

# **Informatika pre biológov**

**Askar Gafurov**

**26.9.2019**

## Formulácia problému, algoritmus

- **Formulácia problému:** jasne zdefinujeme vstupné a výstupné dáta a aký výstup očakávame pre každý vstup.
- Vo formulácii nehovoríme **akým spôsobom** vypočítame výstupy zo vstupov.
- **Správny algoritmus:** Postup, ktorý určuje spôsob, akým pre každý vstup vypočítame príslušný výstup.

**Biologický problém:** Pomocou hmotnostného spektrometra (mass spectrometer) sme odmerali vo vzorke peptid s hmotnosťou  $K$ . Máme databázu proteínov a chceme zistiť, ktorý z proteínov obsahuje peptid s touto hmotnosťou.

**Informatický problém:** Vstup je postupnosť  $n$  kladných čísel  $a[1], \dots, a[n]$  a číslo  $K$ . Nájdite súvislý úsek tejto postupnosti  $a[i], a[i + 1], \dots, a[j]$ , ktorý svojim súčtom dáva číslo  $K$ .

**Príklad:**

$K=19$

3 4 6 3 6 4 9 2 8

^ ^ ^ ^ ^ ^ ^

## Ďalšie príklady informatických problémov z prednášky

### Najdlhšia spoločná podpostupnosť

- Vstup: dva reťazce
- Problém: Ako z nich ubrať čo najmenej znakov tak, aby sa potom rovnali?
- Výstup: Spoločná podpostupnosť po ubraní znakov, resp. jej dĺžka

### Najkratšie spoločné nadslovo

- Vstup: niekoľko reťazcov
- Výstup: najkratší reťazec, ktorý obsahuje všetky vstupné reťazce ako súvislé podreťazce

**Informatický problém:** Vstup je postupnosť  $n$  kladných čísel  $a[1], \dots, a[n]$  a číslo  $K$ . Nájdite súvislý úsek tejto postupnosti  $a[i], a[i + 1], \dots, a[j]$ , ktorý svojim súčtom dáva číslo  $K$ .

**Triviálne riešenie:** skúšame všetky možnosti

```
pre každé i od 1 po n
|   pre každé j od i po n
|   |   suma := 0;
|   |   pre každé u od i po j
|   |   |   suma := suma + a[u]
|   |   ak suma = K, vypíš i, j
```

$K=19$

```
3 4 6 3 6 4 9 2 8
      i       j
```

## Ako dlho takýto program pobeží?

- Naimplementovať do počítača a odmerať
- Na akom počítači? Na akých vstupoch?
- **Časová zložitosť**: (označujeme  $O(f(n))$ )
  - Zvolíme si parameter charakterizujúci množstvo dát  
napr. počet prvkov vstupnej postupnosti  $n$
  - Pre každú veľkosť vstupu **odhadneme najhorší prípad**
  - Zanedbáme konštanty

```
pre každé i od 1 po n
|   pre každé j od i po n
|   |   suma := 0;
|   |   pre každé u od i po j
|   |   |   suma := suma + a[u]
|   |   ak suma = K, vypíš i, j
```

**Časová zložitosť**: kubická, alebo  $O(n^3)$

## Prečo je časová zložitosť dôležitá a konštanty nie?

		$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$	$O(2^n)$
Čas na	10	$\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$
vyriešenie	50	$\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$	2 weeks
problému	100	$\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$	2800 univ.
veľkosti	1000	$\epsilon$	$\epsilon$	0.02s	4.5s	—
...	10000	$\epsilon$	0.01s	2.1s	75m	—
	100000	0.04s	0.12s	3.5m	52d	—
	1 mil.	0.42s	1.4s	5.8h	142yr	—
	10 mil.	4.2s	16.1s	24.3d	140000yr	—
Max veľkosť	1s	2.3 mil.	740000	6900	610	33
problému	1m	140 mil.	34 mil.	53000	2400	39
vyriešená za	1d	200 bil.	35 bil.	2 mil.	26000	49
Zvýšenie	+1	—	—	—	—	$\times 2$
času so	$\times 2$	$\times 2$	$\times 2+$	$\times 4$	$\times 8$	—
zvýšeným $n$						

## Efektívnejší algoritmus

- Najprv si predpočítame pre každý začiatok postupnosti jej súčet

$$S[i] = a[1] + a[2] + \dots + a[i]$$

a:	3	4	6	3	6	4	9	2	8
S[0] := 0	S:	3	7	13	16	22	26	35	...

pre každé  $i$  od 1 po  $n$

$$| \quad S[i] = S[i-1] + a[i]$$

- Potom súčet podpostupnosti od  $i$  po  $j$  vieme spočítať jednoducho ako

$$S[j] - S[i - 1]$$

pre každé  $i$  od 1 po  $n$

| pre každé  $j$  od  $i$  po  $n$

| | ak  $S[j] - S[i-1] = K$ , vypíš  $i, j$

- **Časová zložitosť:** kvadratická, alebo  $O(n^2)$
- Ak sú všetky čísla kladné, dá sa aj v lineárnom čase  $O(n)$



## Ďalší príklad informatického problému z prednášky

### Najkratšie spoločné nadslovo

- Vstup: niekoľko reťazcov
- Výstup: najkratší reťazec, ktorý obsahuje všetky vstupné reťazce ako súvislé podreťazce

### Príklad:

Vstup: GCCAAC, CCTGCC, ACCTTC

Výstup: CCTGCCAACCTTC (najkratšie možné)

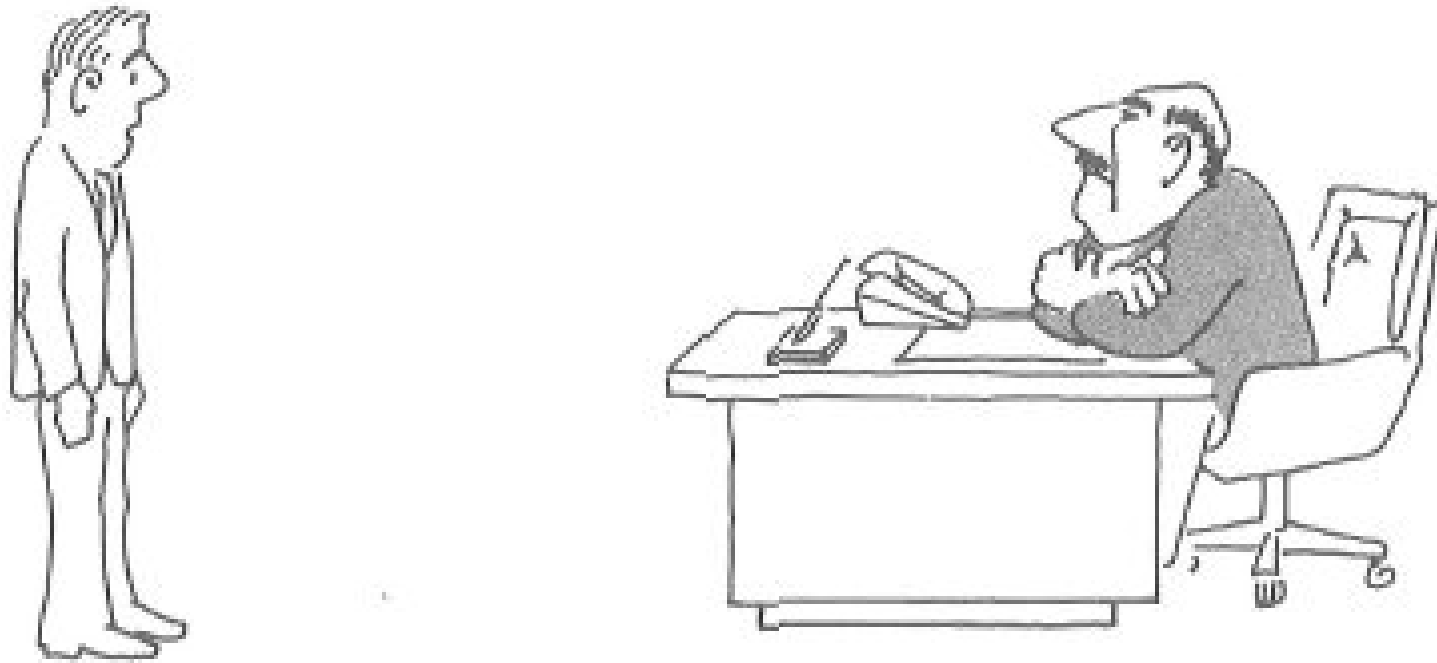
**Ako rýchly algoritmus poznáme pre tento problém?**

**Najkratšie spoločné nadslovo**

Nepoznáme algoritmus, ktorý by bežal v polynomiálnom čase

t.j.  $O(n^k)$  pre nejakú konštantu  $k$

Tento problém je **NP-t'ažký**.



“I can’t find an efficient algorithm, I guess I’m just too dumb.”



“I can’t find an efficient algorithm, because no such algorithm is possible!”



"I can't find an efficient algorithm, but neither can all these famous people."

## Ako sa vysporiadať s NP-ťažkými problémami?

### Heuristické algoritmy

- Nájde **aspoň nejaké riešenie**, aj keď nie nutne optimálne
- Nejde teda o správny algoritmus riešiaci náš problém, lebo pre niektoré vstupy dáva zlú odpoveď
- Radšej ale horšia odpoveď rýchlo, ako perfektná o milión rokov

**Príklad:** Heuristika pre najkratší spoločný nadreťazec: v každom kroku zlepíme dva reťazce s najväčším prekryvom

Príklad: CATATAT, TATATA, ATATATC

Optimum: CATATATATC, dĺžka 10

Heuristika: CATATATCTATATA, dĺžka 14

## Ako so vysporiadať s NP-t'ažkými problémami?

### Aproximačný algoritmus

- Často vieme dokázať, že nejaká heuristika sa vždy priblíži k optimálnemu riešeniu aspoň po určitú hranicu

**Príklad:** Heuristika pre najkratší spoločný nadreťazec: v každom kroku zlepíme dva reťazce s najväčším prekryvom

Je dokázané, že vždy nájde najviac 3,5-krát dlhší reťazec ako najlepšie riešenie.

Informatici predpokladajú, že v skutočnosti najviac 2-krát dlhší, ale nevieme to dokázať.

## Ako so vysporiadať s NP-t'ažkými problémami?

### Exaktný výpočet pomocou iného problému

- Preformulovať do podoby jedného z dobre známych NP-t'ažkých problémov (napr. celočíselné lineárne programovanie, a pod.)
- Múdri ľudia napísali programy, ktoré vedia riešiť tieto známe problémy **aspoň v niektorých prípadoch** (CONCORD, CPLEX, a pod.)

### Preformulovať problém

- Je toto skutočne jediná rozumná formulácia biologického problému ktorý chceme vyriešiť?



## Zhrnutie

- Problémy zo skutočného života je dobré najskôr sformulovať tak, aby bolo jasné, aké výsledky očakávame pre každý možný vstup.
- Takáto formulácia by mala byť oddelená od postupu (algoritmu) riešenia.
- Informatici merajú čas v O-čkach, ktoré abstrahujú od detailov konkrétneho počítača.
- Vytvorenie efektívneho algoritmu je umenie! Časť z toho sú finty (ako napr. dynamické programovanie).
- Pre niektoré problémy poznáme iba Nechutne Pomalé algoritmy (NP-ťažké).
- Aj napriek tomu vo veľa prípadoch vieme pomôcť.

**Úvod do dynamického programovania  
(cvičenie)**

**Broňa Brejová**

**29.9.2022**

## Problém platenia minimálnym počtom mincí

**Vstup:** hodnoty  $k$  mincí  $m_1, m_2, \dots, m_k$  a cieľová suma  $X$  (všetko kladné celé čísla)

**Výstup:** najmenší počet mincí, ktoré potrebujeme na zaplatenie  $X$

**Príklad:**  $k = 3, m_1 = 1, m_2 = 2, m_3 = 5, X = 13$

**Odbočka: ešte matematickejšia formulácia bez slov minca, suma,...**

**Vstup:** kladné celé čísla  $m_1, m_2, \dots, m_k$  a  $X$

**Výstup:** celé číslo  $n$  a  $n$  čísel  $x_1, \dots, x_n$ , pre ktoré platia nasledujúce podmienky:

- $x_i \in \{m_1, m_2, \dots, m_k\}$  pre každé  $i = 1, 2, \dots, n$
- $\sum_{i=1}^n x_i = X$
- $n$  je najmenšie možné.

## Problém platenia minimálnym počtom mincí

**Vstup:** hodnoty  $k$  mincí  $m_1, m_2, \dots, m_k$  a cieľová suma  $X$  (všetko kladné celé čísla)

**Výstup:** najmenší počet mincí, ktoré potrebujeme na zaplatenie  $X$

**Príklad:**  $k = 3, m_1 = 1, m_2 = 2, m_3 = 5, X = 13$

**Príklad:**  $k = 3, m_1 = 1, m_2 = 3, m_3 = 4, X = 6$

## Algoritmus pre všeobecnú sústavu $k$ mincí $m_1, m_2, \dots, m_k$

$$A[i] = 1 + \min\{A[i - m_1], A[i - m_2], \dots, A[i - m_k]\}$$

```
m = [1, 3, 4]
X = 11
k = len(m)
nekonecno = math.inf
A = [0]
for i in range(1, X + 1):
    min = nekonecno
    for j in range(k):
        if i >= m[j] and A[i - m[j]] < min:
            min = A[i - m[j]]
    A.append(1 + min)
print(A)
```

## Program aj s výpisom mincí

```
m = [1, 3, 4]
X = 11
k = len(m)
nekonecno = 1000000
A = [0]
B = [-1]
for i in range(1, X + 1):
    min = nekonecno
    min_minca = -1
    for j in range(k):
        if i >= m[j] and A[i - m[j]] < min:
            min = A[i - m[j]]
            min_minca = m[j]
    A.append(1 + min)
    B.append(min_minca)

while X > 0:
    print(B[X])
    X = X - B[X]
```

## Dynamické programovanie vo všeobecnosti

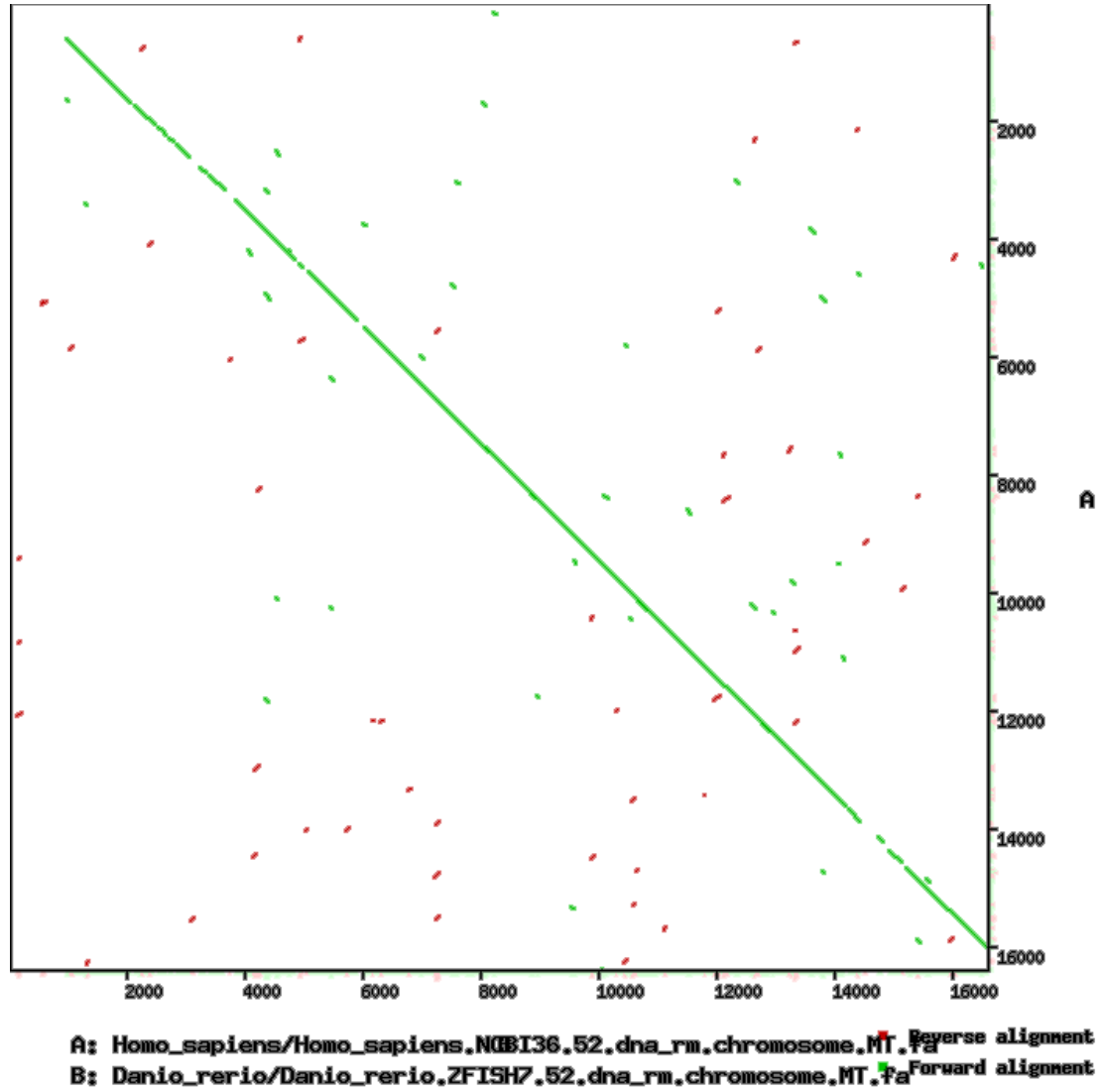
- Okrem riešenia celého problému riešime aj menšie problémy (nazývame ich podproblémy)
- Riešenia podproblémov ukladáme do tabuľky a používame pri riešení väčších podproblémov
- Technika dynamického programovania sa používa na viacero problémov v bioinformatike



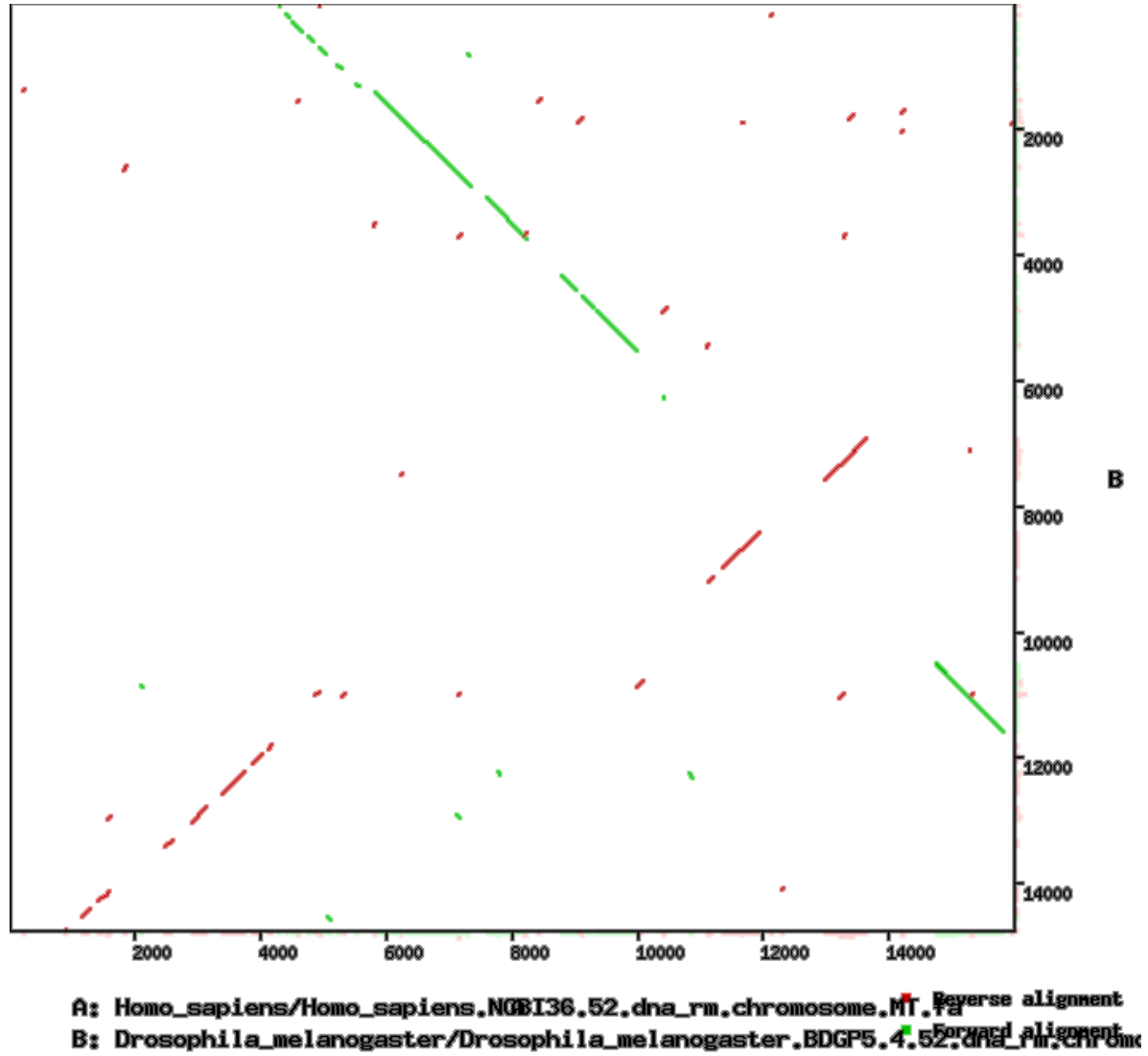
**Zarovnávanie sekvencií  
(cvičenie)**

**Broňa Brejová  
8.10.2021**

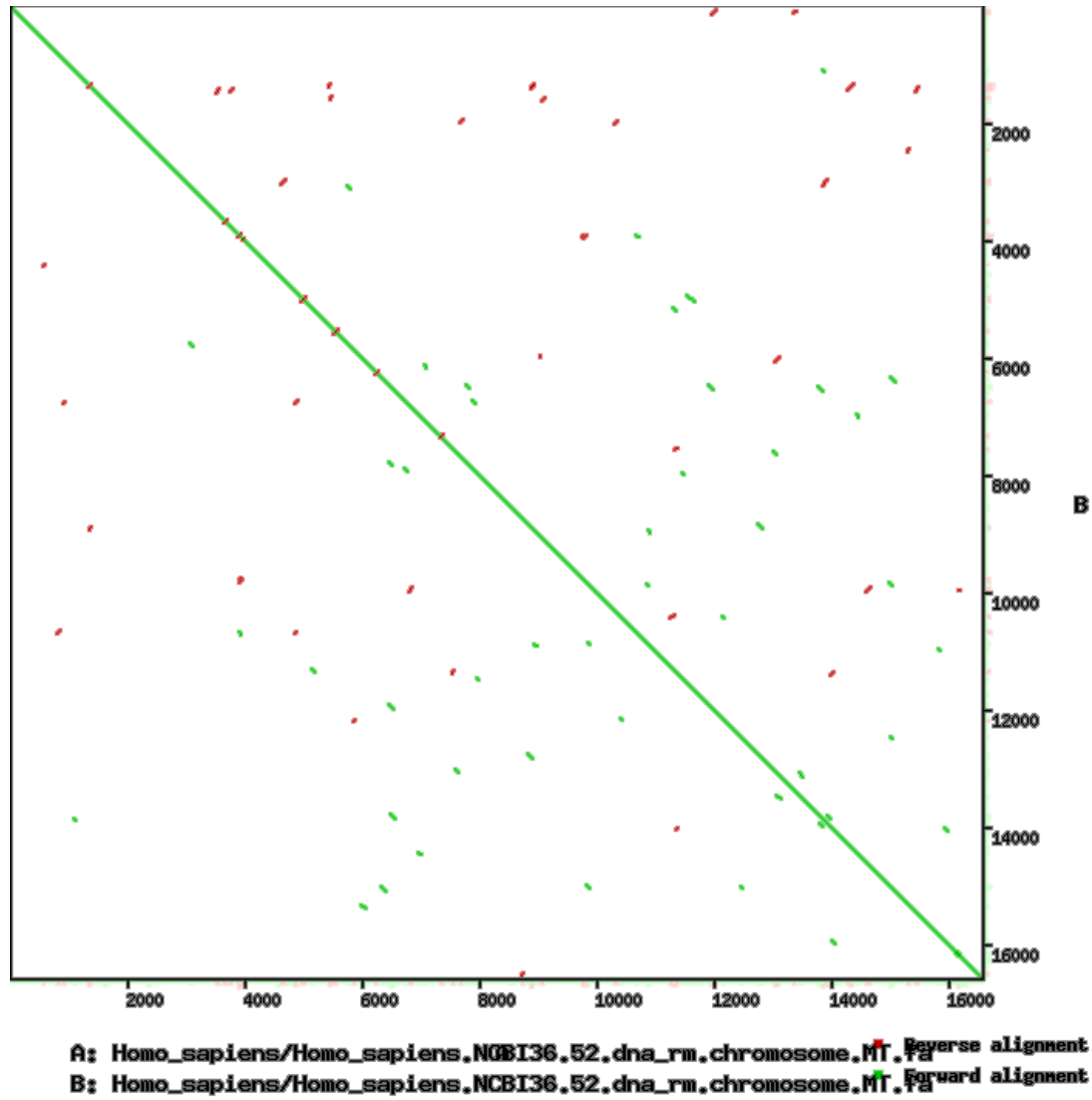
# Mitochondriálny genóm človeka vs. ryba *Danio rerio*

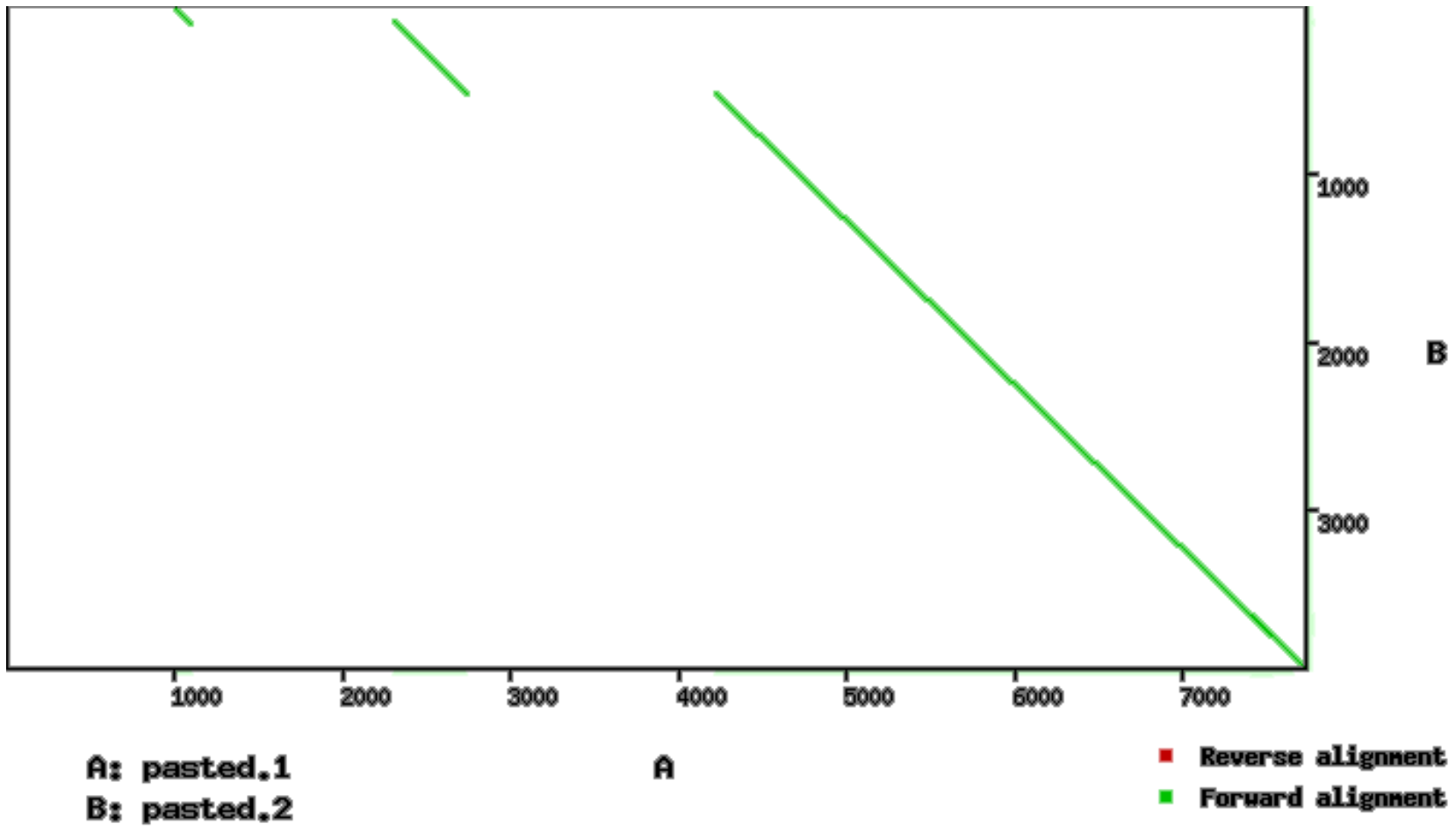


# Mitochondriálny genóm človeka vs. Drosophila melanogaster

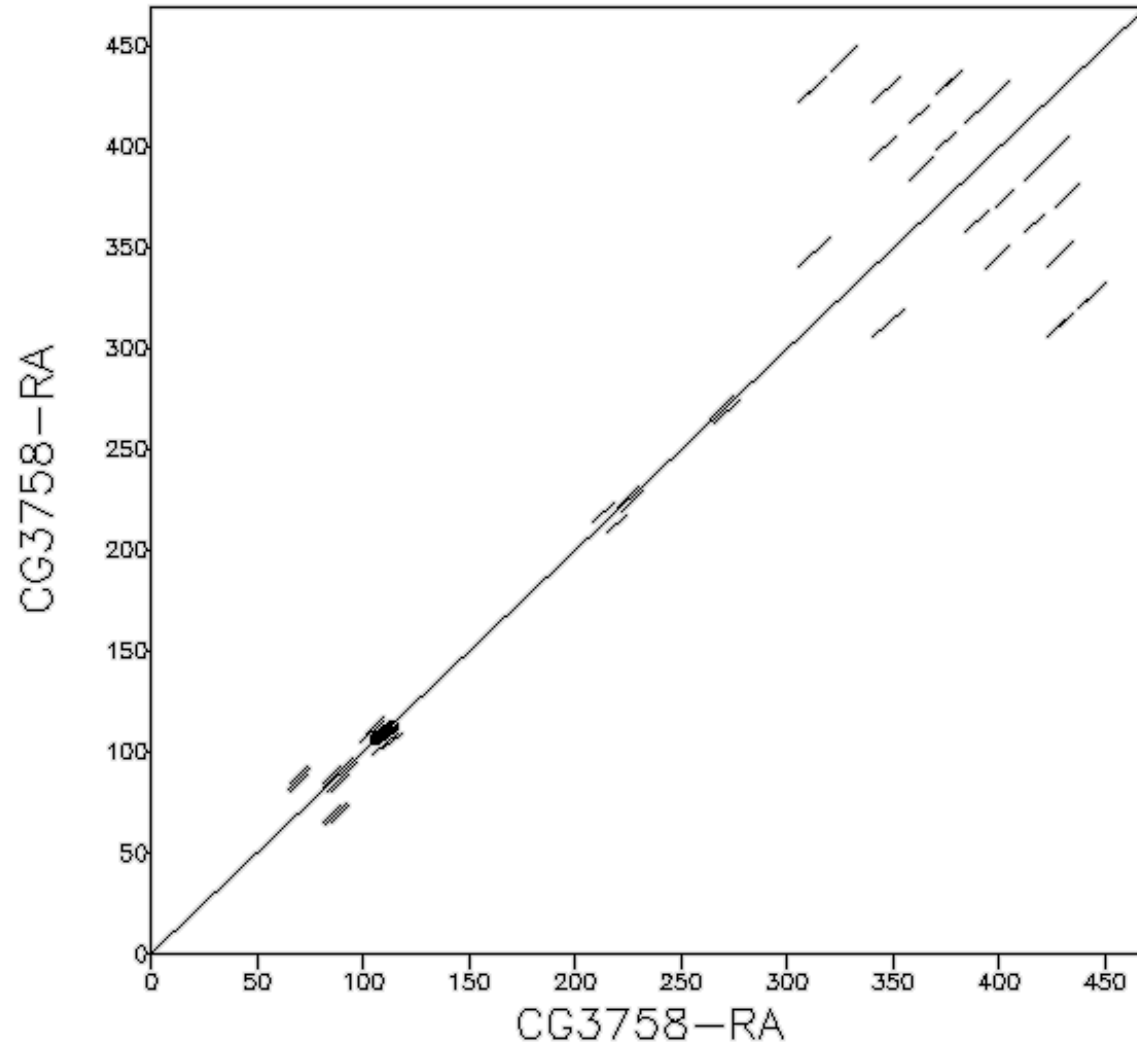


# Mitochondriálny genóm človeka vs. to isté





## Drosophila proteín Escargot zinc finger vs. to isté



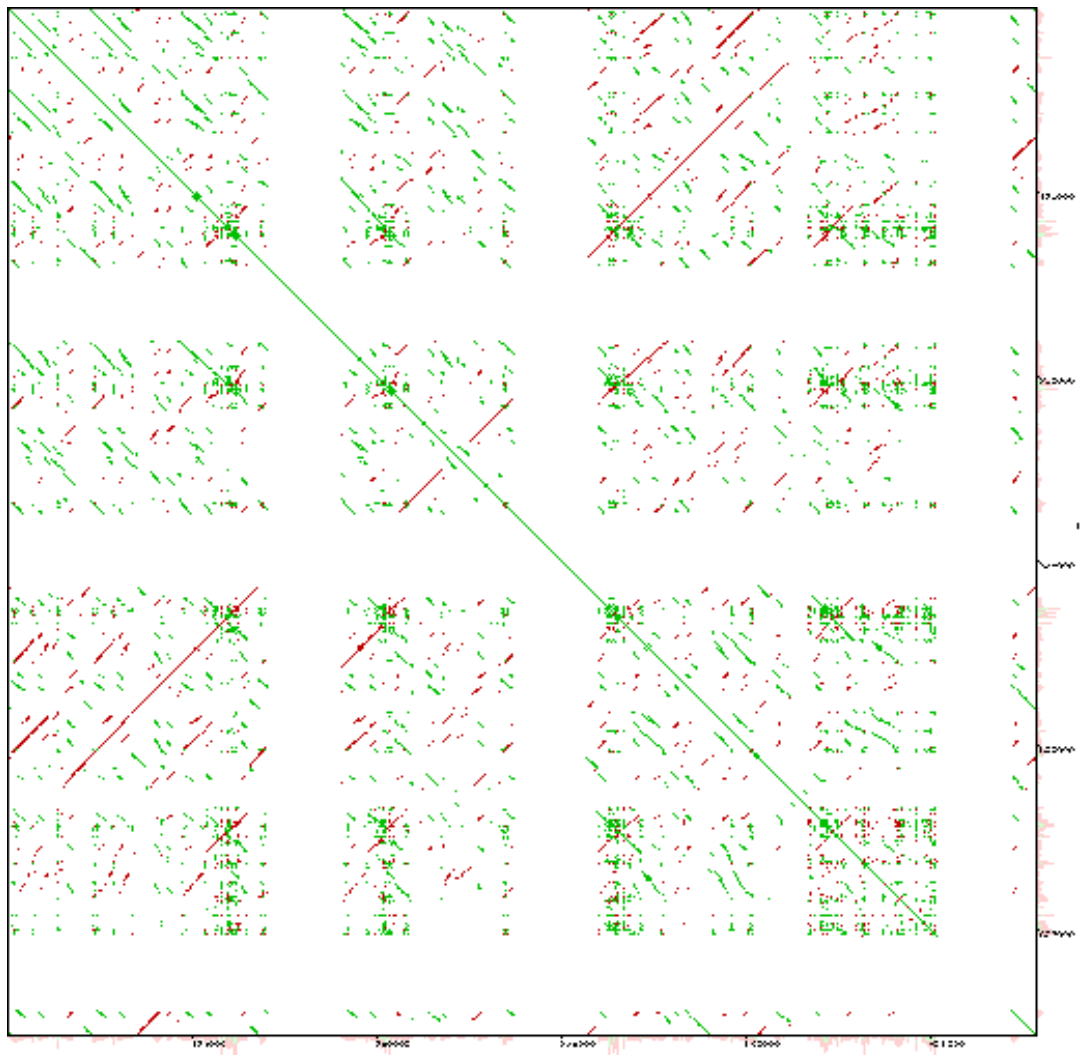
## Drosophila protein Escargot zinc finger

<b>Description:</b>	Protein escargot
<b>Source organism:</b>	<a href="#">Drosophila melanogaster (Fruit fly)</a> <a href="#">View Pfam proteome data.</a>
<b>Length:</b>	470 amino acids

### Pfam domains



Source	Domain	Start <sup>▲</sup>	End
Pfam B	<a href="#">Pfam-B 18487</a>	8	270
low_complexity		71	95
low_complexity		101	129
low_complexity		163	177
low_complexity		244	263
low_complexity		264	274
<b>Pfam A</b>	<a href="#">zf-C2H2</a>	309	332
<b>Pfam A</b>	<a href="#">zf-C2H2</a>	344	366
<b>Pfam A</b>	<a href="#">zf-C2H2</a>	370	392
<b>Pfam A</b>	<a href="#">zf-C2H2</a>	398	420
low_complexity		445	460





**Pravdepodobnosť a E-value  
(cvičenie)**

**Broňa Brejová  
24.10.2019**

## Hračkársky prípad

**Dotaz:** ATGCTCAAAC (dĺžka  $m = 10$ )

**Databáza:** (dĺžka  $n = 300$ )

accacttgcgcacgatttccagattcggtttccctggggcgcacgaagggc  
ccacgaagcgGCTCAAACccggagccttagttagaaggggggtctccgtca  
agagagacggtaagttggaggggtcactagcgggtggactccgaatggaaac  
actgaatagtggcagAACcctaaacctcgttttggatttcctgaaaaaggc  
aggcgctagaggaagaggcacgactgtgctagagataatcacttgtaaga  
ccttgggggatgggcttcgtatgcagaacgcgataaggtatcgaaaacgtg

**Skórovacia schéma:** zhoda  $+1$ , nezhoda  $-1$ , medzera  $-1$

**Lokálne zarovnanie** so skóre  $S = 6$

GCTCAAAC

GCTCA-AC

**E-value:** koľko očakávame lokálnych zarovnaní so skóre aspoň  $S$  v náhodnej databáze dĺžky  $n$  pri náhodnom dotaze dĺžky  $m$

## Náhodný dotaz a dazabáza

Dotaz: GTGCCTGCAG

Databáza:

cctctgatagccttgaaccgggcgagactcatacagacagtgctcctcgg  
gcgataaccatgagatgacaggtccgatgctaataacggacctacag  
tgacatgttaaagtgtccattaagtttataaccggaatcaacgagtgccc  
ccagcgcggcgaccgatggagccCCTGCAGgtatactcacttcaaggatt  
accgctcggtgtaagttagtggtcagtcagactataactaagtattcagtt  
atagagcgttagtaggtcgaccatgagcgggtaggGTGCCGAGatgtgaa

Počet výskytov: 2

## Náhodný dotaz a dazabáza

Dotaz: TCGACCGAAA

Databáza:

```
tactccattagggattataacgactaaagcccgtcgtggcgggatcactt  
tgagattcaactttaacgcatcacagaggaatctgagacaaagcaaaacc  
gatcataatgatcgatccaggtaataagtctccttgatggcgttagactg  
gaaataacagttgacttccgactatagtttaatgaacgttcgttaattaga  
cgatcgtgtaacttaaccaaggctgccccaaactagctgagtaatagc  
tcgtcctgagcatgtaagagtcagcctccacggaacactgcaacgttctt
```

Počet výskytov: 0

## Náhodný dotaz a dazabáza

Dotaz: CCCGTCGTAG

Databáza:

cagcattagccccgttat`ttCGTCGT`tctccaacgggtctgcctttctgg  
aacgtggcgaaccttcacaggtcagtcctgtcatcgccctgcgcttagagcg  
gacggtactcgaaaggtcgggttcagtggtggcgctggaaagaagaatagca  
acacatgcactaatggaaggtcccagtggtgtgggacattctgga`CCCGT`  
`GT`gtgccaacctatgtgagctccggcggtgactcggaggatgttaacaag  
atcaagctgtaggcgacgatccccgcccgggtttcctctactgcctcgagc

Počet výskytov: 2

## Náhodný dotaz a dazabáza

Dotaz: AGGATGAGGA

Databáza:

ttatcgattctccggtgcgccagtacagcacaaggctcggatcctgtaaa  
acactacaccttaaaaactaagtcAGGATGtgatctcccttaaGATGAGa  
cagtctctaatagcggcgtagtgggaccctcgtgaccgagctaagcagttc  
acaatgggcgctctgagcgattggctggagaccttgacttcccggtaggt  
gtggtgtagttctgtgccagagataaccatccaccgtaatggatctcg  
taactttacGATGAAGA ccggcatcatctcagttatatttctaggacggg

Počet výskytov: 3

## Celkovo opakujeme 100 krát

$S = 6, m = 10, n = 300$ , obsah GC 50%

Počet výskytov: 2, 0, 2, 3, 3, 1, 0, 1, 1, 1, 0, 0, 4, 2, 0, 1, 0, 1, 0, 0, 1, 0, 0, 4, 3, 1,  
1, 0, 0, 0, 2, 3, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0, 4, 1, 1, 0, 0, 1, 1, 1, 2, 2, 2, 0, 0, 2, 0, 1, 1,  
0, 1, 2, 2, 1, 0, 0, 1, 1, 2, 0, 1, 0, 0, 1, 0, 3, 2, 0, 2, 2, 1, 0, 0, 2, 0, 0, 1, 2, 1, 1, 3, 2,  
2, 1, 1, 0, 2, 0, 1, 3

Priemerný počet výskytov: 1.05

Keď celé opakujeme viackrát, dostávame hodnoty 0.99, 1.15, 1.02, 1.07, 0.98, ...

Správna hodnota E-value: 0.99

# **Fylogenetické stromy**

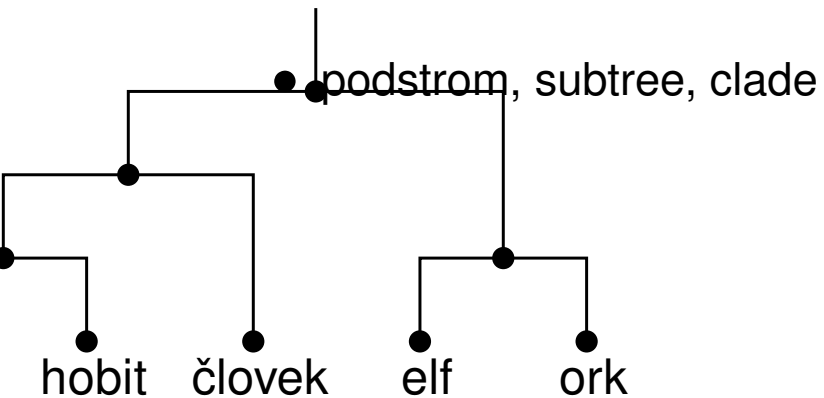
**Broňa Brejová**

**7.11.2022**



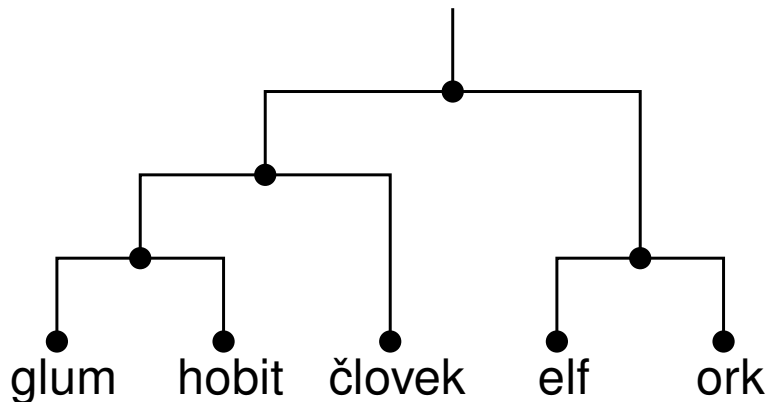
## Terminológia

- zakorenený strom, rooted tree
- nezakorenený strom, unrooted tree
- hrana, vetva, edge, branch
- vrchol, uzol, vertex, node
- list, leaf, leaf node, tip, terminal node
- vnútorný vrchol, internal node
- koreň, root



## Zopár faktov o stromoch

- Majme zakorenený strom s  $n$  listami, v ktorom má každý vnútorný vrchol 2 deti. Takýto strom vždy má  $n - 1$  vnútorných vrcholov a  $2n - 2$  vetiev (prečo?)
- Majme nezakorenený strom s  $n$  listami, v ktorom má každý vnútorný vrchol 3 susedov. Takýto strom vždy má  $n - 2$  vnútorných vrcholov a  $2n - 3$  vetiev.
- Koľkými spôsobmi môžeme zakoreniť nezakorenený strom s  $n$  listami?

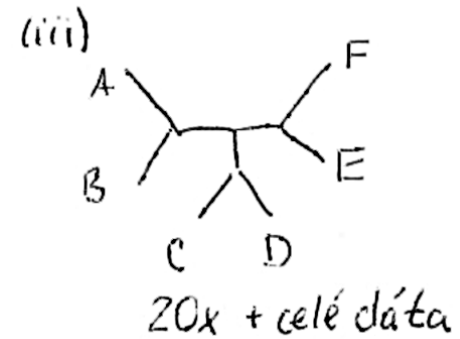
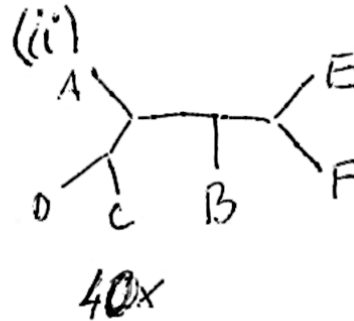
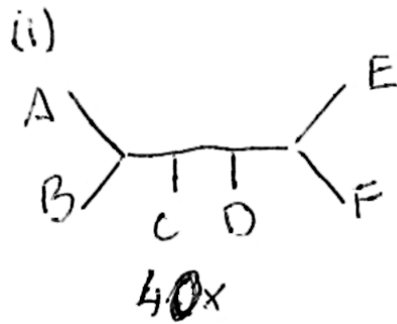


## Bootstrap

- Náhodne vyberieme niektoré stĺpce zarovnaní, zostrojíme strom
- Celé to opakujeme veľa krát
- Značíme si, koľkokrát sa ktorá hrana opakuje v stromoch  
(v nezakorenenom strome je hrana rozdelenie listov na dve skupiny)
- Nakoniec zostavíme strom z celých dát a pozrieme sa ako často sa ktorá jeho hrana vyskytovala
- Môžeme zostaviť aj strom z často sa vyskytujúcich hrán
- Bootstrap hodnoty sú odhadom spoľahlivosti, hlavne ak máme celkovo málo dát (krátke zarovnanie)
- Ak však dáta nezodpovedajú vybranej metóde/modelu, tak aj pre zlý strom môžeme dostať vysoký bootstrap

## Bootstrap

Robili sme  $100\times$  bootstrap, dostali sme tieto výsledky:



Doplňte bootstrap hodnoty hranám výsledného stromu (iii)

Ktoré ďalšie vetvy majú podporu aspoň 20%?

Aký strom by sme dostali, ak by sme chceli nechať iba vetvy s podporou aspoň 80%?

# **K-means clustering**

**Broňa Brejová**

**12.11.2020**

## Formulácia problému

**Vstup:**  $n$ -rozmerné vektory  $x_1, x_2, \dots, x_t$  a počet zhlukov  $k$

**Výstup:** Rozdelenie vektorov do  $k$  zhlukov:

- priradenie vstupných vektorov do zhlukov zapísané ako čísla  $c_1, c_2, \dots, c_t$ , kde  $c_i \in \{1, 2, \dots, k\}$  je číslo zhuku pre  $x_i$
- centrum každého zhuku, t.j.  $n$ -rozmerné vektory  $\mu_1, \mu_2, \dots, \mu_k$

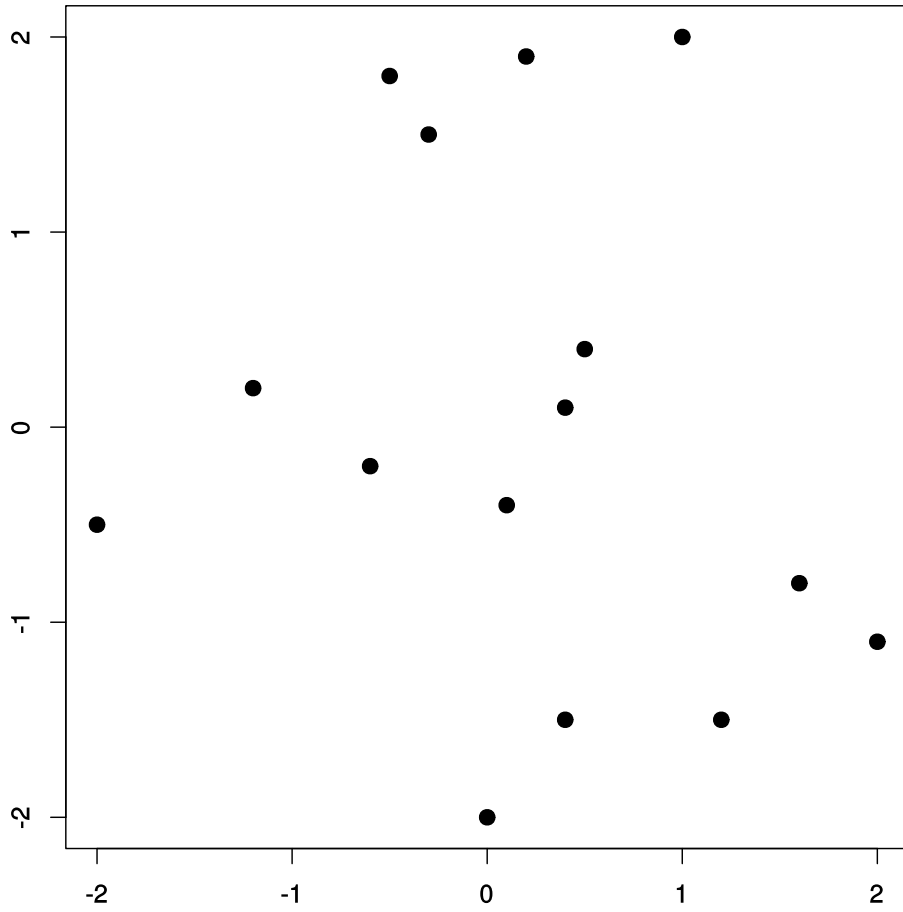
Hodnoty  $c_1, \dots, c_t$  a  $\mu_1, \dots, \mu_k$  volíme tak, aby sme minimalizovali súčet štvorcov vzdialeností od každého vektoru k centru jeho zhuku:

$$\sum_{i=1}^t \|x_i - \mu_{c_i}\|_2^2$$

Pre vektory  $a = (a_1, \dots, a_n)$  a  $b = (b_1, \dots, b_n)$  je druhá mocnina vzdialenosti  $\|a - b\|_2^2 = \sum_{i=1}^n (a_i - b_i)^2$

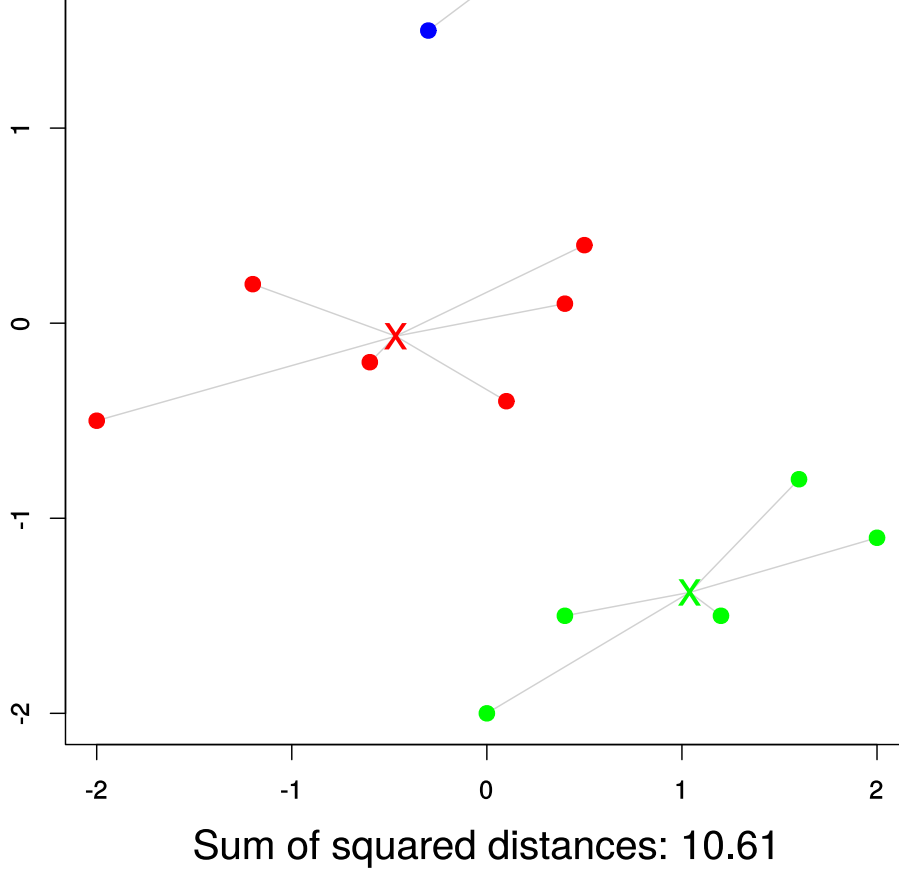
## Príklad vstupu

$x_1$  -2.00 -0.50  
 $x_2$  -1.20 0.20



### Príklad výstupu

$x_1$	-2.00	-0.50	1
$x_2$	-1.20	0.20	1
$x_3$	-0.60	-0.20	1
$x_4$	-0.50	1.80	3
$x_5$	-0.30	1.50	3
$x_6$	0.00	-2.00	2
$x_7$	0.10	-0.40	1
$x_8$	0.20	1.90	3
$x_9$	0.40	0.10	1
$x_{10}$	0.40	-1.50	2
$x_{11}$	0.50	0.40	1
$x_{12}$	1.00	2.00	3
$x_{13}$	1.20	-1.50	2
$x_{14}$	1.60	-0.80	2
$x_{15}$	2.00	-1.10	2
$\mu_1$	<b>-0.47</b>	<b>-0.07</b>	
$\mu_2$	<b>1.04</b>	<b>-1.38</b>	
$\mu_3$	<b>0.10</b>	<b>1.80</b>	





## Algoritmus

Heuristika, ktorá nenájde vždy najlepšie zhlukovanie.

Začne z nejakého zhlukovania a postupne ho zlepšuje.

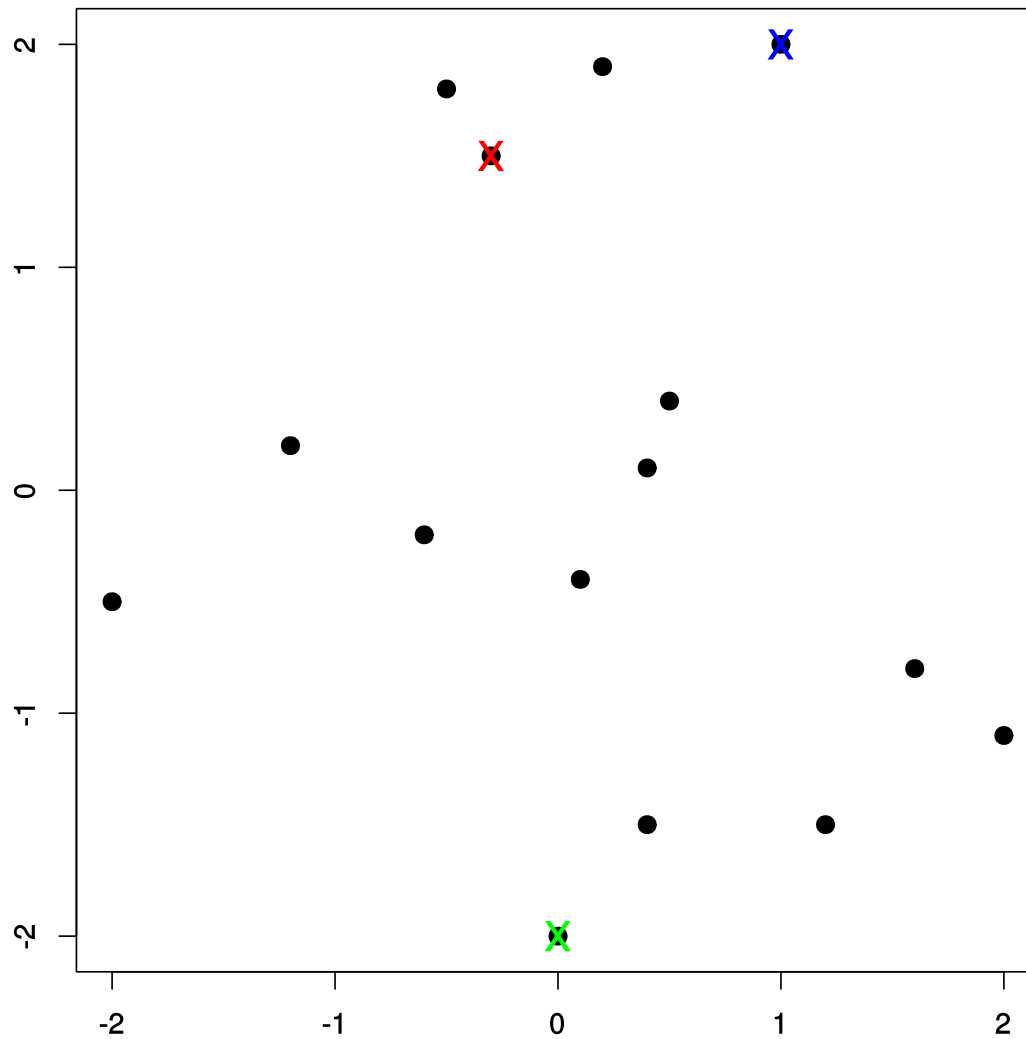
### Inicializácia:

náhodne vyber  $k$  centier  $\mu_1, \mu_2, \dots, \mu_k$  spomedzi vstupných vektorov

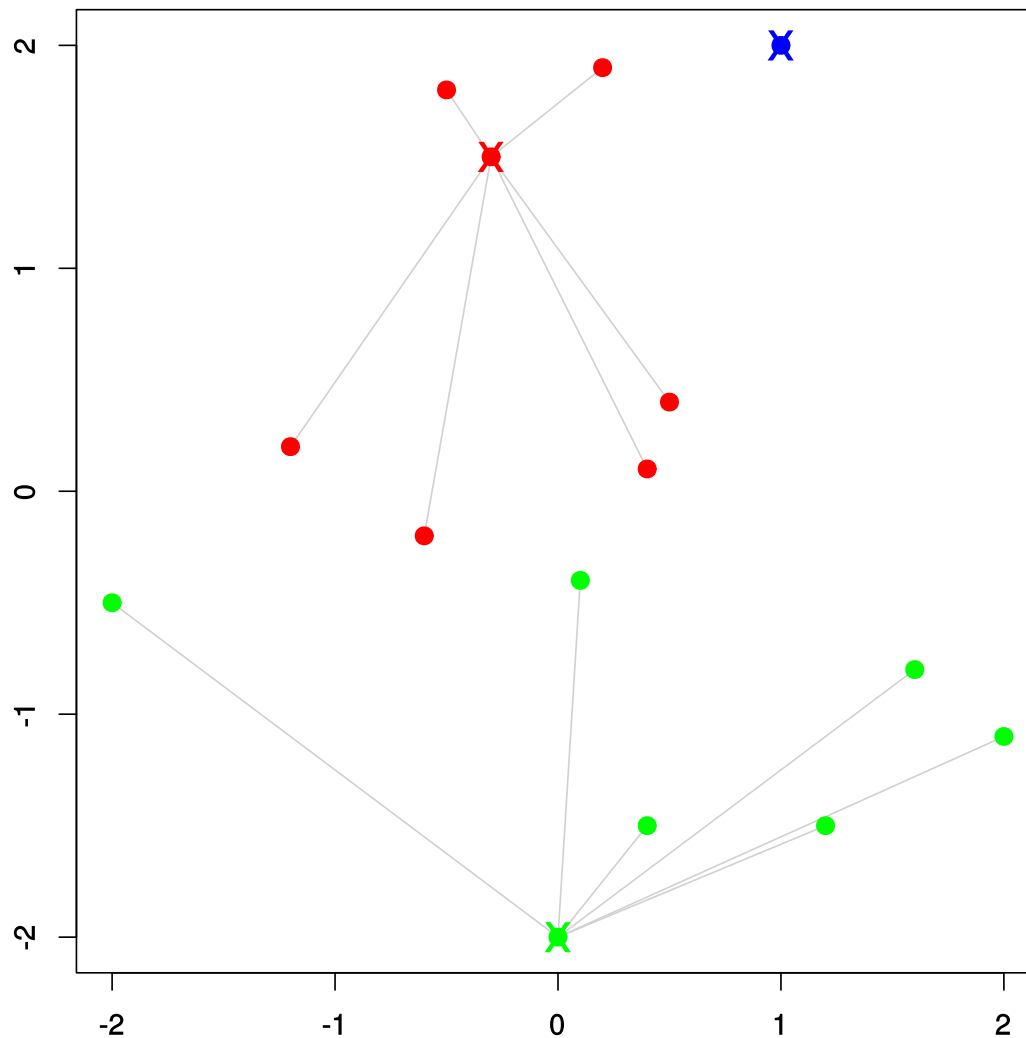
### Opakuj, kým sa niečo mení:

- prirad' každý bod najbližšiemu centru:  $c_i = \arg \min_j \|x_i - \mu_j\|_2$
- vypočítaj nové centroidy:  $\mu_j$  bude priemerom (po zložkách) z vektorov  $x_i$ , pre ktoré  $c_i = j$

Zvolíme náhodné centrá  $\mu_i$

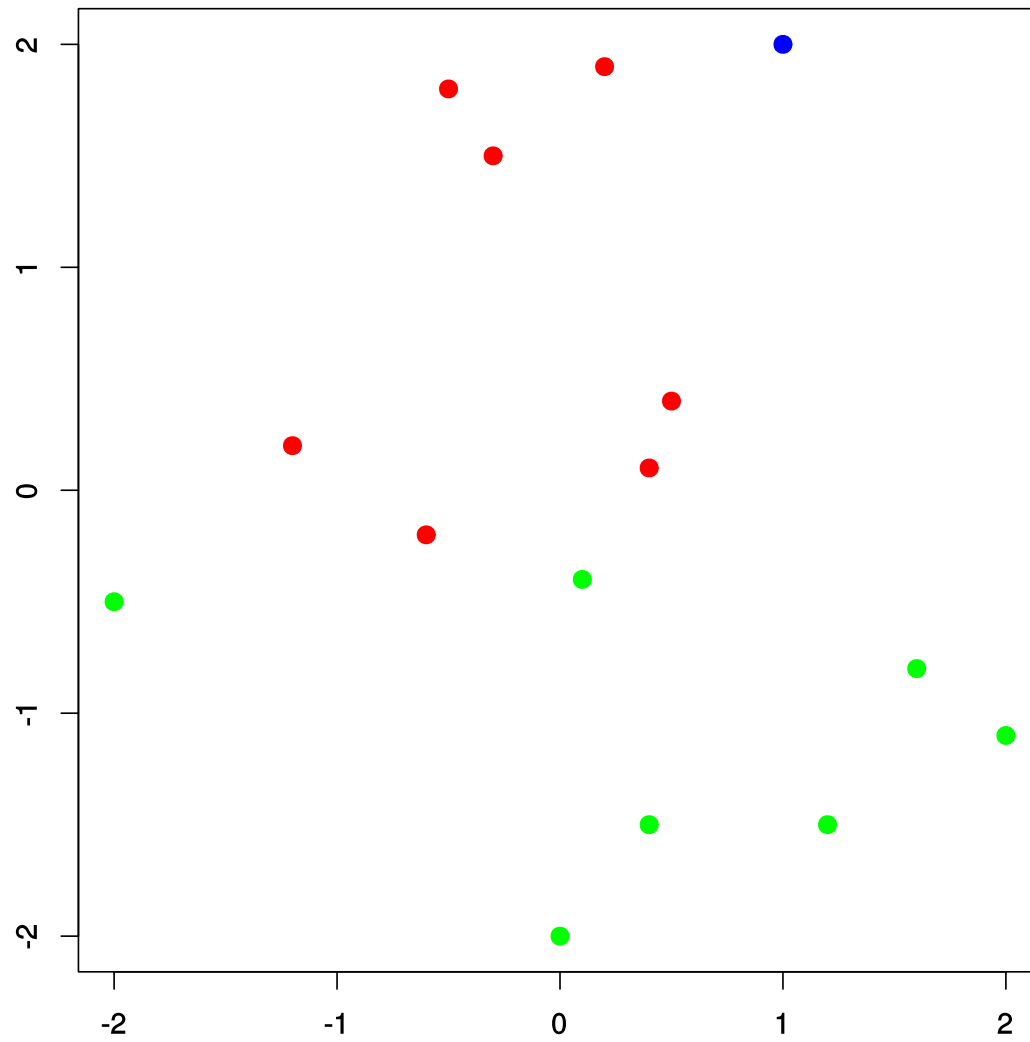


### Vektory priradíme do zhlukov (hodnoty $c_i$ )

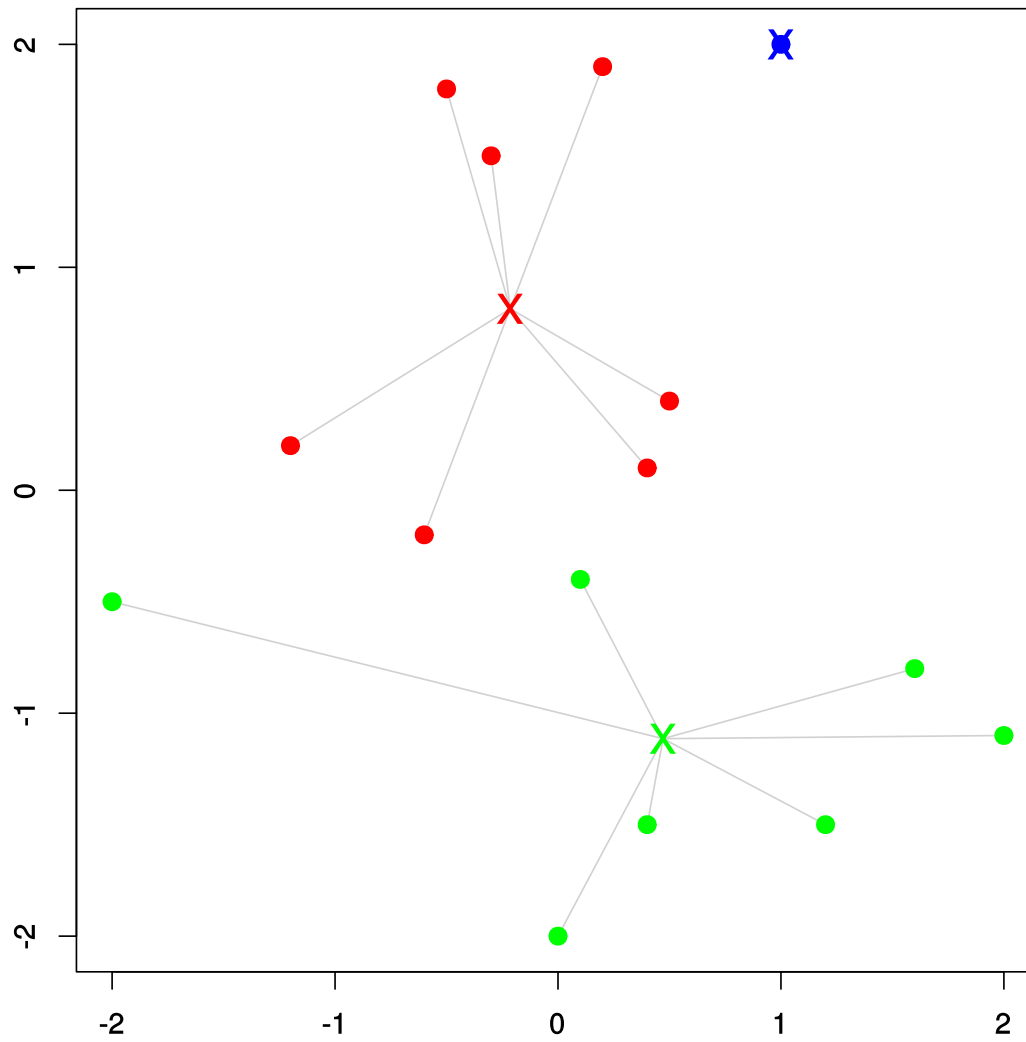


Sum of squared distances: 30.05

Zabudneme  $\mu_i$

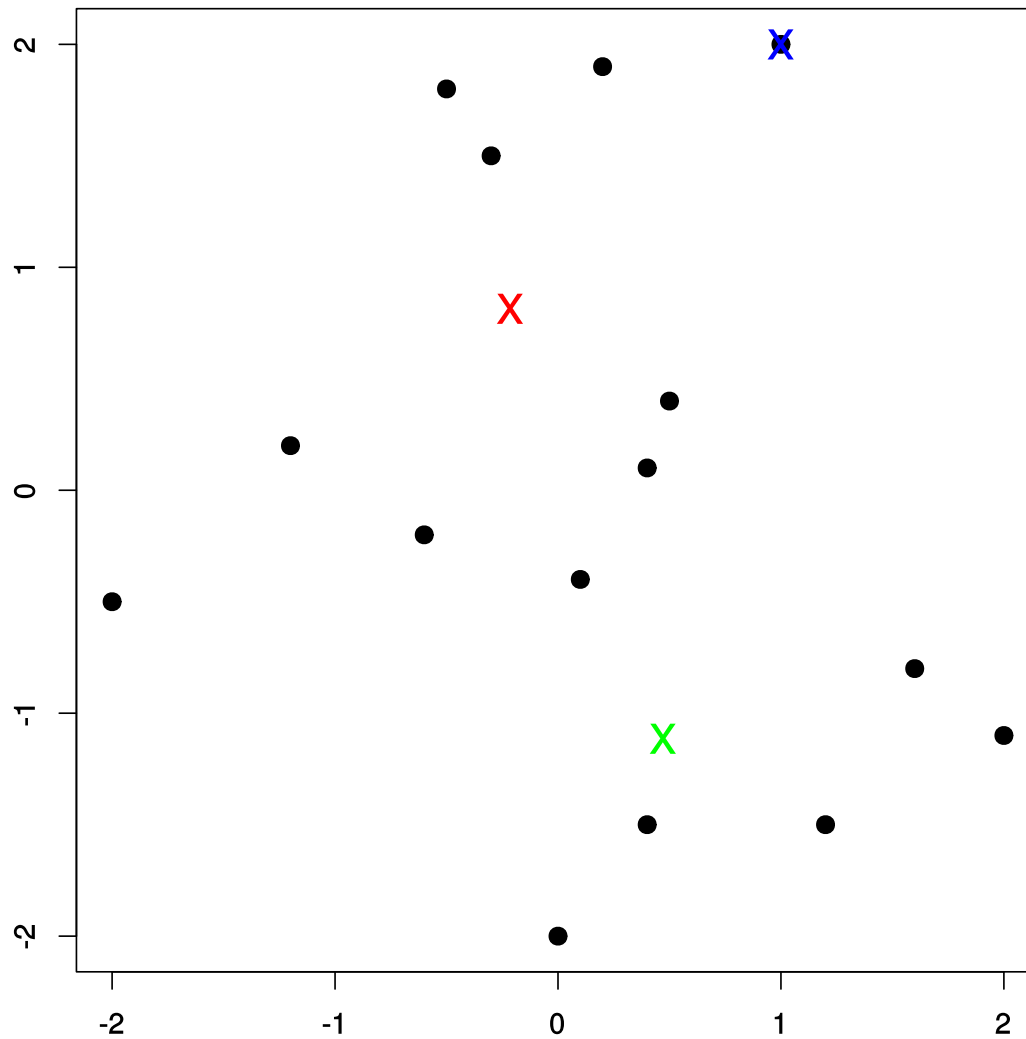


Dopočítame nové  $\mu_i$  (suma klesla z 30.05 na 19.66)

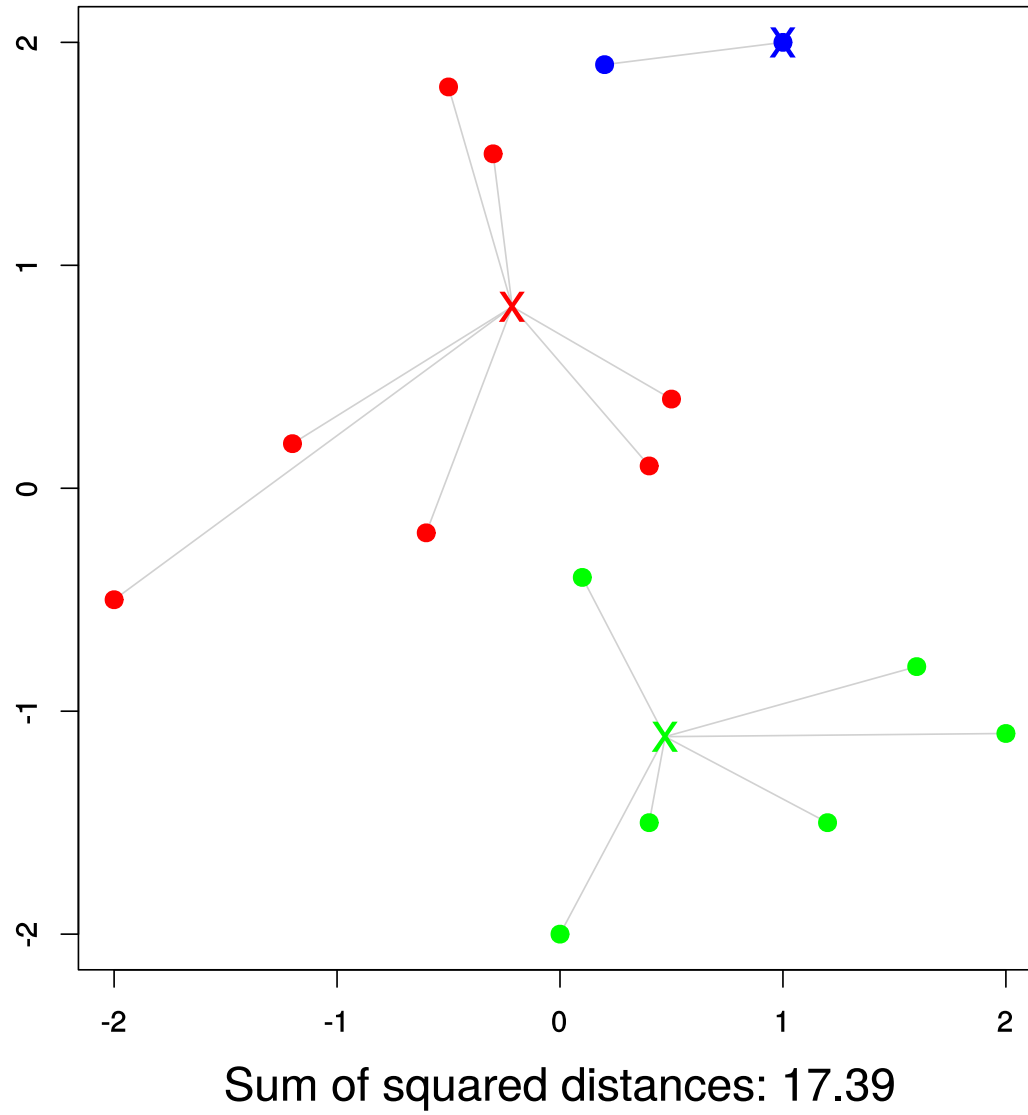


Sum of squared distances: 19.66

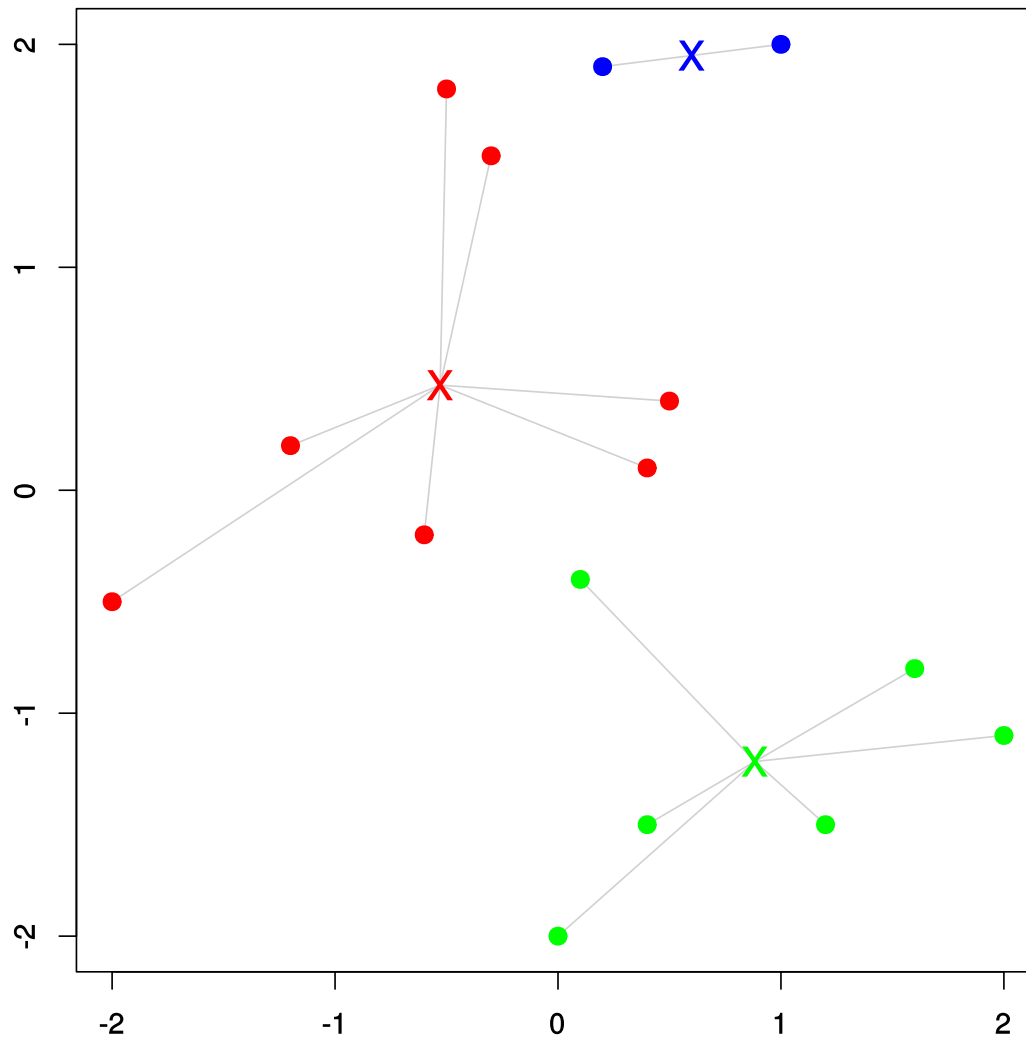
Zabudneme  $C_i$



Dopočitame nové  $c_i$  (suma klesla z 19.66 na 17.39)



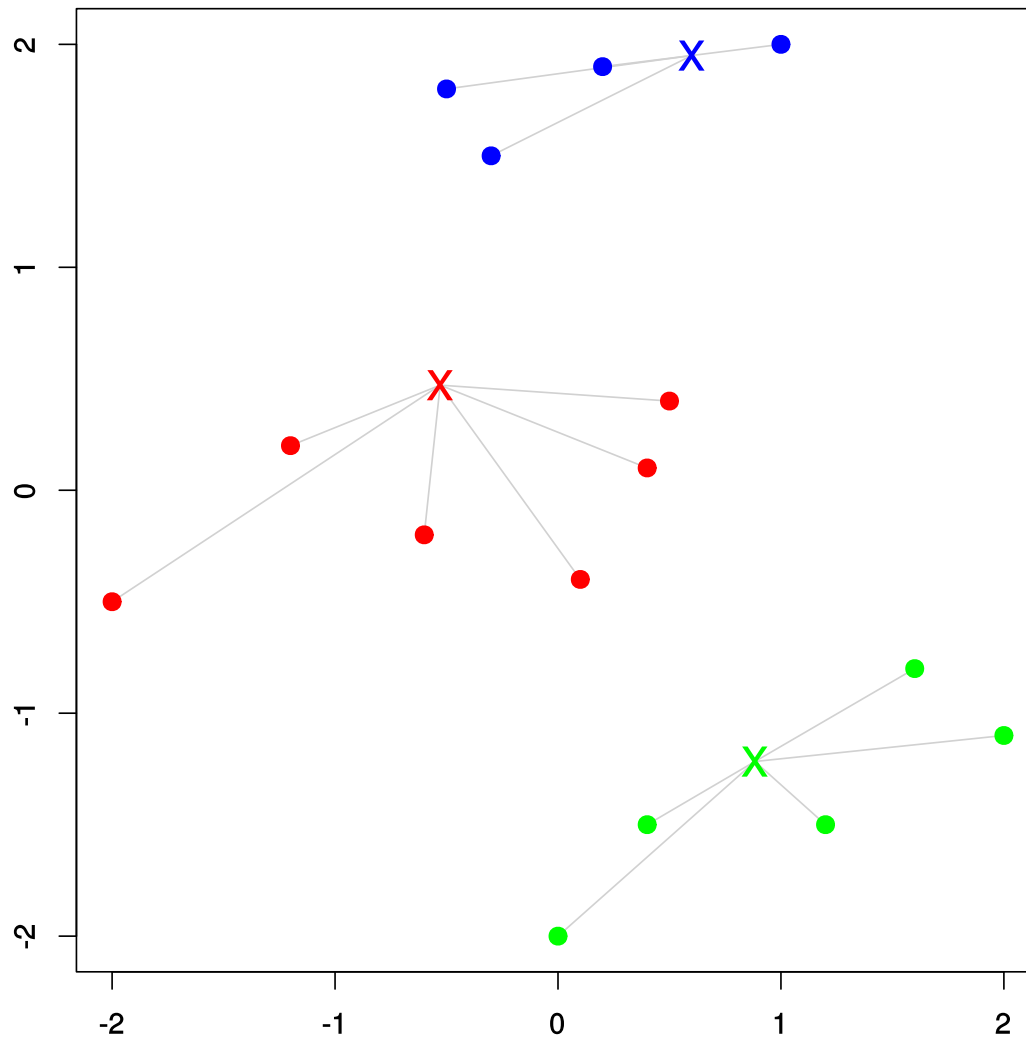
Prepočítame  $\mu_i$



Sum of squared distances: 14.47

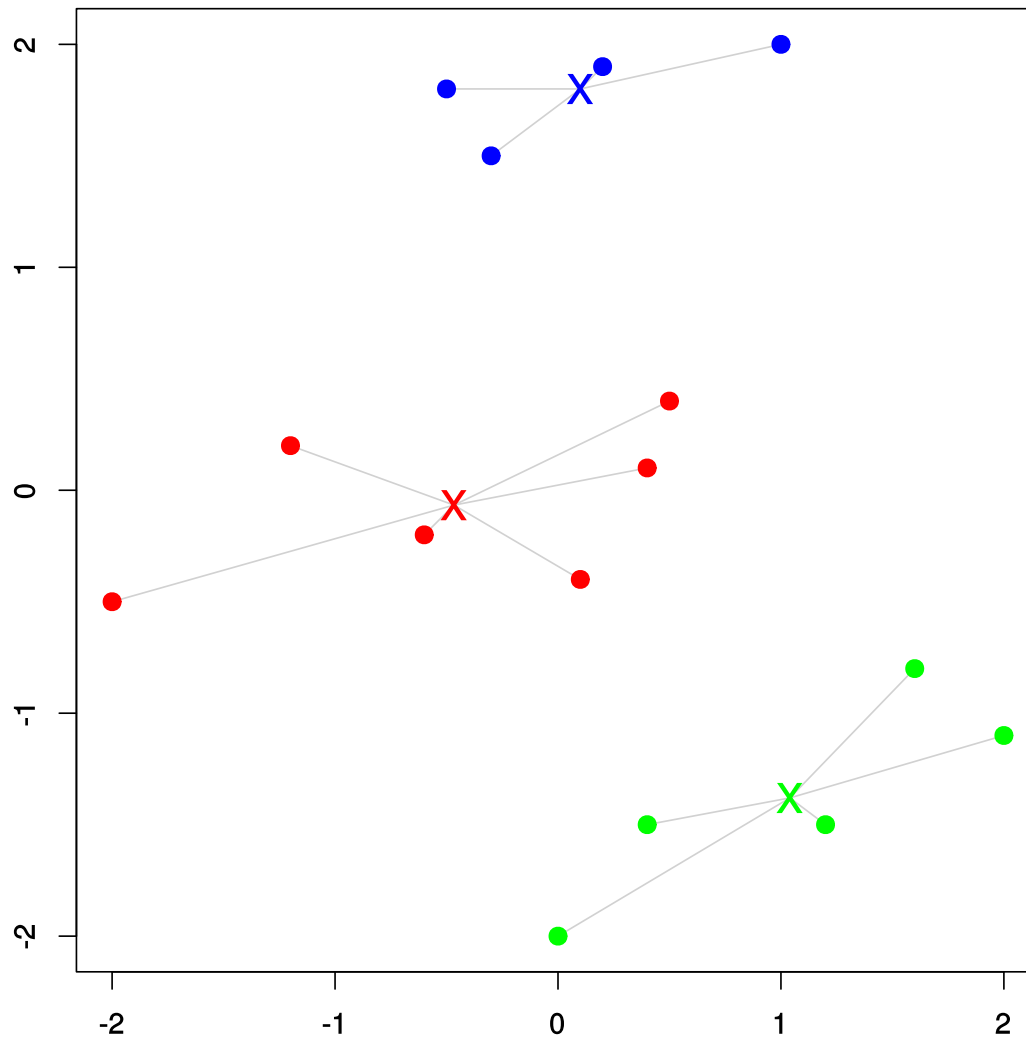


Prepočítame  $C_i$



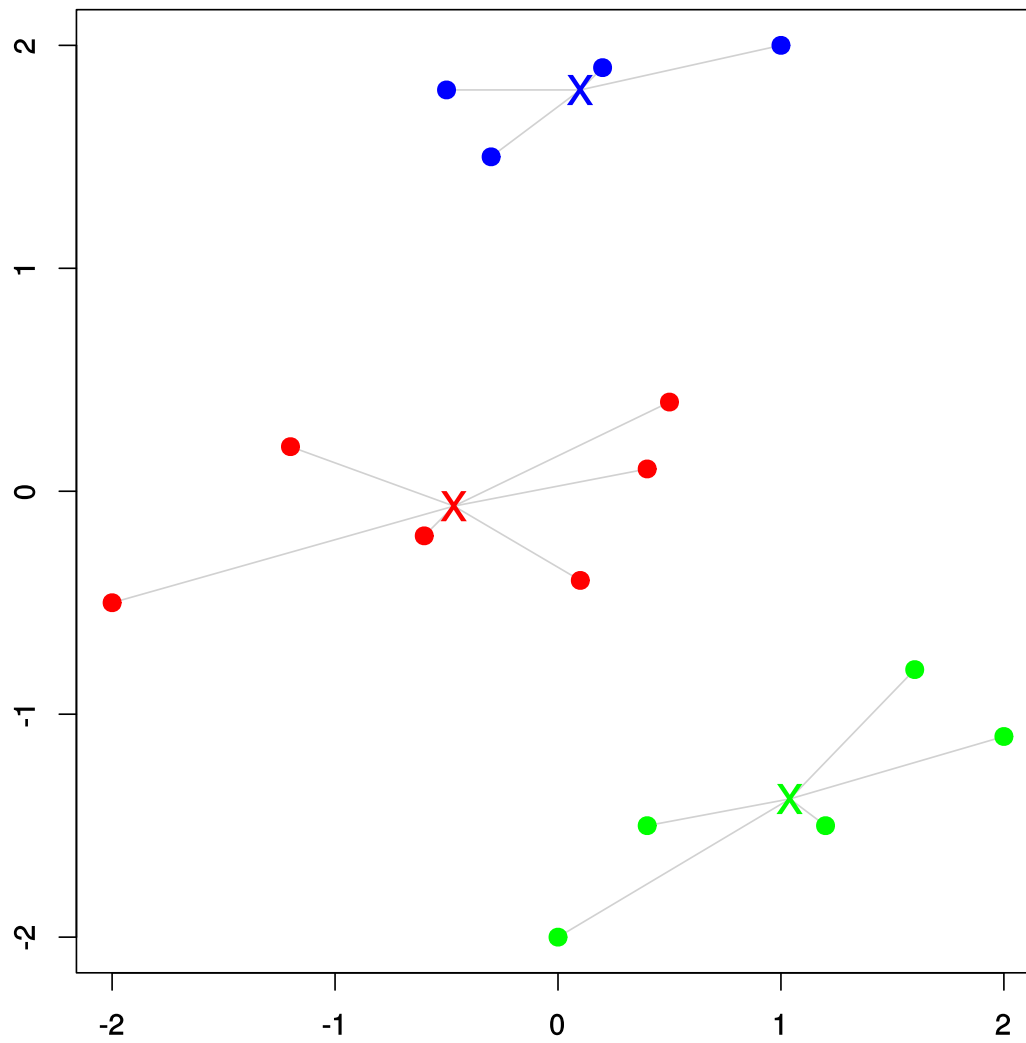
Sum of squared distances: 13.71

Prepočítame  $\mu_i$



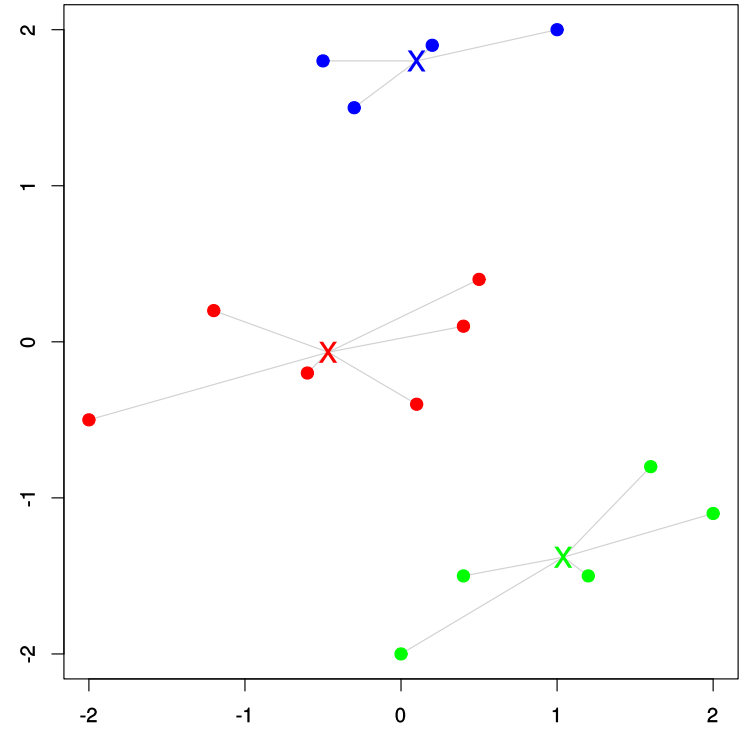
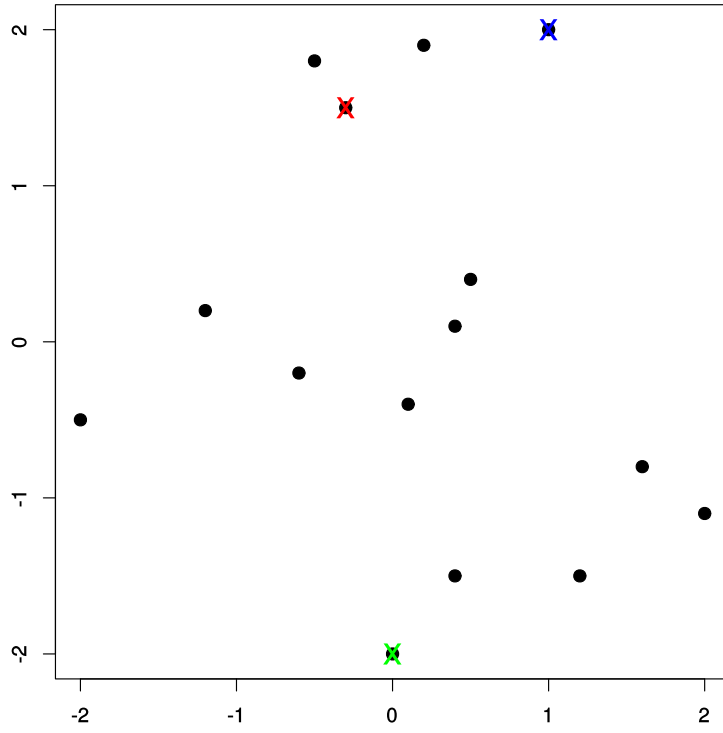
Sum of squared distances: 10.61

Prepočítame  $c_i$  (žiadna zmena, končíme)



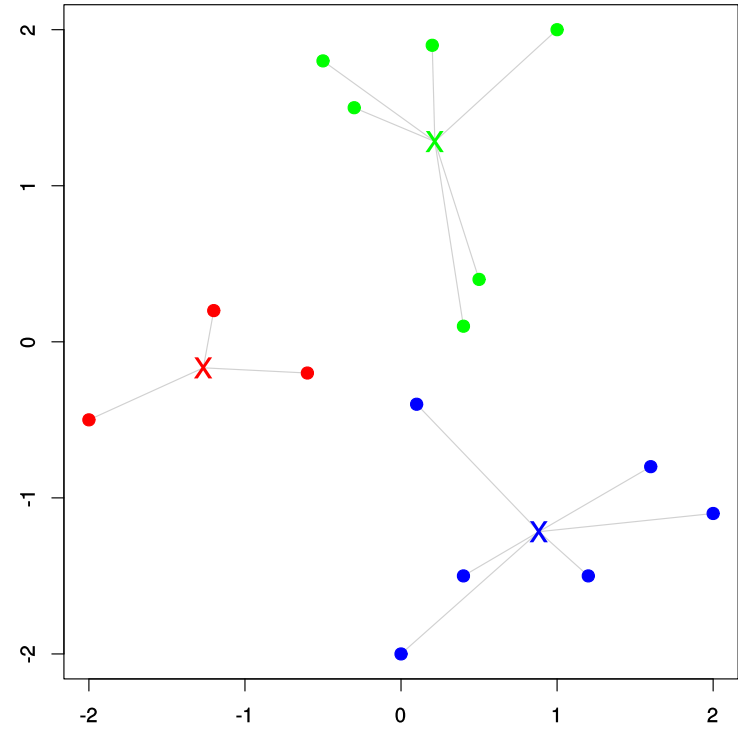
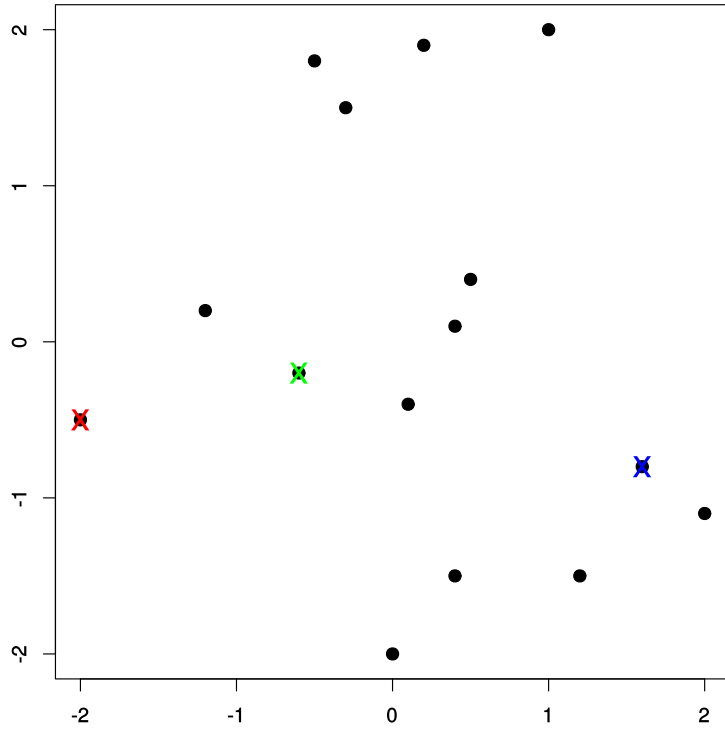
Sum of squared distances: 10.61

# Príklady niekoľkých behov programu



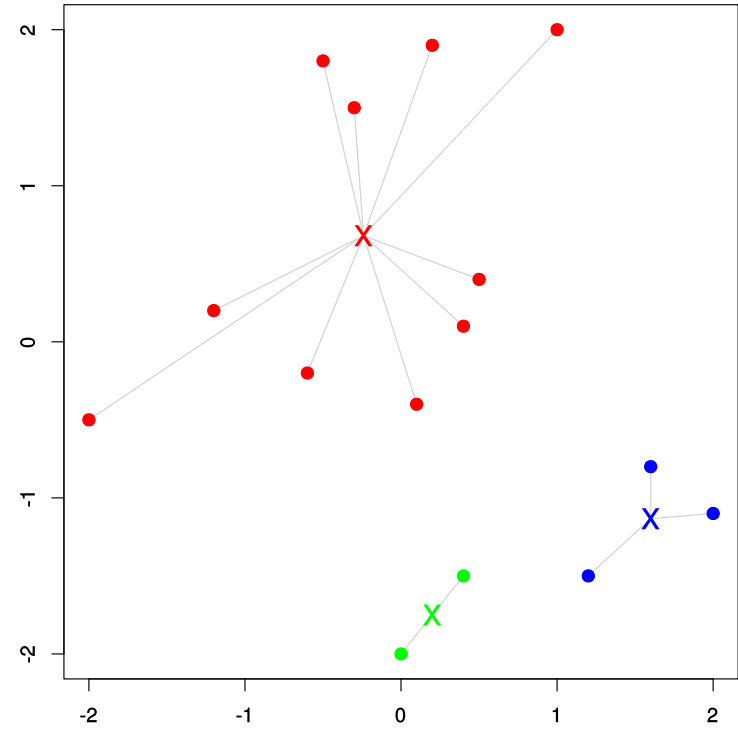
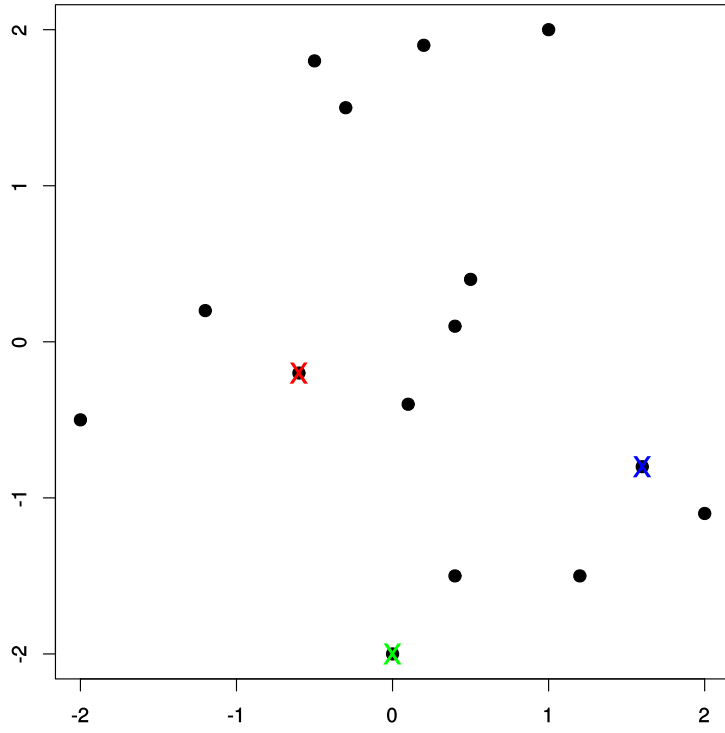
Sum of squared distances: 10.61

# Príklady niekoľkých behov programu



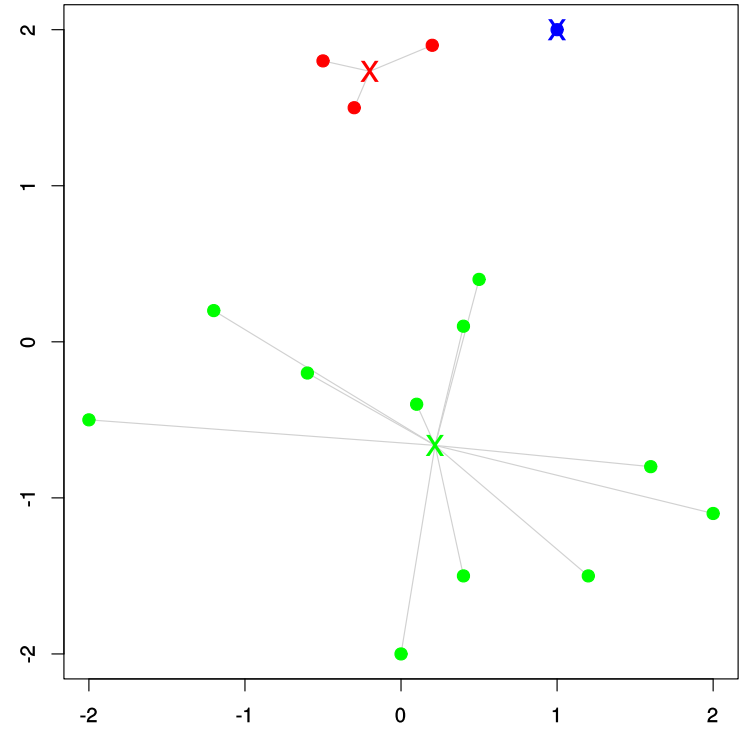
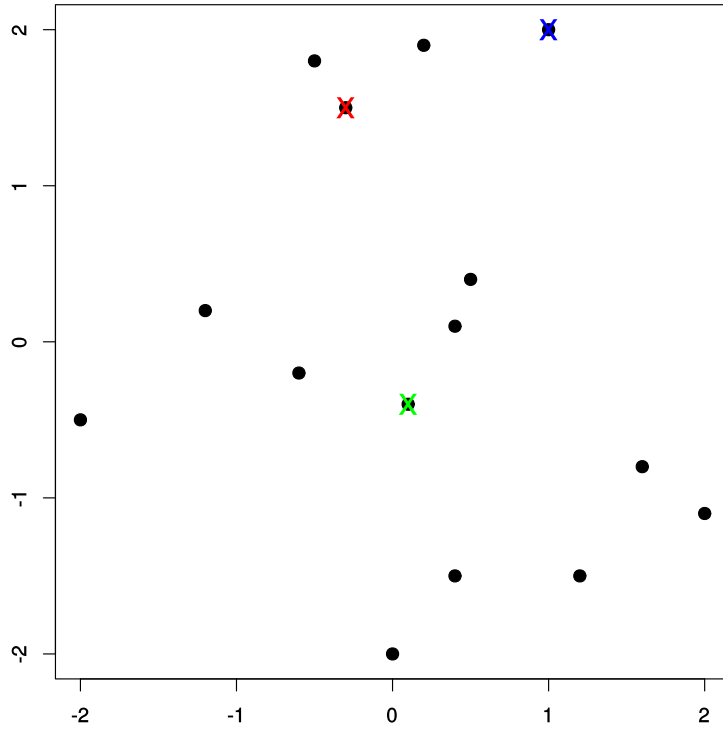
Sum of squared distances: 11.25

# Príklady niekoľkých behov programu



Sum of squared distances: 16.93

# Príklady niekoľkých behov programu



Sum of squared distances: 20.37

# **GO Enrichment**

**Broňa Brejová**

**12.11.2020**



## Gene list analysis

Many analyses yield lists of genes. Examples:

- genes with positive selection in comparative genomics
- overexpressed or underexpressed genes in expression analysis
- genes regulated by a specific transcription factor

Some of genes in a list will have a known function, others may be less studied

## What to do with such a gene list?

- Look at several interesting candidates and study them in detail (bioinformatics / wet lab)
- Determine if the whole set is enriched in genes with some property
  - for example, genes under positive selection are often enriched for functions in immunity
  - this is caused by evolutionary pressure from pathogens

## Example from Kosiol et al 2008

16,529 genes total

70 genes innate immune response (0.4% of all genes)

400 genes positive selection

8 genes positive selection + innate immune response (2% of pos. sel.)

### Contingency table

	Pos.sel.	No pos.sel.	Total
Immunity	8 ( $n_{ip}$ )	62	70 ( $n_i$ )
Not immunity	392	16067	16459
Total	400 ( $n_p$ )	16129	16529 ( $n$ )

### Observations:

Innate immune response only a small fraction of pos.sel.

But large enrichment from 0.4% to 2%

Is it by chance (due to small numbers)?

## Example from Kosiol et al 2008

---

	Pos.sel.	No pos.sel.	Total
Immunity	8 ( $n_{ip}$ )	62	70 ( $n_i$ )
Not immunity	392	16067	16459
Total	400 ( $n_p$ )	16129	16529 ( $n$ )

---

Is enrichment due to chance?

### Want p-value:

What would be a chance of obtaining such an enrichment if positive selection and role in innate immune response independent (null hypothesis)

## Null hypothesis

---

	Pos.sel.	No pos.sel.	Total
Immunity	8 ( $n_{ip}$ )	62	70 ( $n_i$ )
Not immunity	392	16067	16459
Total	400 ( $n_p$ )	16129	16529 ( $n$ )

---

Urn with  $n_i = 70$  white balls and  $n - n_i = 16459$  black balls

Draw  $n_p = 400$  balls from the urn

Denote by  $X$  the number of white balls in the selection

On average we expect  $E(X) = n_p(n_i/n) = 1.7$

In reality we see  $n_{ip} = 8$  pos. sel. genes with role in innate immunity

This is  $4.7\times$  more

How likely is this by chance?

## Null hypothesis

Urn with  $n_i = 70$  white balls and  $n - n_i = 16459$  black balls

Draw  $n_p = 400$  balls from the urn

Denote by  $X$  the number of white balls in the selection

Variable  $X$  has **hypergeometric distribution**:

$$\Pr(X = n_{ip}) = \frac{\binom{n_i}{n_{ip}} \binom{n - n_i}{n_p - n_{ip}}}{\binom{n}{n_p}}$$

P-value is  $\Pr(X \geq n_{ip}) = \Pr(X = n_{ip}) + \Pr(X = n_{ip} + 1) + \dots$

Tail of the distribution

In our case  $\Pr(X \geq 8) = 0.00028$

This is called **Hypergeometric** or Fisher's exact test

It can be approximated by  $\chi^2$  test

## Multiple testing correction

Often we do many tests of the same type, for example

- Test 1000 genes for positive selection, select those with p-value  $\leq 0.05$
- Test enrichment of 1000 functional categories in a list of genes, select those with p-value  $\leq 0.05$

**Problem:** If each category has 5% chance of being there by chance, we expect 50 purely random results.

If the total number of positive tests was 100, half of them were false.

**Multiple testing correction:** lower threshold on p-value so that false positives do not constitute a large portion of results

Several techniques, e.g. FDR (false discovery rate)

**Cvičenia pre biológov, 10.12.2020**

**Zhrnutie semestra**



## Tvorba bioinformatického nástroja

- Sformulujeme biologické ciele  
(aké máme dáta, aké typy otázok sa chceme pýtať).
- Sformulujeme informaticky/matematicky  
(napr. ako pravdepodobnostný model).  
Dostaneme informatické zadanie problému, v ktorom je presne daný vzťah medzi vstupom a želaným výstupom  
(napr. nájsť zarovnanie s max. skóre v určitej skórovacej schéme).
- Hľadáme efektívne algoritmy na riešenie informatického problému.
- Ak sa nám nepodarí nájsť dosť rýchly algoritmus, použijeme heuristiky, ktoré dávajú približné riešenia.
- Testujeme na reálnych dátach, či sú výsledky biologicky správne  
(či bol model dobre zvolený, či heuristiky dobre fungujú).

## Použitie bioinformatického nástroja

- Sformulujeme biologické ciele (aké máme dáta, aké typy otázok sa chceme pýtať).
- Porozmýšľame, aký typ nástroja, resp. ich kombinácia by nám mohli pomôcť
- Alebo hľadáme v literatúre nástroj na typ problému, s ktorým sme sa ešte nestretli
- Pre správne nastavenie parametrov a interpretovanie výsledkov je dôležité poznať model, predpoklady, ktoré autori nástroja použili, resp. zdroj dát v príslušnej databáze
- Konkrétne nástroje a webstránky sa rýchlo menia, celkové princípy sa menia pomalšie

## Prehľad preberaných tém

- Zostavovanie genómov (najkratšie spoločné nadslovo, heuristiky, de Bruijnov graf)
- Zarovnanie (skórovanie ako pravdepodobnostný model, dynamické programovanie, heuristické zarovnávanie, E-value a P-value, lokálne vs. globálne, párové vs. viacnásobné, celogenómové)
- Evolúcia (pravdepodobnostné modely substitúcií, metóda maximálnej vierohodnosti, metóda maximálnej úspornosti, metóda spájania susedov)
- Hľadanie génov (skryté Markovove modely)
- Komparatívna genomika (hľadanie konzervovaných oblastí, komparatívne hľadanie génov, pozitívny výber, fylogenetické HMM, kodónové matice)

## Prehľad preberaných tém (pokračovanie)

- Expresia génov (zhlukovanie, klasifikácia, regulačné siete, transkripčné faktory, hľadanie motívov)
- Proteíny (predikcia štruktúry, profily a profilové HMM rodín/domén, protein threading)
- RNA štruktúra (dynamické programovanie, stochastické bezkontextové gramatiky)
- Populačná genetika (mapovanie asociácií, väzbová nerovnováha, genetický drift, štruktúra a história populácie)

## Nahliadli sme do sveta informatiky

- Algoritmus, časová zložitosť
- NP-ťažké problémy, presné algoritmy, heuristiky, aproximačné algoritmy
- Dynamické programovanie
- Stromy, grafy
- Skryté Markovove modely a bezkontextové gramatiky

## Ďalšie predmety

- **Genomika** N-mCBI-303, Nosek a kol. (LS, 2P, 3kr)
- **Seminár z bioinformatiky 1, 2** Vinař (ZS/LS, 2S, 2kr)
- **Linux pre používateľov** 1-AIN-500, Nagy (LS, 2K, 2kr)
- **Programovanie (1)** 1-MAT-130, Salanci (ZS, 2P/2C, 5kr)
- <http://compbio.fmph.uniba.sk/vyuka/>

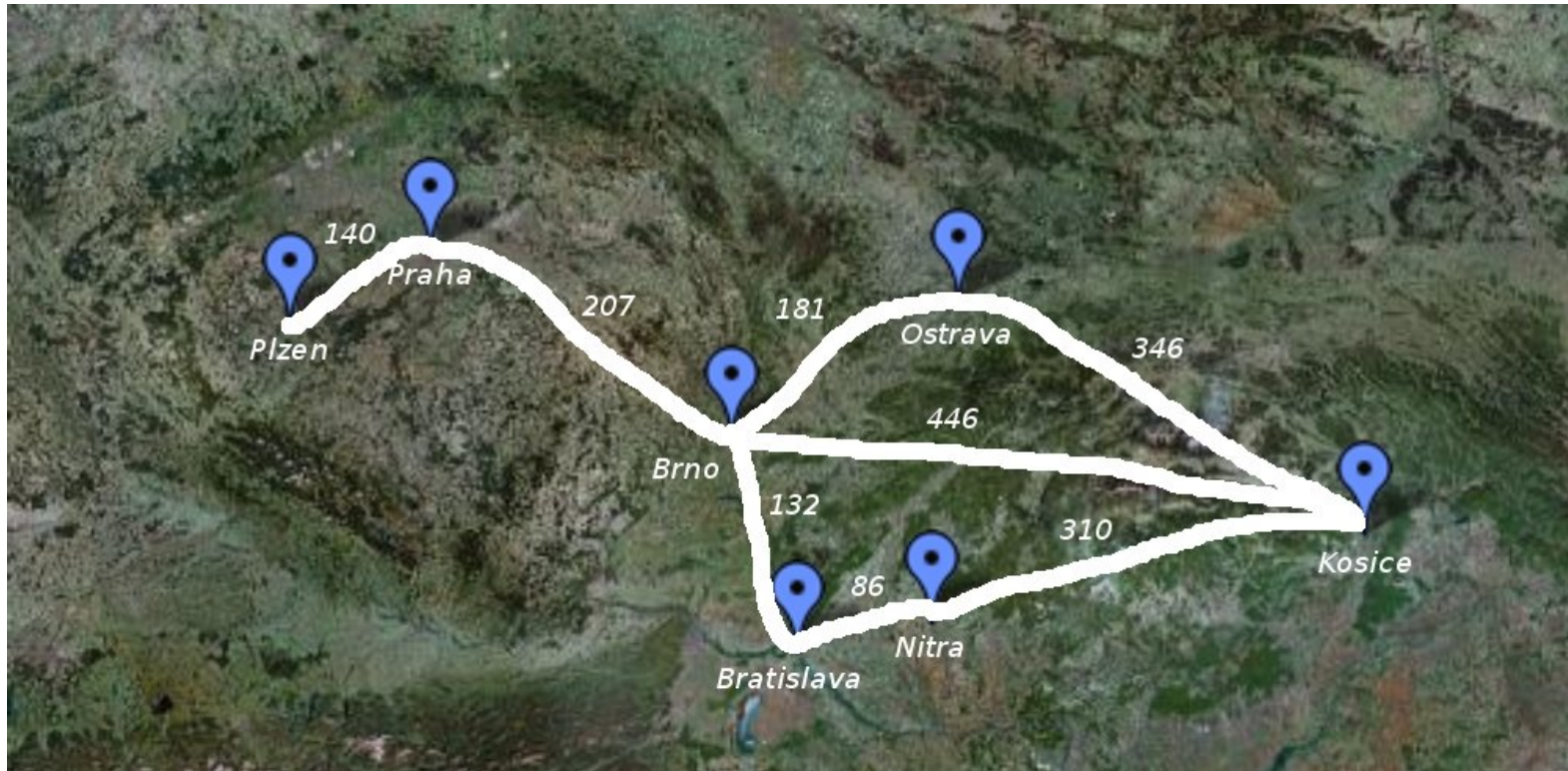
# **Teória grafov**

**Broňa Brejová**

**19.12.2020**

## Grafy a grafové algoritmy

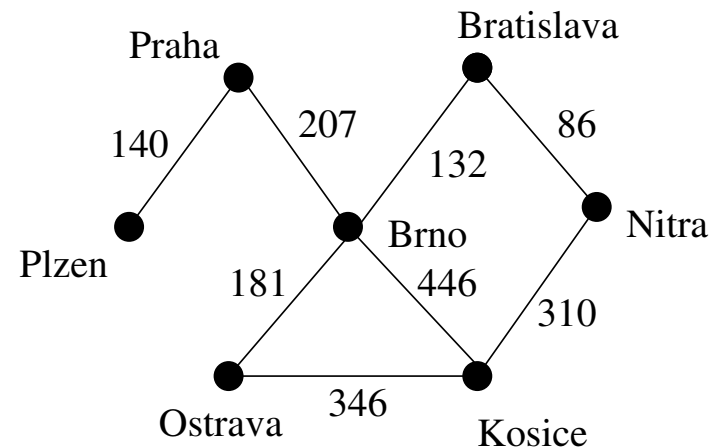
**Graf:** 7 vrcholov (mestá), 8 hrán (cestné spojenia)





Počet vrcholov  $n$ , počet hrán  $m$   
Nezáleží na rozmiestnení vrcholov

**Cesta:** Postupnosť nadväzujúcich hrán,  
žiadny vrchol sa neopakuje



Napr. Plzeň–Praha–Brno–Bratislava je cesta

Brno–Ostrava–Košice–Brno–Praha nie je cesta

**Najkratšia cesta z  $a$  do  $b$ :** Cesta spájajúca vrcholy  $a$  a  $b$  s najmenším súčtom vzdialeností na hranách

Možno spočítať v čase  $O(n^2)$  **Dijkstrovym algoritmom.**

**Cyklus:** Postupnosť nadväzujúcich hrán, ktorá sa vracia do východzieho bodu, nemá žiadne iné opakujúce sa vrcholy.



Proctor and Gamble súťaž, 1962

## Problém obchodného cestujúceho

**Vrcholy:** mestá na mape

**Hrany:** medzi každými dvoma vrcholmi, váha je vzdušná vzdialenosť

**Úloha:** obcestovať všetky mestá tak, aby celková vzdušná vzdialenosť bola minimálna (**Hamiltonovská kružnica**)

**Jednoduchá heuristika:** Vždy pokračuj v najbližšom meste, ktoré sme ešte nenavštívili.



Správny a efektívny algoritmus? Nanešťastie, obchodný cestujúci je **NP-ťažký problém**.

## Príklad: Sieť interakcií proteínov

**Vrcholy:** proteíny

**Hrany:** priame interakcie

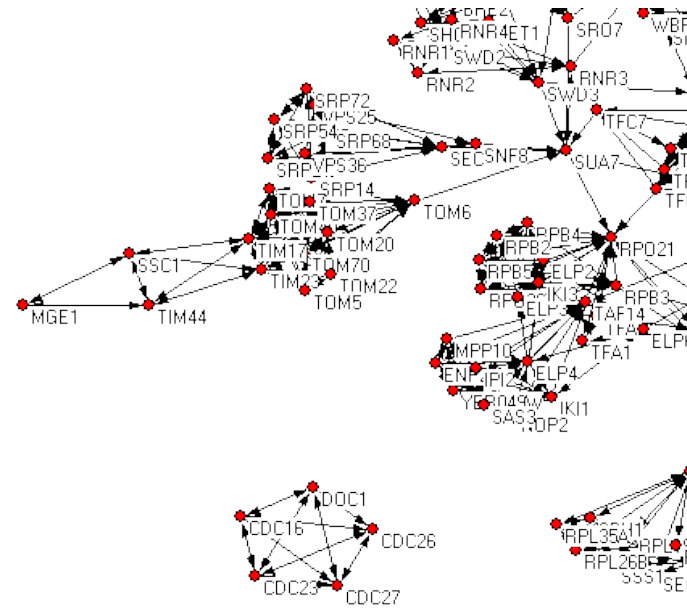
Metabolické dráhy zodp. **cestám**

Metabolické cykly zodp. **cyklom**

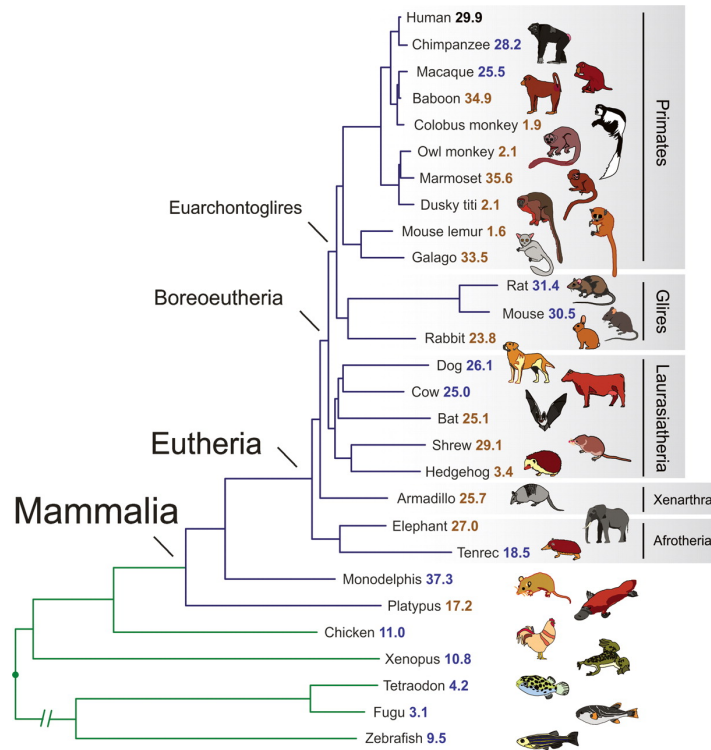
**Kliky:** Skupiny vrcholov priamo prepojené každý s každým

Komplexy zodpovedajú **klikám**

**Komponenty súvislosti:** Najväčšie skupiny vrcholov tak, aby sa v každom komponente dalo dostať z každého vrcholu do každého.



## Príklad: Fylogenetický strom



- **Stromy** sú špeciálna podtrieda grafov (acyklické, súvislé)
- Vrcholy: listy, vnútorné (spolu  $n$ )
- Hrany:  $n - 1$
- **Binárny strom:** každý vnútorný vrchol má 2 synov

**Ďalšie príklady stromov:** hierarchické zhlukovanie, dátové štruktúry na rýchle vyhľadávanie

**Ďalšie príklady grafov:** de Bruijnov graf, fylogenetická sieť (evolúcia s horizontálnym prenosom génov alebo rekombináciou), regulačné siete, hierarchia GO (gene ontology)