de clienți care au comportamente similare
- **User profiles extraction** = identificarea grupurilor de utilizatori ai unui serviciu web caracterizați prin comportamente similare
- **Document clustering** = identificarea unor grupuri de documente similare din punct de vedere al conținutului
- **Image segmentation** = identificarea unor regiuni omogene într-o imagine
- **Gene expression analysis** = gruparea genelor cu roluri similare

## Gruparea permite:

- sumarizarea și/sau vizualizarea datelor cu scopul de a le înțelege mai bine

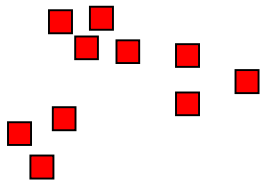
# Particularități ale grupării datelor

Este un proces nesupervizat:

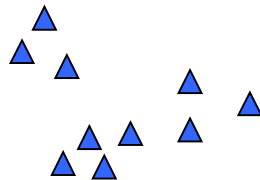
- Setul de antrenare conține doar valori ale atributelor
- Eticheta clasei nu e cunoscută



Câte clustere?

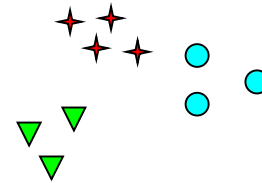


Două clustere

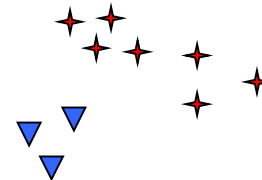
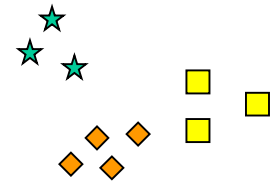


Problema grupării este “rău-definită”:

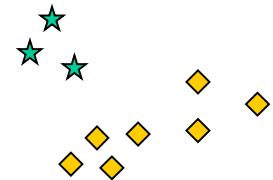
- Identificarea grupurilor este dificilă
- Decizia poate avea caracter subiectiv



Șase clustere



patru clustere



# Concepte de bază

- **Cluster** = grup de date care sunt similare
- **Matrice de (di)similaritate** pt un set de  $n$  date = matrice  $n \times n$  în care pe linia  $i$  coloana  $j$  se află valoarea similarității/disimilarității între data  $i$  și data  $j$
- **Clustering (grupare)** = proces de identificare a clusterelor
- **Prototip (reprezentant)** = “obiect” reprezentativ pentru datele dintr-un cluster
  - **Centroid** = media datelor dintr-un cluster – centroidul nu este neapărat un element din setul de date
  - **Medoid** = data din cluster care este cea mai apropiată de media clusterului – medoidul aparține setului de date
- **Raza clusterului** = media distanțelor dintre datele din cluster și prototipul acestuia
- **Diametrul clusterului** = distanța (disimilaritatea) maximă dintre oricare două date ale clusterului

# Tipuri de clustering

- Crisp vs fuzzy clustering

- Crisp clustering = fiecare dată aparține unui singur cluster
- Fuzzy clustering = o dată poate aparține mai multor clustere (poate fi caracterizată prin grad diferit de apartenență pentru fiecare cluster)

- Flat vs hierarchical clustering

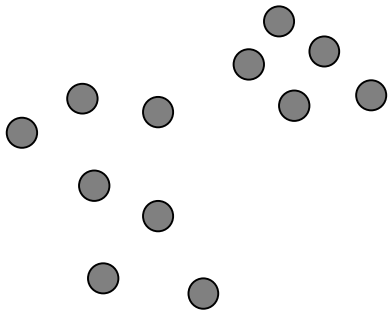
- Flat (partitional) clustering = rezultatul este un set de clustere (o partiție)
- Hierarchical clustering = rezultatul este o ierarhie de partiții

- Variante de algoritmi

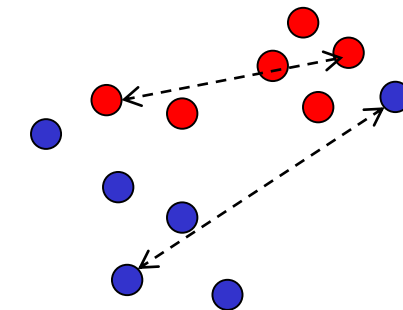
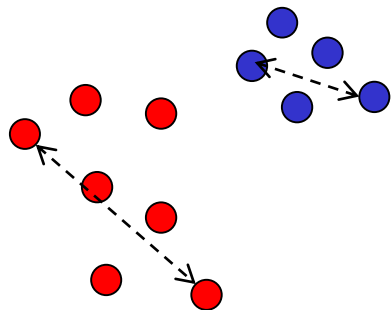
- Algoritmi partiționali (ex: kMeans, Fuzzy cMeans)
- Algoritmi hierarhici (alg. aglomerativi, alg. divizivi)
- Algoritmi bazați pe densitate (ex: DBSCAN, OPTICS, HDBSCAN)
- Algoritmi bazați pe modele probabiliste (ex: EM = Expectation Maximization)
- Algoritmi bazați pe analiza spectrului (valori și vectori proprii) ai grafului de similaritate asociat datelor

# Măsuri de calitate

- Nu există un indicator unic pentru evaluarea calității unei grupări
- Cea mai comună abordare constă în estimarea:
  - **Compacității** clusterelor (variabilitate **intra-cluster** – ar trebui să fie mică)
  - Gradului de **separare** dintre datele aparținând unor cluster diferite (variabilitate **inter-cluster** – ar trebui să fie mare)



Calitate bună



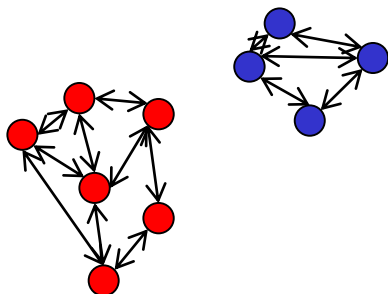
Calitate slabă



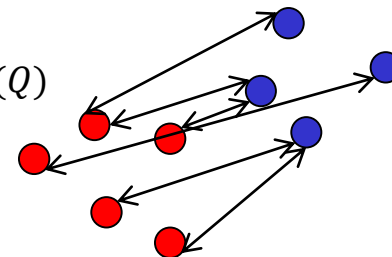
# Măsuri de calitate

Notatii:

- $P$ : setul de perechi de date ce aparțin aceluiași cluster
- $Q = D \times D - P$  (restul perechilor: o dată din pereche aparține unui cluster iar cealaltă unui alt cluster)



$$Intra = \sum_{(x_i, x_j) \in P} d(x_i, x_j) / card(P)$$
$$Inter = \sum_{(x_i, x_j) \in Q} d(x_i, x_j) / card(Q)$$



Exemple de perechi de date utilizate în calculul distanțelor intra-cluster

Exemple de perechi de date utilizate în calculul distanțelor inter-cluster

- Intra-cluster to inter-cluster distance ratio = Intra-cluster/Inter-cluster (**valori mici corespund unei calități mai bune**)

# Măsuri de calitate

- Silhouette coefficient

(N = numărul de date din set)

Obs:

- S ia valori în (-1,1)
- Valori mai mari indică o grupare mai bună

$$S = \frac{1}{N} \sum_{i=1}^N S_i$$

$$S_i = \frac{D \min_i^{out} - D avg_i^{in}}{\max\{D \min_i^{out}, D avg_i^{in}\}}$$

$D avg_i^{in}$  = media distantelor dintre  $x_i$  si celelalte date din clusterul lui  $x_i$

$D avg_i^j$  = media distantelor dintre  $x_i$  si datele din clusterul  $j$  ( $j \neq$  clusterul lui  $x_i$ )

$$D \min_i^{out} = \min_j D avg_i^j$$

# Algoritmi partiționali

- **Input:** set de date  $D=\{x_1, x_2, \dots, x_N\}$ ,  $K$  = nr de clustere
- **Output:** partiție  $P=\{C_1, C_2, \dots, C_K\}$  a setului de date  $D$
- Cel mai popular algoritm partițional: **kMeans (varianta Lloyd)**
- Folosește ideea că prototipul fiecărui cluster este media aritmetică a elementelor din clusterul respectiv (denumit **centroid**)
- Este un algoritm euristic (bazat pe principiul greedy) având ca scop **minimizarea distanțelor dintre date și centroizii corespunzători** (varianța intra-cluster) și care constă într-o prelucrare iterativă care repetă doi pași:
  - **Pas 1: (re)asignează** datele la clustere pe criteriul celui mai apropiat centroid – în general se utilizează distanța euclidiană
  - **Pas 2: recalculează** centroizii folosind datele asignate la fiecare cluster

# kMeans

- **Input:** set de date  $D=\{x_1, x_2, \dots, x_N\}$ ,  $K$  = nr de clustere
- **Output:** partiție  $P=\{C_1, C_2, \dots, C_K\}$  a setului de date  $D$

kMeans ( $D, k$ )

Initializare centroizi  $C_1, C_2, \dots, C_K$  (prin **selecție aleatoare** din setul de date sau folosind o strategie euristică)

**repeat**

- **asignează** fiecare dată din  $D$  la clusterul corespunzător celui mai apropiat centroid (în raport cu măsura de similaritate/disimilaritate folosită)
- **actualizează** fiecare centroid prin calculul mediei elementelor asiginate clusterului

**until** <partiția este stabilă (la ultima iterație nu s-a modificat asignarea datelor la clustere) >

**Obs:**

- condiția de oprire poate fi bazată pe utilizarea unei măsuri de calitate (de exemplu varianța intra-cluster este suficient de mică)

# kMeans

- **Caracteristici**

- kMeans este o metodă bazată pe prototipuri care are ca scop minimizarea sumei pătratelor erorilor (SSE) – distanțele dintre date și centroizii corespunzători

$$SSE(X, C) = \sum_{k=1}^K \sum_{x \in C_k} d^2(x, c_k) = \sum_{k=1}^K \sum_{x \in C_k} \sum_{j=1}^n (x_j - c_{kj})^2$$

(în cazul distanței euclidiene)

- **Complexitate:**  $O(n \cdot N \cdot K \cdot \text{iterații})$  ( $n$ =nr de attribute,  $N$ =nr de date,  $K$ =nr de clustere)
- **Pre-procesare utilă:** normalizare (eventual reducerea dimensiunii – ex: PCA)
- **Post-procesare utilă:**
  - Eliminarea clusterelor mici (sunt considerate excepții)
  - Fragmentarea clusterelor caracterizate prin variabilitate mare
  - Reunirea clusterelor apropiate

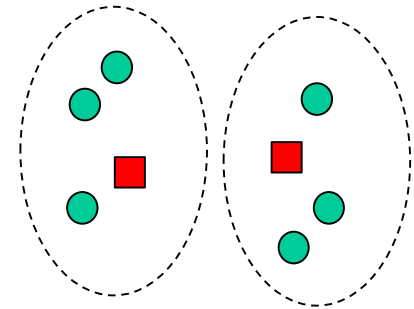
# kMeans

- **Inițializarea centrozilor**

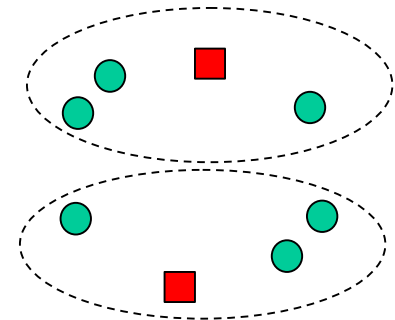
- Performanța algoritmului kMeans este influențată de inițializarea centrozilor
- În varianta bazată pe selecție aleatoare, o strategie de a determina o inițializare adecvată este de a aplica **algoritmul repetat** pornind de la valori diferite ale seed-ului corespunzător generatorului de valori aleatoare

- **kMeans++:**

- Primul centru se alege aleator din setul de date
- Al doilea centru se alege folosind o distribuție de probabilitate care favorizează datele îndepărtate de primul centroid
- Al treilea centru se alege folosind o distribuție de probabilitate care favorizează datele îndepărtate de primii doi centroizi
- Se continuă procesul până la alegerea tuturor celor K centroizi



Inițializare favorabilă

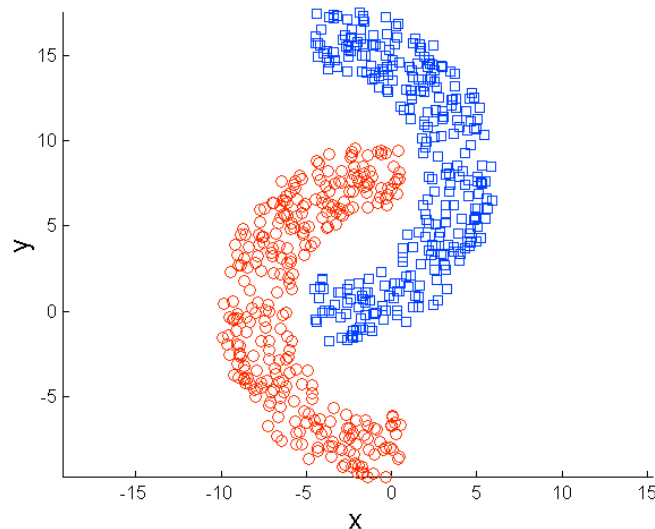


Inițializare nefavorabilă

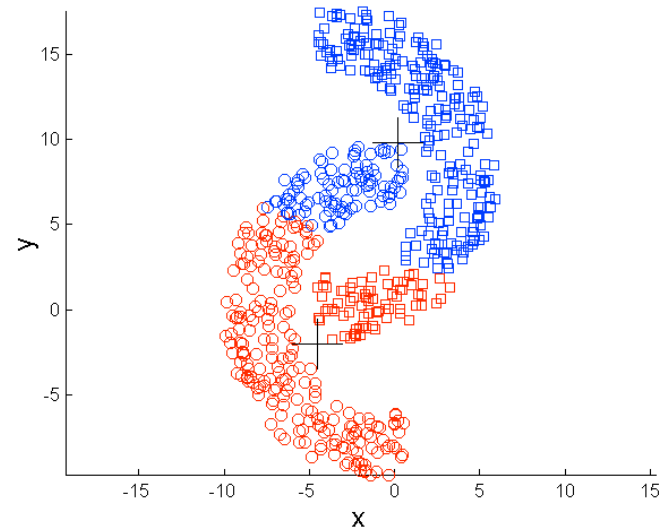
# kMeans

## Limite:

- Nu funcționează bine dacă datele nu sunt “sferice” sau bine separate (distribuțiile de probabilitate corespunzătoare clusterelor nu sunt neapărat normale)
  - **Soluție:** utilizarea altor abordări (e.g. clustering bazat pe densitate)



Clusterele reale



Rezultatul Kmeans

# kMeans

## Limite:

- Rezultatul poate fi ușor influențat de erorile din date sau de datele atipice
  - **Soluție:** utilizarea medianei în locul mediei în construirea centroizilor (reprezentanții clusterilor)
    - **kMedian** – se determină mediana la nivel de componentă (corespunde cazului în care se utilizează distanța Manhattan în criteriul de optimizat)
    - **kMedoid** – se determină elementul din setul de date care este cel mai apropiat de media clusterului (sau elementul din cluster care minimizează suma distanțelor dintre celelalte elemente ale clusterului și acel element); cel mai reprezentativ algoritm este **Partitioning around Medoids – PAM** (mai puțin eficient decât kMeans)
- Varianta bazată pe calculul mediei poate fi aplicată doar datelor numerice; pentru date ordinale se poate utiliza varianta bazată pe mediană (**kMedian**) iar pentru cele nominale se poate utiliza moda (cea mai frecventă componentă – **kMode**)



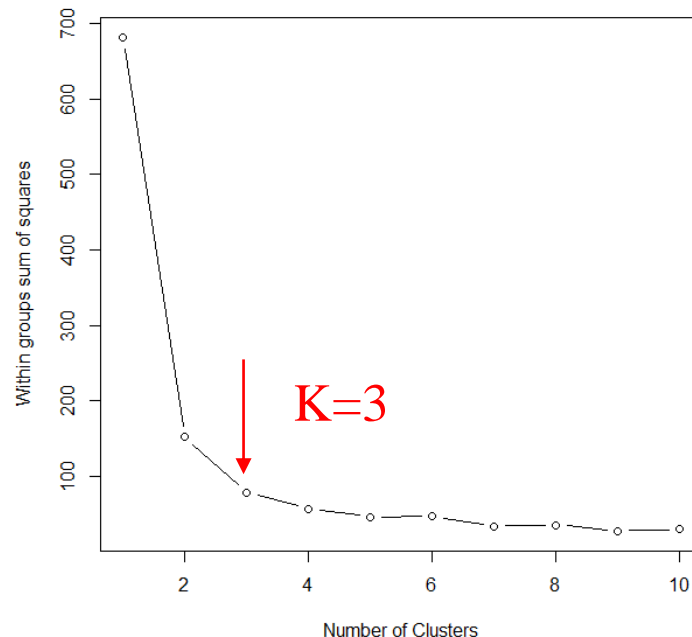
# kMeans

**Limite:** necesită cunoașterea apriori a numărului de clustere

**Soluții:**

- aplică algoritmul pt diferite valori ale lui K și se selectează varianta care corespunde celor mai bune valori ale criteriilor de calitate (un exemplu de metodă este cea cunoscută sub denumirea de “**elbow**”: se calculează **varianța intra-cluster** pentru diferite valori ale lui K și se alege valoarea lui K pentru care se schimbă modul în care descrește varianța = punct de cotitură)

Set date:  
Iris → K=3



# ISODATA

Post-procesarea rezultatelor procesului de clustering prin partiționarea clusterelor cu variabilitate mare și reunirea clusterelor apropiate (ex: alg. ISODATA)

## Ideii principale ale alg ISODATA

- Dacă dimensiunea unui cluster este mai mică decât  $N_{min}$  (un parametru care specifică numărul minim de date dintr-un cluster) atunci clusterul ar trebui reunit cu cel mai apropiat alt cluster
- Dacă distanța dintre două cluster (de exemplu distanța dintre prototipurile clusterelor) este mai mică decât  $D_{min}$  (un parametru care specifică distanța minimă dintre cluster) atunci clusterele ar trebui reunite
- Dacă varianța unui cluster este mai mare decât  $V_{max}$  și numărul de date conținute este mai mare decât  $2 \cdot N_{min}$  atunci clusterul poate fi divizat în două alte cluster:
  - Identifică atributul  $j$  pt care varianța este maxmă
  - Din prototipul  $c_k$  sunt construite două alte prototipuri  $c'$  și  $c''$  prin înlocuirea valorii atributului  $j$  din  $c_k$  cu  $c_k(j) - b$  respectiv  $c_k(j) + b$ , ( $b$  este un parametru setat de către utilizator)

# Fuzzy cMeans

Ideea grupării fuzzy (soft):

- O dată nu aparține unui singur cluster ci poate aparține mai multor cluster (cu un anumit **grad de apartenență** pentru fiecare cluster)
- Rezultatul unei grupări fuzzy este o **matrice de apartenență M** de dimensiune  $N \times K$

( $N$ = nr date,  $K$ = nr cluster);

$M(i,j)$  = o valoare din  $[0,1]$  care corespunde **gradului de apartenență** a datei  $i$  la clusterul  $j$

**Obs:** Fuzzy cMeans poate fi utilizată în segmentarea imaginilor, prelucrarea datelor în geostatistică, gruparea datelor din metabolomică etc.

# Fuzzy cMeans

## Algorithm

- Initializare matrice de apartenență (M)
- **Repeat**
  - Calcul centroizi ( $c_k$ ,  $k=1,\dots,K$ )
  - Actualizare valori matrice de apartenență ( $m_{ij}$ ,  $i=1,\dots,N$ ,  $j=1,\dots,K$ )
- until** <matricea de apartenență nu se mai modifică>

**Obs:** pentru a obține o partiție, la finalul prelucrării fiecare dată este asignată la clusterul ce corespunde celui mai mare grad de apartenență

## Calculul centroizilor

$$c_j = \frac{\sum_{i=1}^n M_{ij}^p x_i}{\sum_{i=1}^n M_{ij}^p}, \quad j = \overline{1, K}$$

$p > 1$  este un parametru  
(de exemplu,  $p=2$ )

## Calculul gradului de apartenență

$$M_{ij} = \frac{1/\|x_i - c_j\|^{2/(p-1)}}{\sum_{k=1}^K 1/\|x_i - c_k\|^{2/(p-1)}}$$

$$i = \overline{1, N}, j = \overline{1, K}$$

# Algoritmi ierarhici

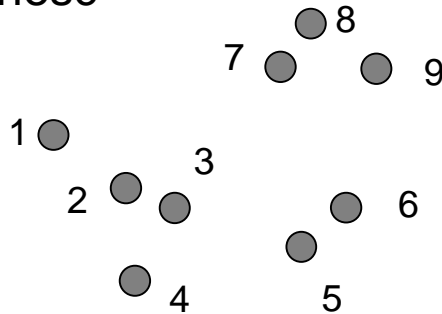
**Obs:** una dintre principalele limite ale algoritmilor partiționali e faptul că necesită cunoașterea numărului de clustere.

**Altă abordare:** se construiește o **ierarhie de partiții** – conduce la o structură arborescentă numită **dendrogramă**

- In varianta **bottom-up** (metoda **aglomerativă**)
  - Se pornește cu o partiție de clustere ce conțin fiecare câte o singură dată (reprezintă frunze în structura arborescentă)
  - Se reunesc clusterelor care sunt suficient de similare între ele – procesul de reunire se repetă până se ajunge la un singur cluster (reprezintă rădăcina arborelui)
- In varianta **top-down** (metoda **divizivă**)
  - Se pornește cu o partiție ce conține un singur cluster (cu toate datele)
  - Se partiționează clusterii mari aplicând o tehnica partițională (ex: kMeans) – procesul continuă până când se ajunge la clustere ce conțin câte o singură dată.

# Metoda aglomerativă

**Idee:** se identifică la fiecare etapă care sunt cele mai similare clustere și se reunesc



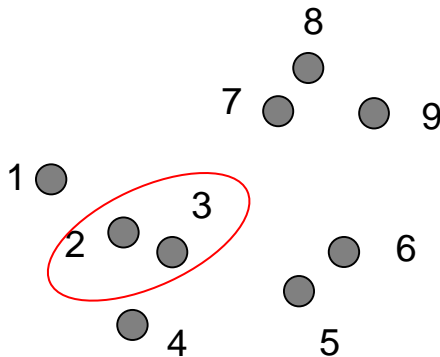
Matrice de disimilaritate

	1	2	3	4	5	6	7	8	9
1	0	2	3	4	7	8	6	8	10
2	2	0	1	2	4	6	7	8	9
3	3	1	0	2	3	5	6	8	9
4	4	2	2	0	3	6	9	10	11
5	7	4	3	3	0	1	4	6	5
6	8	6	5	6	1	0	3	4	3
7	6	7	6	9	4	3	0	1	2
8	8	8	8	10	6	4	1	0	2
9	10	9	9	11	5	3	3	2	0

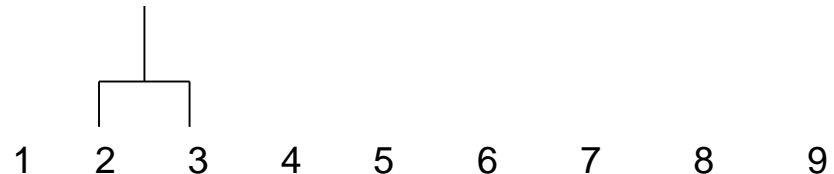
1 2 3 4 5 6 7 8 9

# Metoda aglomerativă

**Idee:** se identifică la fiecare etapă care sunt cele mai similare clustere și se reunesc

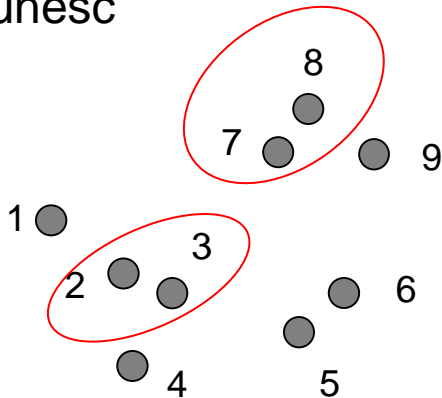


	1	2	3	4	5	6	7	8	9
1	0	2	3	4	7	8	6	8	10
2	2	0	1	2	4	6	7	8	9
3	3	1	0	2	3	5	6	8	9
4	4	2	2	0	3	6	9	10	11
5	7	4	3	3	0	1	4	6	5
6	8	6	5	6	1	0	3	4	3
7	6	7	6	9	4	3	0	1	2
8	8	8	8	10	6	4	1	0	2
9	10	9	9	11	5	3	3	2	0



# Metoda aglomerativă

**Idee:** se identifică la fiecare etapă care sunt cele mai similare clustere și se reunesc



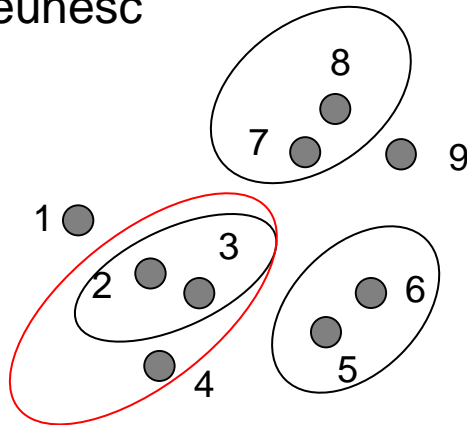
	1	2	3	4	5	6	7	8	9
1	0	2	3	4	7	8	6	8	10
2	2	0	1	2	4	6	7	8	9
3	3	1	0	2	3	5	6	8	9
4	4	2	2	0	3	6	9	10	11
5	7	4	3	3	0	1	4	6	5
6	8	6	5	6	1	0	3	4	3
7	6	7	6	9	4	3	0	1	2
8	8	8	8	10	6	4	1	0	2
9	10	9	9	11	5	3	3	2	0



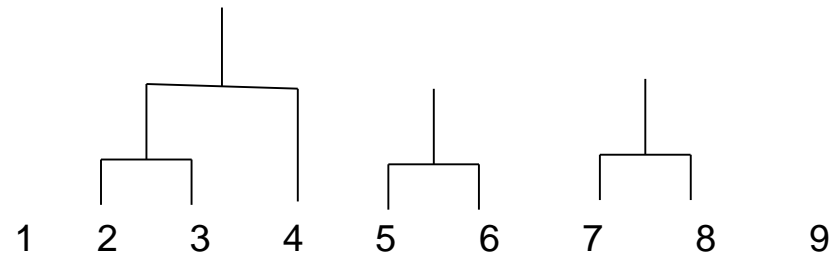


# Metoda aglomerativă

**Idee:** se identifică la fiecare etapă care sunt cele mai similare clustere și se reunesc

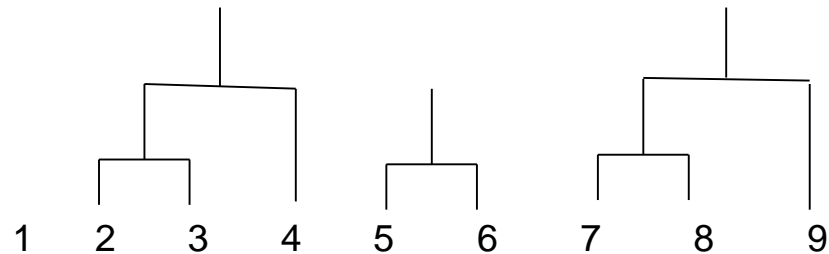
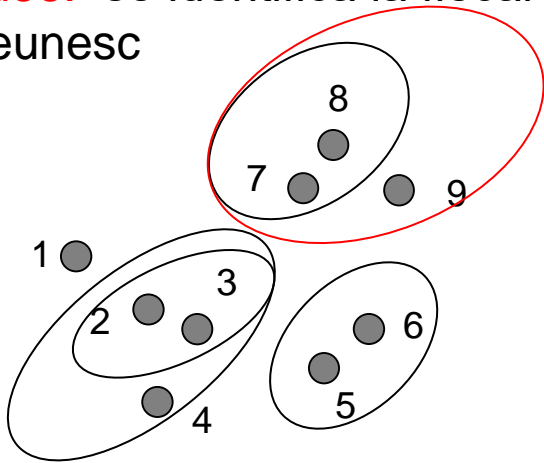


	1	2	3	4	5	6	7	8	9
1	0	2	3	4	7	8	6	8	10
2	2	0	1	2	4	6	7	8	9
3	3	1	0	2	3	5	6	8	9
4	4	2	2	0	3	6	9	10	11
5	7	4	3	3	0	1	4	6	5
6	8	6	5	6	1	0	3	4	3
7	6	7	6	9	4	3	0	1	2
8	8	8	8	10	6	4	1	0	2
9	10	9	9	11	5	3	3	2	0



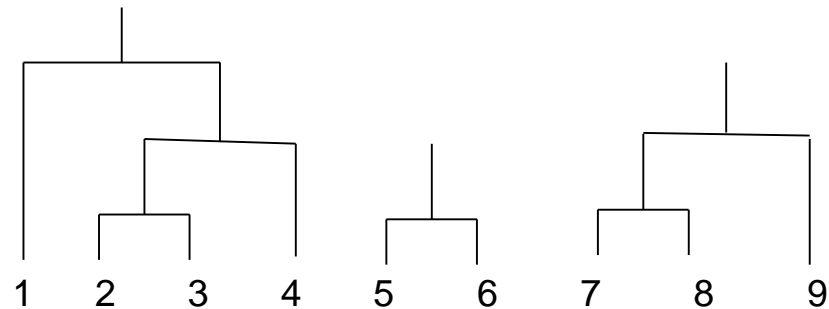
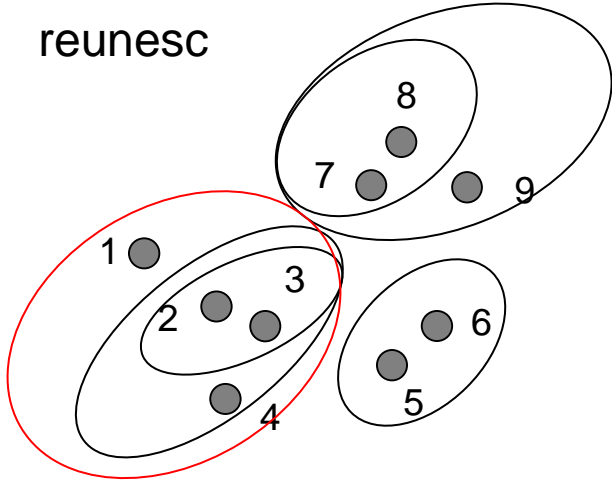
# Metoda aglomerativă

**Idee:** se identifică la fiecare etapă care sunt cele mai similare clustere și se reunesc



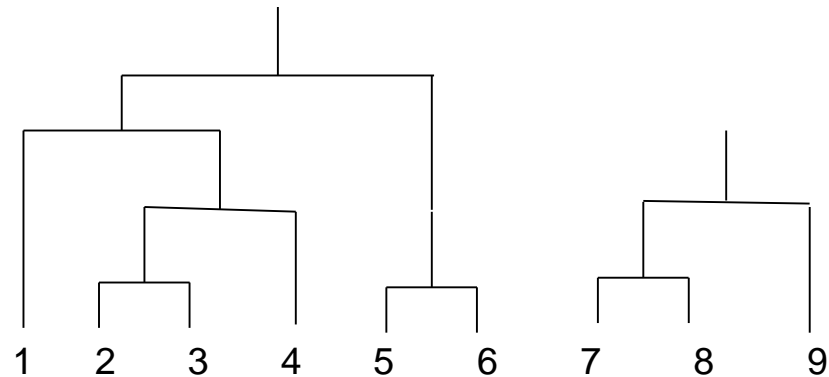
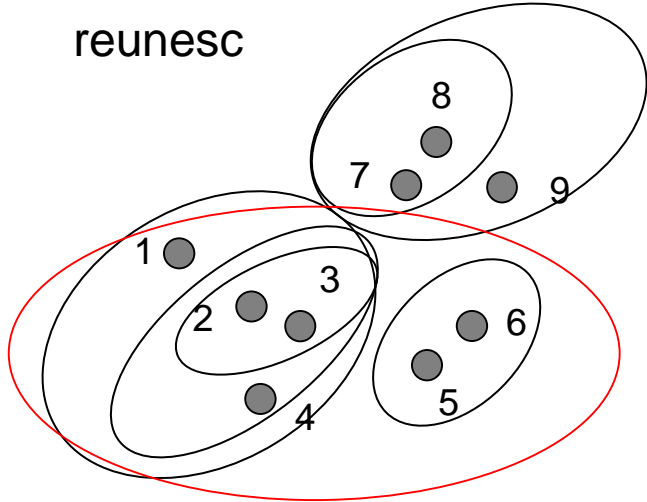
# Metoda aglomerativă

**Idee:** se identifică la fiecare etapă care sunt cele mai similare clusteruri și se reunesc



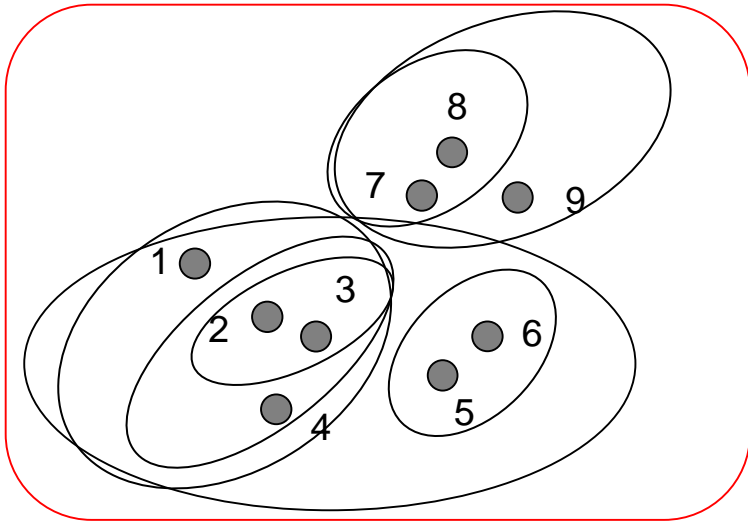
# Metoda aglomerativă

**Idee:** se identifică la fiecare etapă care sunt cele mai similare clusteruri și se reunesc

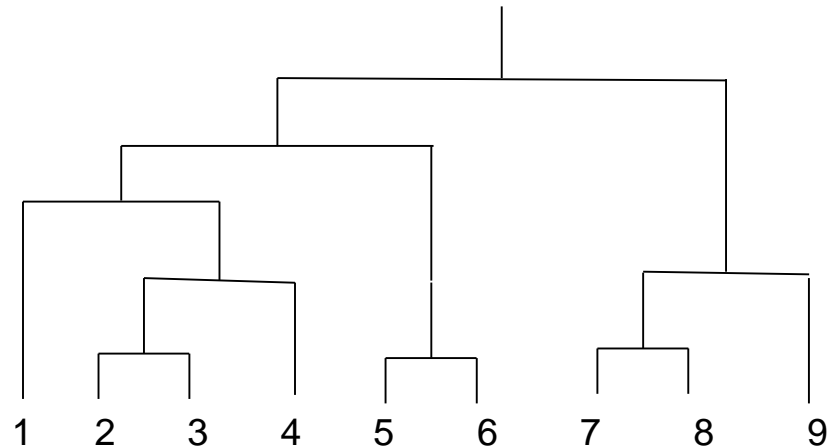


# Metoda aglomerativă

**Idee:** se identifică la fiecare etapă care sunt cele mai similare clustere și se reunesc



## ▪ Dendrograma rezultată

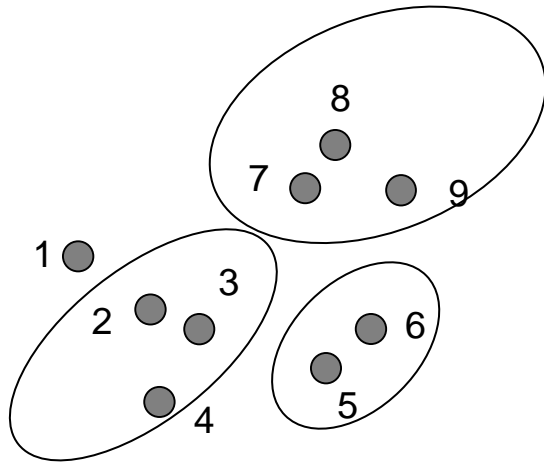


## ▪ Reprezentarea unei dendrogame: ca set de triplete ordonate

(nivel, nr de clustere, clustere)

$\{(0,9,\{\{1\},\{2\},\dots,\{9\}\}) , (1,6,\{\{1\},\{2,3\},\{4\},\{5,6\},\{7,8\},\{9\}\}),$   
 $(2,4,\{\{1\},\{2,3,4\},\{5,6\},\{7,8,9\}\}), (3,3,\{\{1,2,3,4\},\{\{5,6\},\{7,8,9\}\}),$   
 $(4,2,\{\{1,2,3,4,5,6\},\{7,8,9\}), (5,1,\{\{1,2,3,4,5,6,7,8,9\}\})\}$

# Metoda aglomerativă



Partiție:

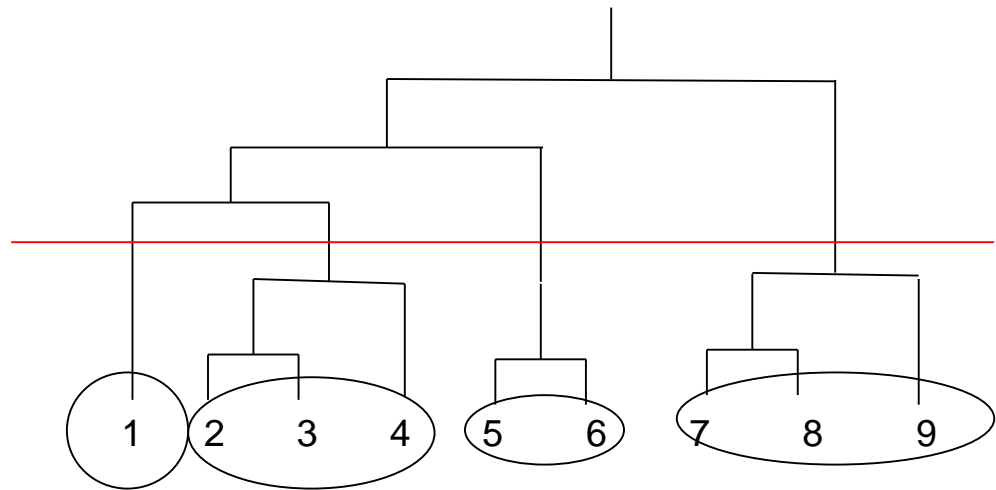
$C1=\{1\}$

$C2=\{2,3,4\}$

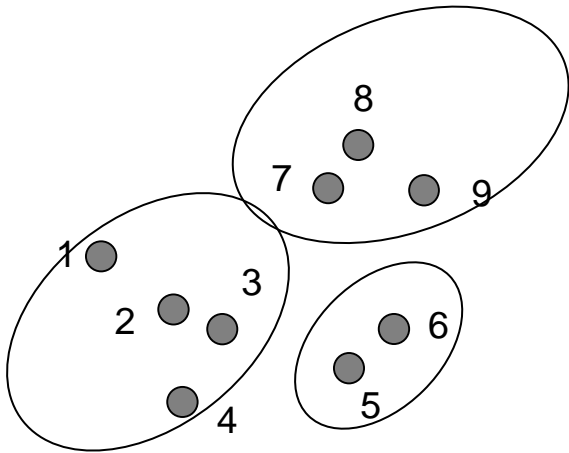
$C3=\{5,6\}$

$C4=\{7,8,9\}$

- Pentru a obține o partiție dendrograma trebuie secționată



# Metoda aglomerativă



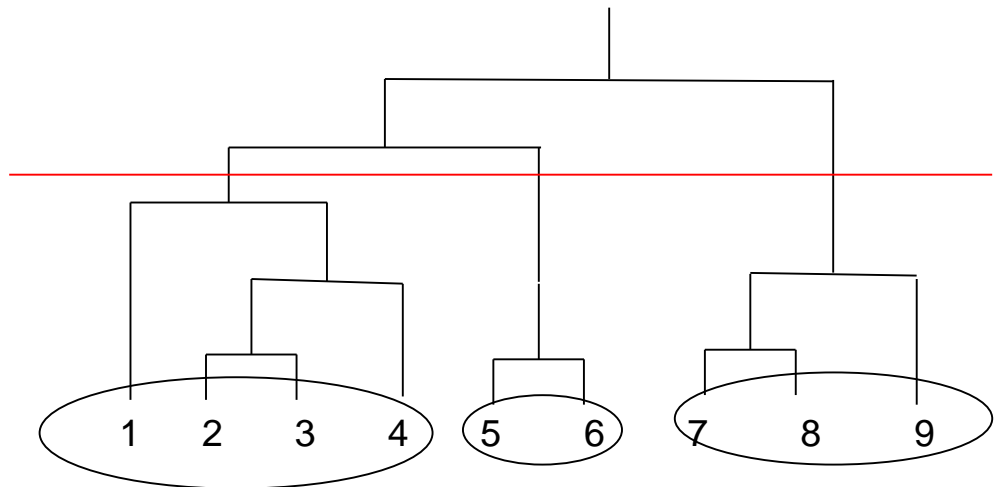
Partiție:

$C1=\{1,2,3,4\}$

$C2=\{5,6\}$

$C3=\{7,8,9\}$

- Schimbând nivelul de secționare se obține o altă partiție

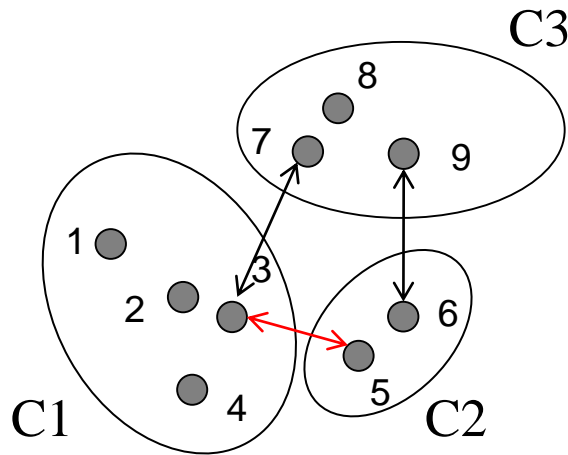


# Metoda aglomerativă

**Problema:** care e criteriul de selecție a clusterelor care se reunesc?

**Răspuns:** se folosește o măsură de disimilaritate între cluster; sunt mai multe moduri de a calcula această măsură:

- **Single-linkage:** cea mai **mică** disimilaritate (distanță) între datele aparținând unor cluster diferite



$$D_{SL}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$$

**Obs:**

- Single linkage permite identificarea unor cluster care nu au neapărat formă elipsoidală
- Este sensibil la erori și date atipice

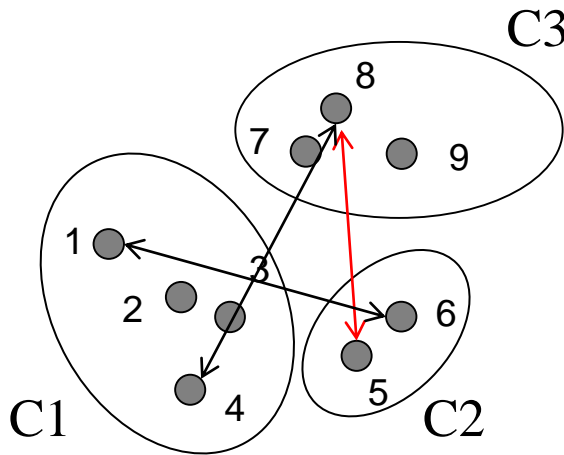


# Metoda aglomerativă

**Problema:** care e criteriul de selecție a clusterelor care se reunesc?

**Răspuns:** se folosește o măsură de disimilaritate între cluster; sunt mai multe moduri de a calcula această măsură

- **Complete-linkage:** cea mai **mare** disimilaritate (distanță) între datele aparținând unor cluster diferite



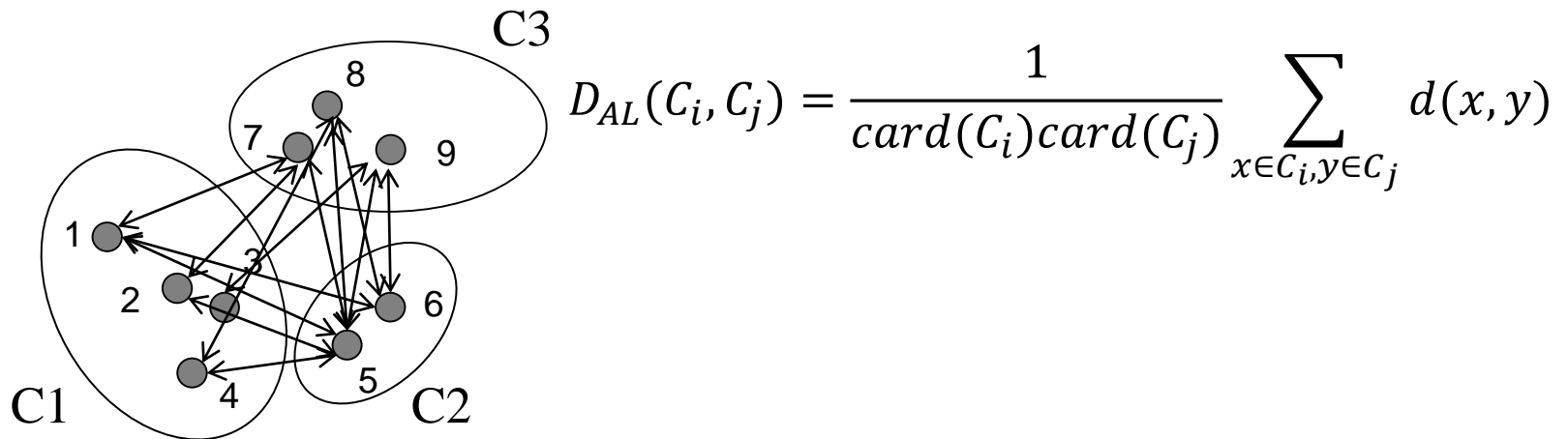
$$D_{CL}(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$$

# Metoda aglomerativă

**Problema:** care e criteriul de selecție a clusterelor care se reunesc?

**Răspuns:** se folosește o măsură de disimilaritate între cluster; sunt mai multe moduri de a calcula această măsură

- **Average-linkage:** media distanțelor dintre datele aparținând unor cluster diferite



# Metoda aglomerativă

**Problema:** care e criteriul de selecție a clusterelor care se reunesc?

**Răspuns:** se folosește o măsură de disimilaritate între cluster; sunt mai multe moduri de a calcula această măsură

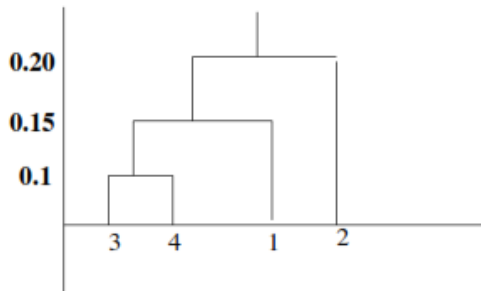
- **Criteriul Ward:** se alege perechea de cluster care a căror reuniune conduce la cea mai mică creștere în varianța intra-cluster ( $SSE(X,C)$ )

# Metoda aglomerativă

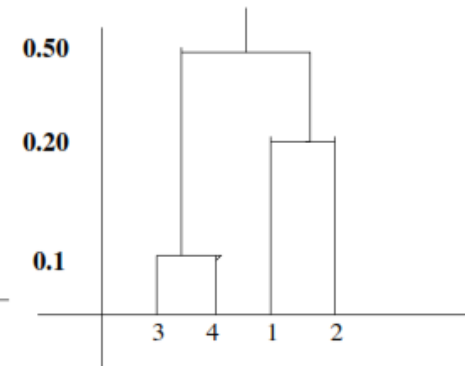
Măsura de disimilaritate între clustere (criteriul de reunire a clusterelor) influențează rezultatul grupării:

	1	2	3	4
1	0.0	0.20	0.15	0.30
2	0.20	0.0	0.40	0.50
3	0.15	0.40	0.0	0.10
4	0.30	0.50	0.10	0.0

(a) Dissimilarity Matrix



(b) Single Link



(c) Complete Link

[Data Clustering: Algorithms and Applications, 2014]

# Metoda aglomerativă

## Algoritm

**Input :** set de date cu N instanțe

$X=\{x_1, x_2, \dots, x_N\}$  + matrice disimilaritate D

**Output:** dendrograma (set de triplete)

agglomerative(X,D)

level=0; k=N

$C=\{\{x_1\}, \{x_2\}, \dots, \{x_N\}\}$ ;  $DE=\{(\text{level}, k, C)\}$

**repeat**

    oldk=k

    level=level+1

    (k,C) = **mergeClusters**(k,C,D)

    D(k) = **recompute** (D(oldK), C)

    DE = **union** (DE, (level,k,C))

    k = k-1

**until** k=1

## Obs

- Funcția **mergeClusters** identifică cele mai apropiate clustere și le reunește
- Funcția **recompute** recalculează matricea de similaritate corespunzătoare setului curent de clustere (folosind single/complete/average linkage)

## Dezavantaje

- Complexitate implementare bazată pe abordare greedy:  $O(N^3)$  – nr operații,  $O(N^2)$  – spațiu memorie
- Variantele de implementare care utilizează cozi de priorități au ordin de complexitate  $O(N^2 \log N)$
- Este un algoritm determinist dar este sensibil la erorile din date

# Metoda divizivă

## Structura generică

**Input** : set de date cu N instanțe  $X=\{x_1, x_2, \dots, x_N\}$

**Output**: dendrograma (tree) T

divisive(X,D)

Initializează arborele T cu nodul rădăcină (corespunde întregului set de date)

### **Repeat**

    Selectează un nod frunză L din T

    Utilizează un **algorithm partițional de clustering** pentru a descompune L în  $L_1, L_2, \dots, L_k$

    Adaugă  $L_1, L_2, \dots, L_k$  ca fii ai lui L în arborele T

**until** <a stopping criterion>

**Obs**: algoritmul partițional poate fi kMeans; un caz particular este **bisecting kMeans** care se bazează pe partiționarea unui cluster în două alte cluster (aplicând la fiecare etapa kMeans pt  $k=2$ )

# Eficientizarea algoritmilor ierarhici

- Algoritmii ierarhici tradiționali au ordin cubic/pătratic de complexitate (în raport cu numărul de date din set) – construirea dendrogramei devine costisitoare în cazul prelucrării unui volum mare de date
- Variante mai eficiente:
  - **BIRCH** – Balanced Iterative Reducing and Clustering using Hierarchies (Zhang et al, 1996) – constă în două etape:
    - Se construiește o structura arborescentă care reprezintă o variantă comprimată a datelor inițiale
    - Se aplică o strategie de grupare pe datele corespunzătoare nodurilor frunză ale structurii arborescente construite în prima etapă
  - **CURE** – Clustering using Representatives (Guho et al, 1998)
    - Fiecare cluster este specificat prin mai mulți reprezentanți
    - Gruparea este constituită folosind eșantioane aleatoare din setul inițial de date

# Sumar

## Algoritm kMeans

- E algoritm partițional → o singură partiție
- Necesită specificarea numărului de clustere
- Este specific pentru date numerice – necesită acces la date
- Generează clustere “sferice” – adecvat pentru clustere cu distribuție apropiată de cea normală
- Cost computațional mic
- Fiecare cluster are un centroid → datele pot fi asignate la clustere pe baza distanței față de centroid (algoritm de tip inductiv)

## Algoritm aglomerativ

- E algoritm ierarhic → ierarhie de partiții
- Nu necesită specificarea numărului de clustere
- Este suficient să se cunoască matricea de (di)similaritate
- Cost computațional mare
- Nu se bazează pe prototipuri → nu oferă un criteriu de asignare a datelor noi la clustere (algoritm de tip transductiv)



# Cursul următor

- Algoritmi bazați pe analiza densității datelor
  - DBSCAN
  - DENCLUE
- Algoritmi bazați pe modele probabiliste
  - EM (Expectation Maximization)