Askar Gafurov

FMFI UK

23/09/2021

Dnešné ciele

1. Pojem problému, algoritmu
2. Ukážka prevodu biologického problému na informatický
3. Efektivita algoritmu, pojem časovej zložitosti, O-notácia
4. NP-ťažké algoritmy

- *Formulácia problému*: jasne definované vstupné a výstupné dáta a aký výstup očakávame pre každý vstup.
- Formulácia neuvádza *akým spôsobom* sa majú zo vstupov vypočítať výstupy.
- *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 $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Príklad

```
K=19
3 4 6 3 6 4 9 2 8
    ^^^^^^^
```

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Príklad

```
K=19
3 4 6 3 6 4 9 2 8
    ^^^^^^^^^
```

Ako túto úlohu vyriešiť?

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, 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* - počet logických operácií, ktoré program vykoná, v závislosti od množstva dát.
- Pre každú veľkosť vstupu *odhadneme najhorší možný prípad*

```
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
```

Výpočet časovej zložitosti

```
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
```

Počet operácií $:=$ a $+$

$$T(n) = \sum_{i=1}^{n}\left(\sum_{j=i}^{n}\left(1 + \sum_{u=i}^{j} 2\right)\right) = \ldots = \frac{1}{6}n^3 - n^2 + \frac{5}{6}n$$

Zaujíma nás *najvýznamnejší* člen tejto sumy, a to je $\frac{1}{6}n^3$. Navyše, nezaujíma nás konštanta pri tom člene. Výsledok takéhoto "zjednodušenia" píšeme ako $O(n^3)$ a hovoríme, že daný algoritmus má *kubickú časovú zložitosť*.

Prečo používame O-notáciu?

Úlohou O-notácie je odpovedať na otázky typu "ak budem mať X krát viac dát, koľkokrát dlhšie budem čakať na výsledok?"

Napríklad, $T(10^5) = \dfrac{1}{6} \cdot 10^{15} - 10^{10} + \dfrac{5}{6} \cdot 10^5 = 166656666750000$

$T(2 \cdot 10^5) = \dfrac{1}{6} \cdot 2^3 \cdot 10^{15} - 2^2 \cdot 10^{10} + 2 \cdot \dfrac{5}{6} \cdot 10^5 = 1333293333500000$

$$\frac{1333293333500000}{166656666750000} = 8.000240011400564$$

Tento výpočet môžem spraviť len s najvýznamnejšími členmi: $\dfrac{\frac{1}{6} \cdot 2^3 \cdot 10^{15}}{\frac{1}{6} \cdot 10^{15}} = 2^3 = 8$. Všimnite si, že na konštante $\frac{1}{6}$ nezáleží.

Prečo používame O-notáciu?

Teda, namiesto porovnávania presných funkcií stačí porovnať ich pomocou "zjednodušených" funkcií:

$$\frac{T(X \cdot n)}{T(n)} \approx \frac{(X \cdot n)^3}{n^3} = \frac{X^3 \cdot n^3}{n^3} = X^3$$

Ak by časová zložitosť bola napríklad $O(2^n)$, tak by zmena času behu algoritmu vyzerala nasledovne:

$$\frac{2^{X \cdot n}}{2^n} = 2^{(X-1) \cdot n}$$

Všimnite si, že pri exponenciálnej zložitosti nárast závisí nielen od X, ale aj od pôvodnej veľkosti vstupu n.

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

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$?

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \dots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \dots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \dots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \dots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \dots + a[j]$? $a[i] + \dots + a[j] = S[j] - S[i-1]$

Efektívnejší algoritmus

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$? $a[i] + \ldots + a[j] = S[j] - S[i-1]$
- Ako vieme spočítať hodnoty $S[i]$?

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$? $a[i] + \ldots + a[j] = S[j] - S[i-1]$
- Ako vieme spočítať hodnoty $S[i]$?

```
S[0] := 0
pre každé i od 1 po n:
|    S[i] := S[i-1] + a[i]
```

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \dots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \dots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \dots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \dots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \dots + a[j]$? $a[i] + \dots + a[j] = S[j] - S[i-1]$
- Ako vieme spočítať hodnoty $S[i]$?

```
S[0] := 0
pre každé i od 1 po n:
|    S[i] := S[i-1] + a[i]
```

- Akú má časovú zložitosť výpočet $S[i]$?

Efektívnejší algoritmus

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$? $a[i] + \ldots + a[j] = S[j] - S[i-1]$
- Ako vieme spočítať hodnoty $S[i]$?

```
S[0] := 0
pre každé i od 1 po n:
|    S[i] := S[i-1] + a[i]
```

- Akú má časovú zložitosť výpočet $S[i]$? $O(n)$

Problém # 2: Najkratšie spoločné nadslovo

Formulácia problému

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

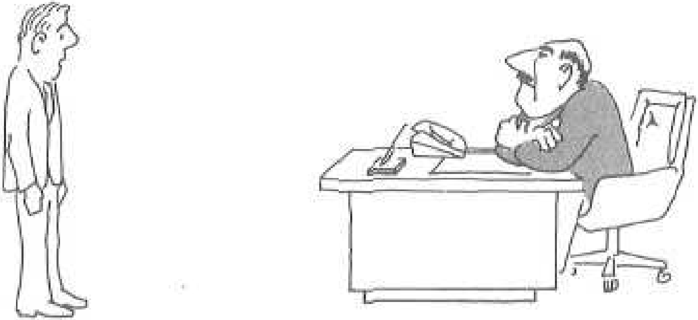
Príklad

Vstup: GCCAAC, CCTGCC, ACCTTC
Výstup: CCTGCCAACCTTC (najkratšie možné)

Problém # 2: Najkratšie spoločné nadslovo

Najlepší algoritmus?

- Nepoznáme algoritmus, ktorý by bežal v polynomiálnom čase t.j. $O(n^k)$ pre nejakú konštantu k.
- Daný problém je *NP-ťažký*.



"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-ťažkými 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-ťažkými problémami?

Exaktný výpočet pomocou iného problému

- Preformulovať do podoby jedného z dobre známych NP-ťažkých 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 problémy).
- 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 $m_1, m_2, \ldots, m_k$ 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 $m_1, m_2, \ldots, m_k$ a X

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

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

Problém platenia minimálnym počtom mincí

Vstup: hodnoty k mincí $m_1, m_2, \ldots, m_k$ 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 $m_1, m_2, \ldots m_k$

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

```
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

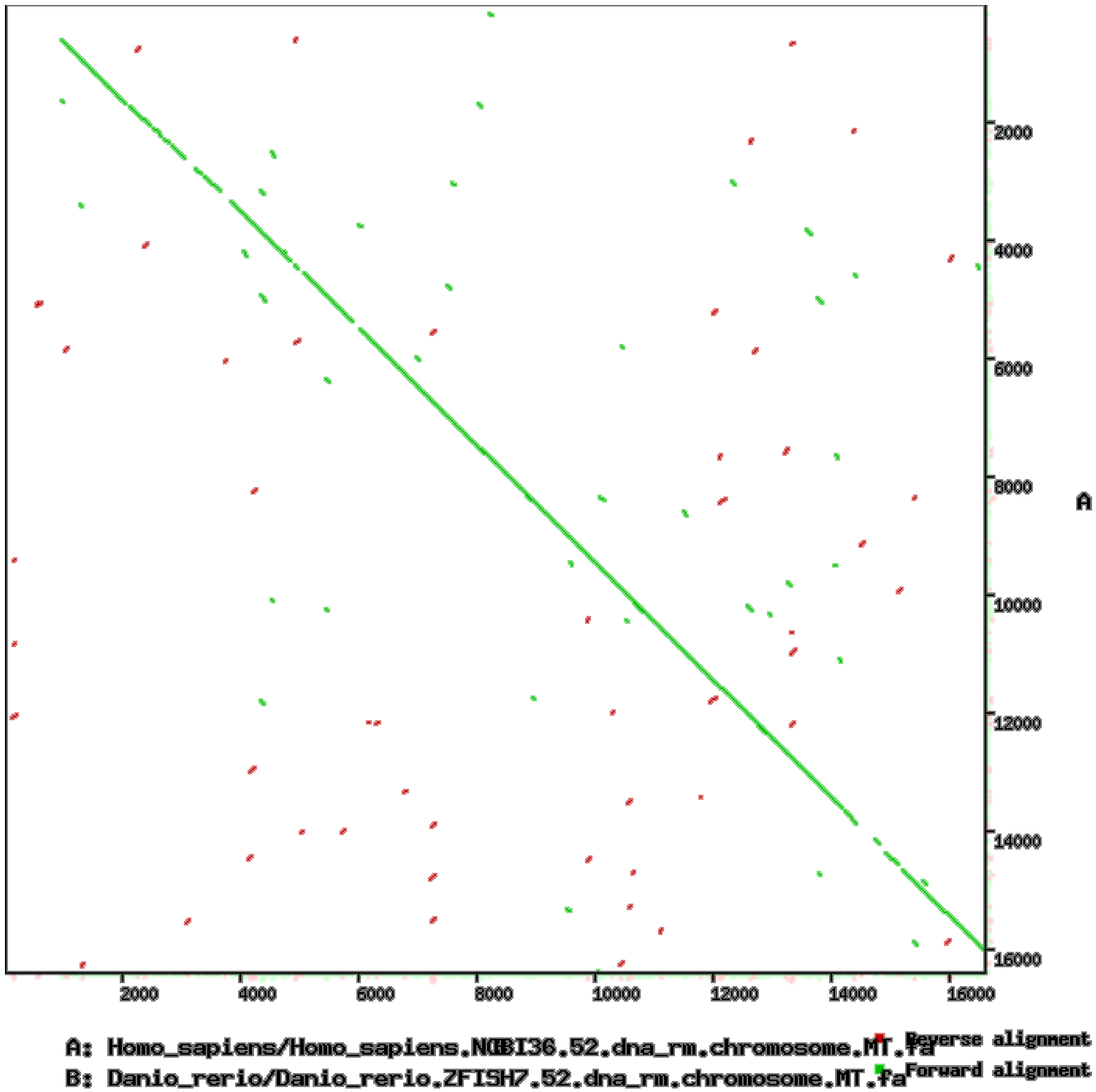
Zarovnávanie sekvencií

(cvičenie)

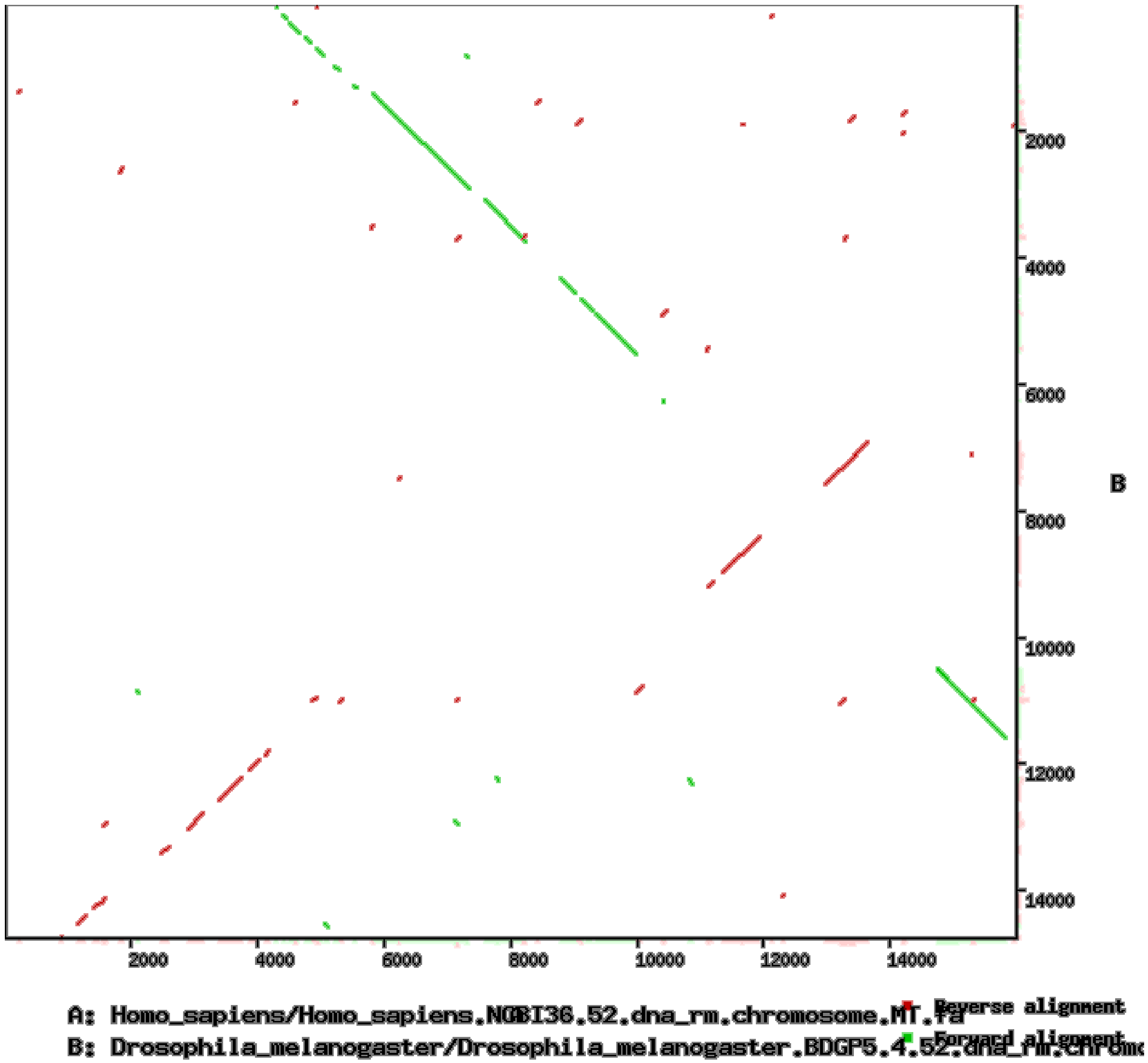
Broňa Brejová

8.10.2021

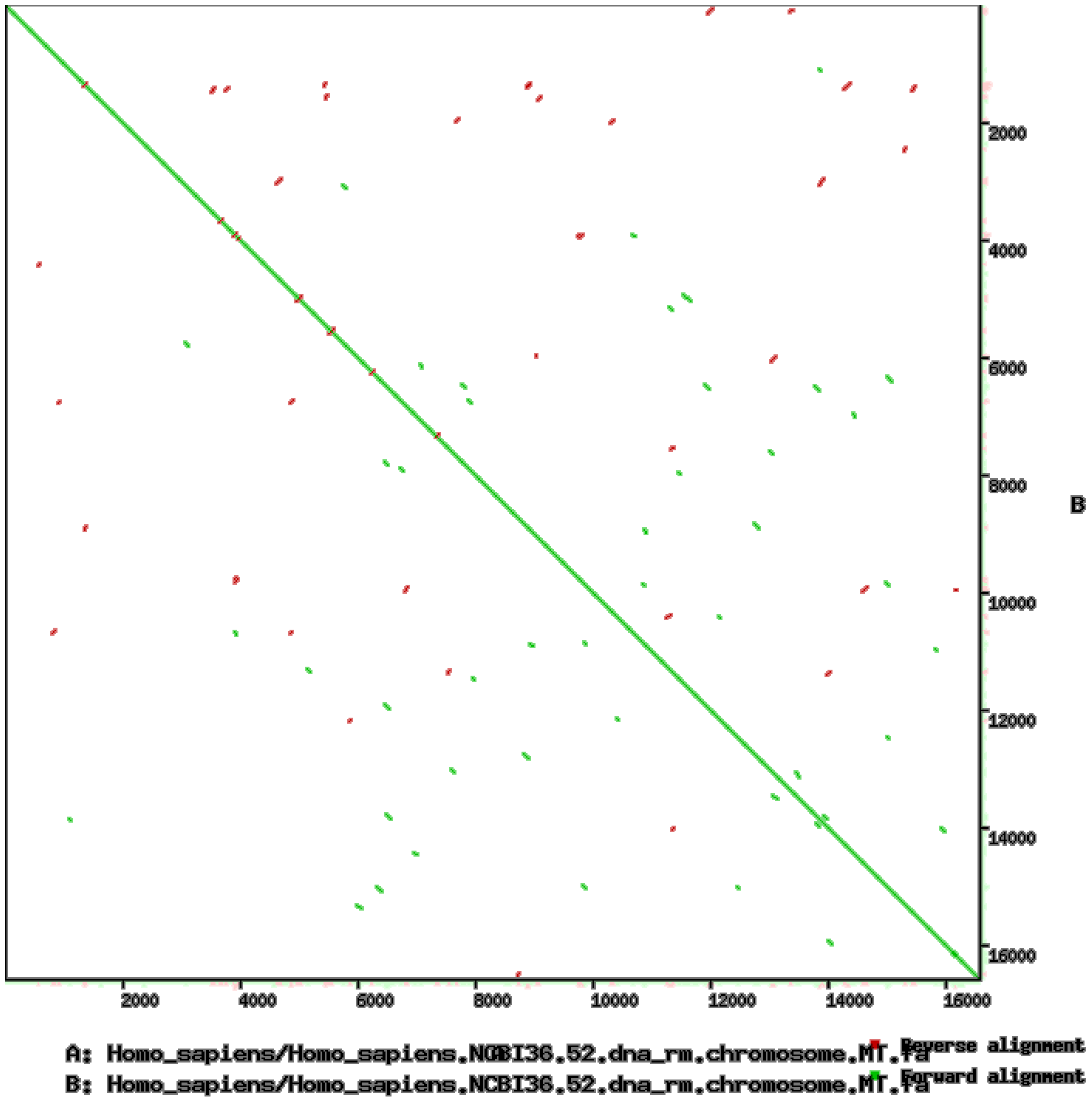
Mitochondriálny genóm človeka vs. ryba Danio rerio

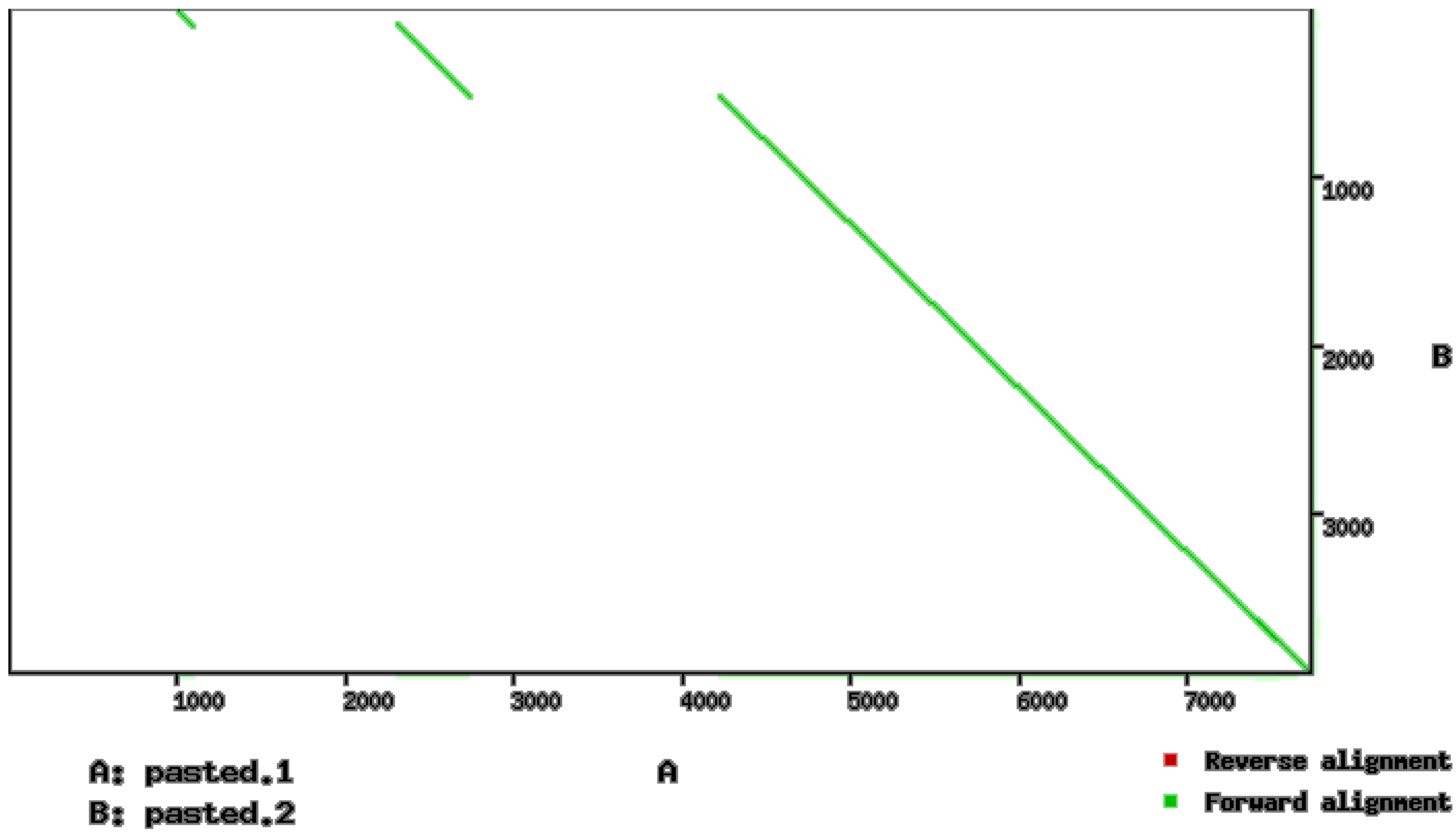


Mitochondriálny genóm človeka vs. Drosophila melanogaster



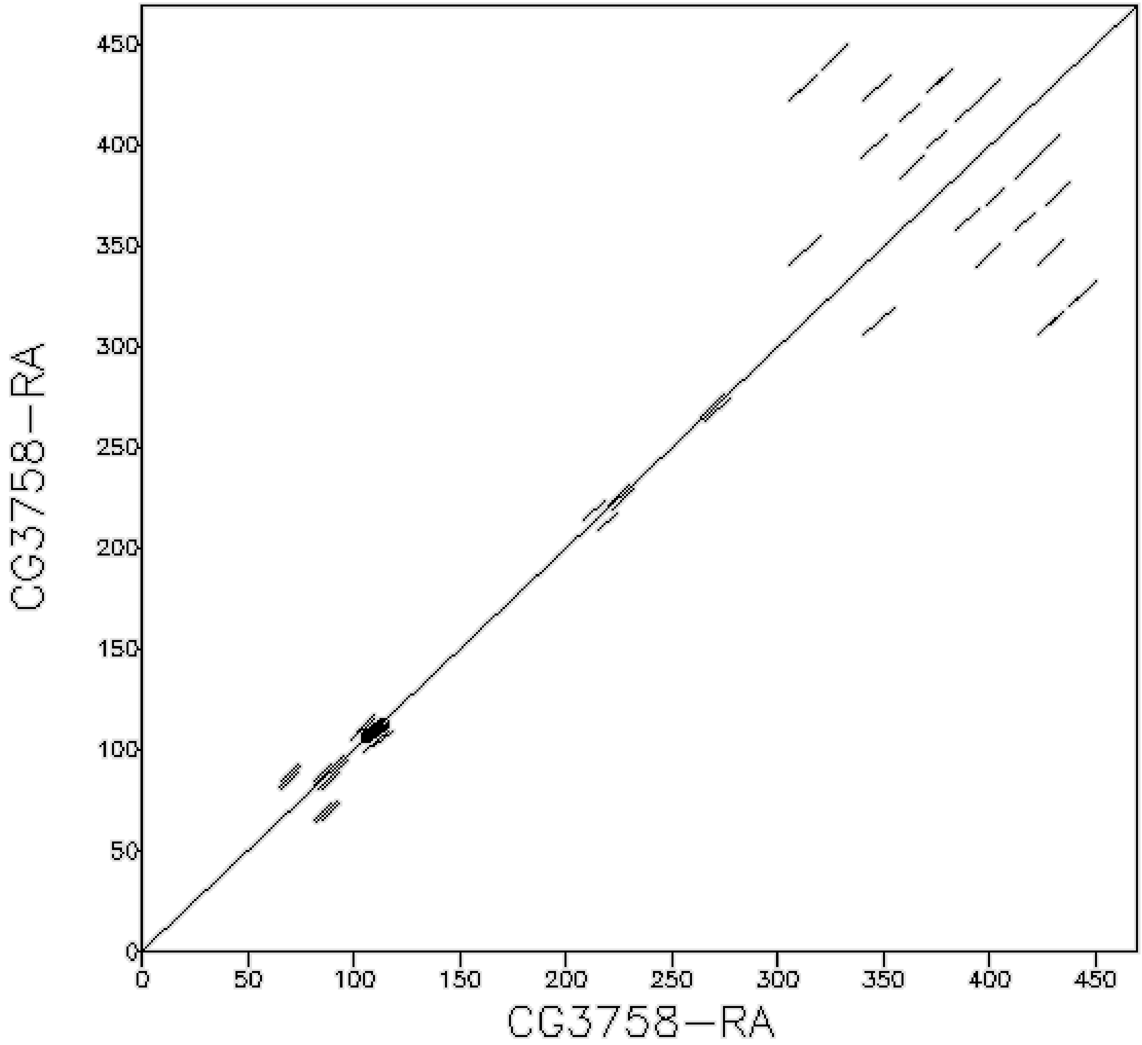
Mitochondriálny genóm človeka vs. to isté





1000
2000
3000
4000
5000
6000
7000
A
1000
2000
3000
B
A: pasted.1
B: pasted.2
Reverse alignment
Forward alignment

Drosophila proteín Escargot zinc finger vs. to isté

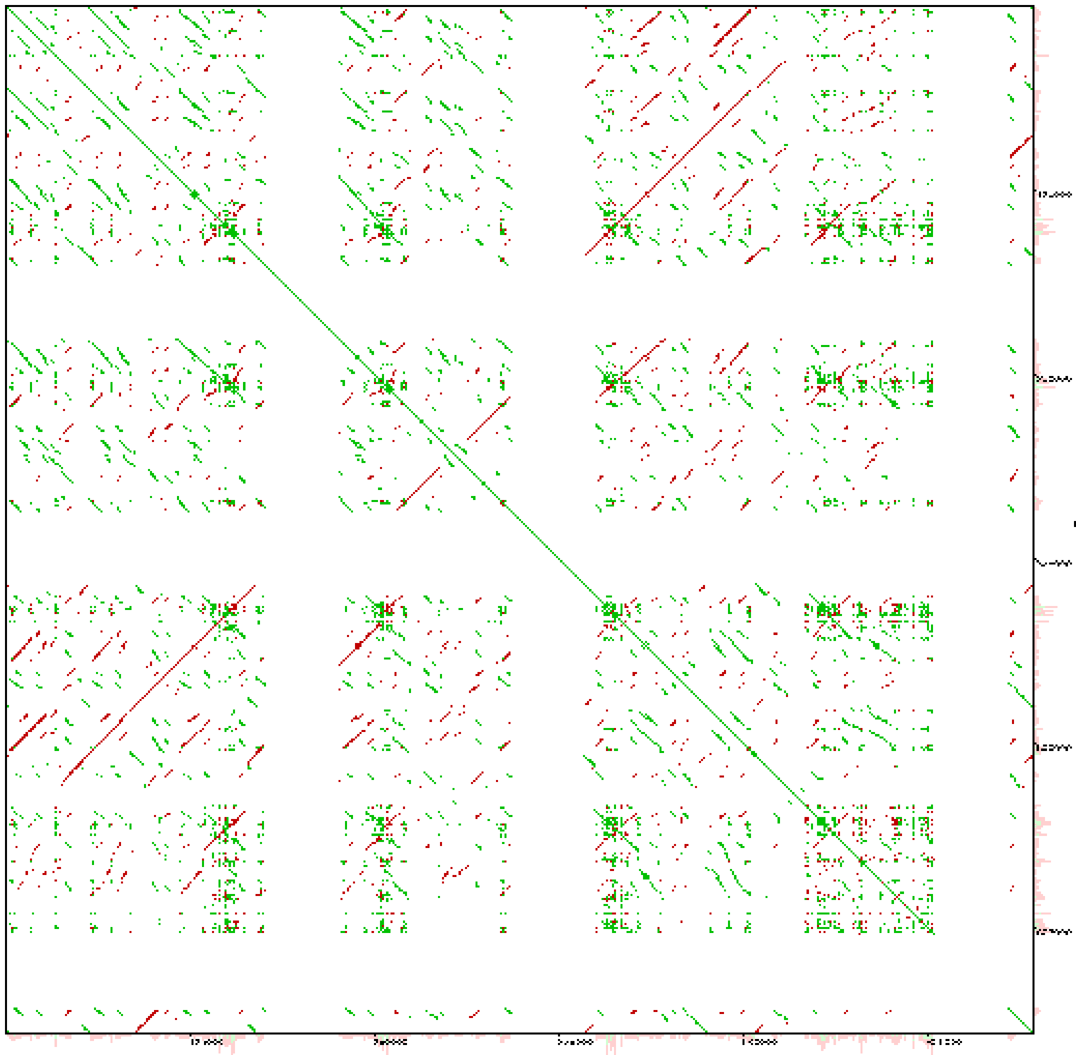


Drosophila proteín Escargot zinc finger

Description:	Protein escargot
Source organism:	Drosophila melanogaster (Fruit fly) View Pfam proteome data.
Length:	470 amino acids

Pfam domains

Source	Domain	Start	End
Pfam B	Pfam-B_18487	8	270
low_complexity		71	95
low_complexity		101	129
low_complexity		163	177
low_complexity		244	263
low_complexity		264	274
Pfam A	zf-C2H2	309	332
Pfam A	zf-C2H2	344	366
Pfam A	zf-C2H2	370	392
Pfam A	zf-C2H2	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 $n = 300$)

accacttgcgcacgatttccagattcggtttccctgggcgcacgaagggc
ccacgaagcgGCTCAACccggagccttagttagaagggggtctccgtca
agagagacggtaagttggagggtcactagcggtggactccgaatggaaac
actgaatagtggcagaacctaaacctcgttttggatttcctgaaaaggc
aggcgctagaggaagaggcacgactgtgctagagataatcacttgtaaga
ccttggggatgggcttcgtatgcagaacgcgataaggtatcgaaaacgtg

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
gcgataaccatgagatgacaggtccgatgctaatgttaacggacctacag
tgacatgttaaagtgtccattaagtttataccggaatcaacgagtgtccc
ccagcgcggcgaccgatggagccCCTGCAGgtatactcacttcaaggatt
accgctcggtgtaagttagtgttcagtcagactatactaagtattcagtt
atagagcgttagtaggtcgaccatgagcgggtaggGTGCCGAGatgtgaa
```

Počet výskytov: 2

Náhodný dotaz a dazabáza

Dotaz: `TCGACCGAAA`

Databáza:

```
tactccattagggattataacgactaaagcccgtcgtggcgggatcactt
tgagattcaactttaacgcatcacagaggaatctgagacaaagcaaaacc
gatcataatgatcgatccaggtaataagtctccttgatggcgttagactg
gaaataacagttgacttccgactatagtttaatgaacgttcgtaattaga
cgatcgtgtaacttaaccaaaggctgcccccaaactagctgagtaatagc
tcgtcctgagcatgtaagagtcagcctccacggaacactgcaacgttctt
```

Počet výskytov: 0

Náhodný dotaz a dazabáza

Dotaz: CCCGTCGTAG

Databáza:

cagcattagccccgttatttCGTCGTtctccaacgggtctgcctttctgg
aacgtggcgaaccttcacaggtcagtctgtcatcgcctgcgcttagagcg
gacggtactcgaaaggtcggttcagtgtggcgctggaaagaagaatagca
acacatgcactaatggaaggtcccagtggtgtgggacattctggaCCCGT
GTgtgccaacctatgtgagctccggcgttgactcggaggatgttaacaag
atcaagctgtaggcgacgatccccgccgggtttcctctactgcctcgagc

Počet výskytov: 2

Náhodný dotaz a dazabáza

Dotaz: AGGATGAGGA

Databáza:

ttatcgattctccggtgcgccagtacagcacaaggctcggatcctgtaaa
acactacaccttaaaaactaagtcAGGATGtgatctcccttaaGATGAGa
cagtctctaatgcggcgtagtgggaccctcgtgaccgagctaagcagttc
acaatgggcgctctgagcgattggctggagaccttgacttcccggtaggt
gtggtgttagttctgtgcccagagataaccatccaccgtaatggatctcg
taactttacGATGAAGAccggcatcatctcagttatatttctaggacggg

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

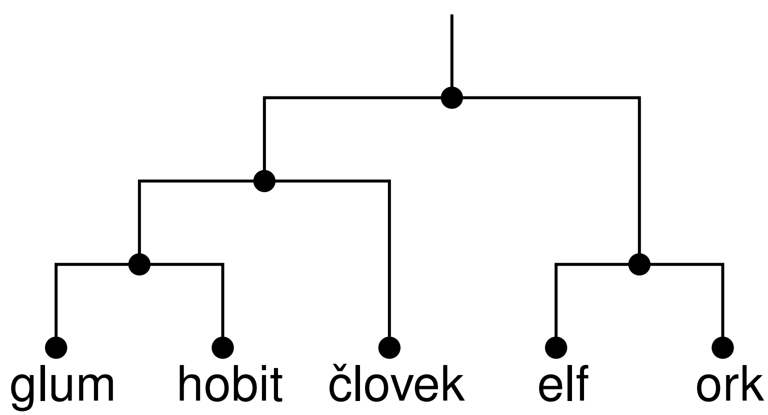
Fylogenetické stromy

Broňa Brejová

26.10.2023

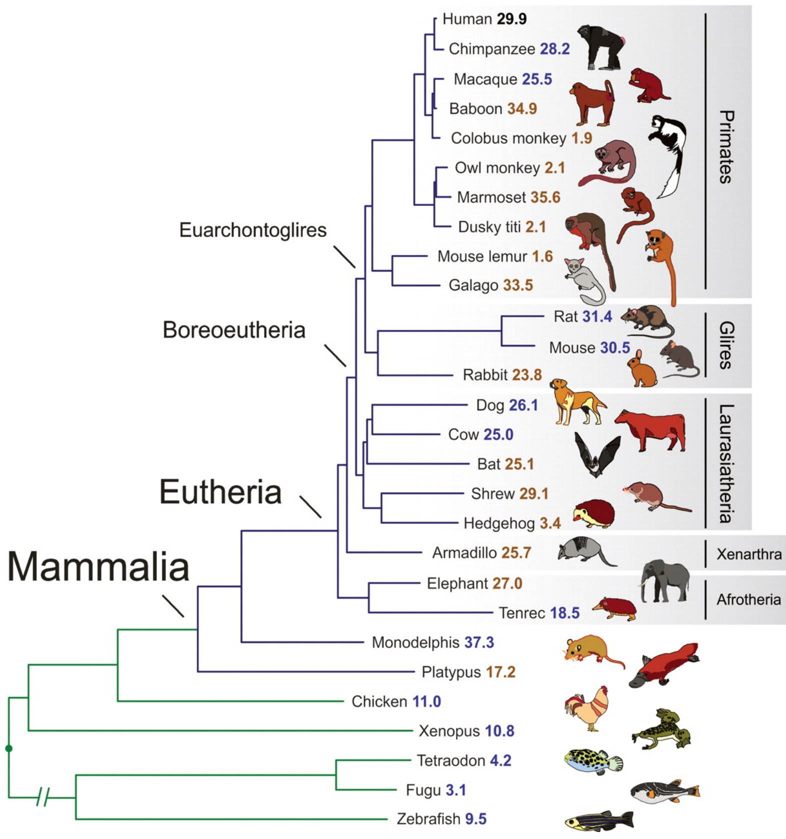
Terminológia

- 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
- podstrom, subtree, clade



Reálne ukážky stromov z článkov (zakorenený/nezakorenený)

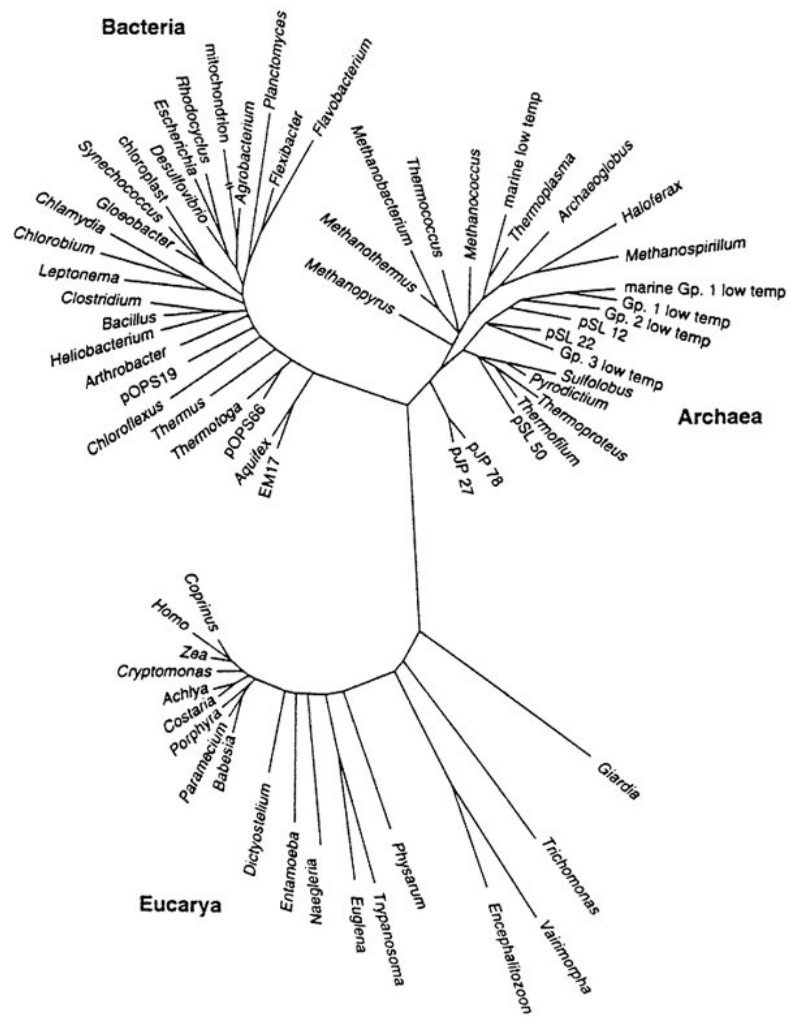
[Margulies et al. 2007]



zakorenený pomocou

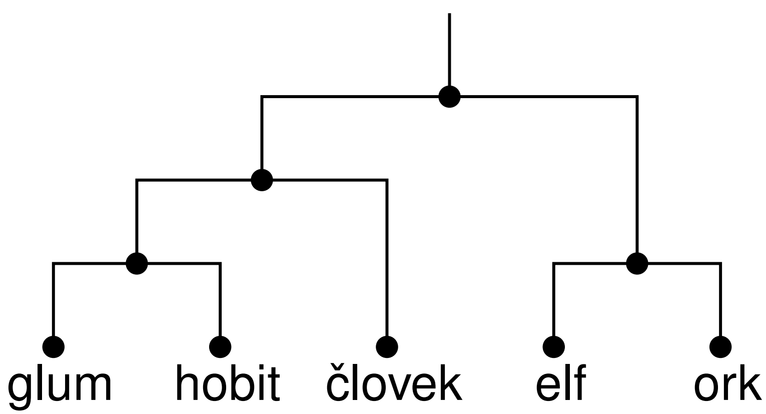
vonkajšej skupiny (outgroup)

[Pace et al 1997]



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?

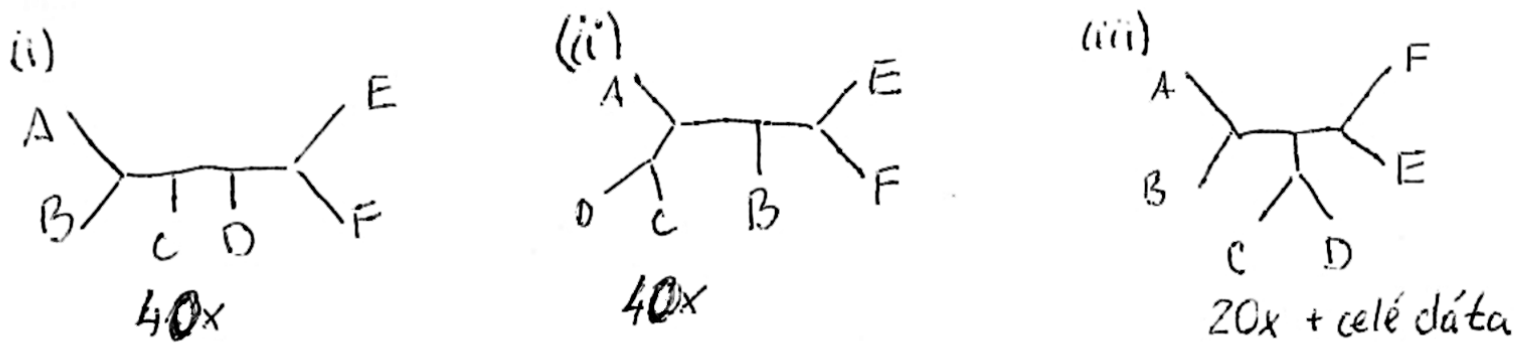


Bootstrap

- Náhodne vyberieme niektoré stĺpce zarovnania, 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%?

Opakovanie pravdepodobnostných modelov

Keď počítame pravdepodobnosť, rozmýšľame o **myšlienkovom experimente**, v ktorom hádžeme kockou, ťaháme gulôčky z vreca a pod.

- Dôležité je vždy si poriadne uvedomiť, ako tento experiment prebieha
- Experimenty nastavujeme tak, aby odzrkadľovali nejaké aspekty reality, napr. skutočných DNA sekvencií, ich evolúcie a pod.
- Pravdepodobnosti, ktoré spočítame v idealizovanom svete nám možno niečo povedia o reálnom svete
- Slávny citát štatistika Georga Boxa:
  All models are wrong, but some are useful.

Aké sme doteraz videli modely

- **Skórovacie matice:** porovnávame model náhodných sekvencií a model náhodných zarovnaní
- **E-value v BLASTe:** náhodne vygenerujeme databázu a dotaz (query), koľko bude v priemere medzi nimi lokálnych zarovnaní so skóre aspoň S?
- **Hľadanie génov:** model generujúci sekvenciu+anotáciu naraz (parametre nastavené na známych génoch).
  Pre danú sekvenciu, ktorá anotácia je najpravdepodobnejšia?
- **Evolúcia, Jukes-Cantorov model:** model generujúci stĺpec zarovnania.
  Neznáme parametre: strom, dĺžky hrán.
  Pre danú sadu stĺpcov zarovnania, ktoré parametre povedú k najväčšej pravdepodobnosti? $\max_{param} \Pr(data|param)$

Evolúcia, Jukes-Cantorov model

Model generujúci stĺpec zarovnania.

Neznáme parametre: strom, dĺžky hrán.

Pre danú sadu stĺpcov zarovnania, ktoré parametre povedú k najväčšej pravdepodobnosti? $\max_{param} \Pr(data|param)$

- Pravdepodobnosť zmeny/nezmeny na hrane dĺžky t:
  $Pr(A|A,t) = (1 + 3e^{-\frac{4}{3}t})/4,$
  $P(C|A,t) = (1 - e^{-\frac{4}{3}t})/4$
- Ak poznáme ancestrálne sekvencie, vieme spočítať pravdepodobnosť dát
- Ancestrálne sekvencie sú náhodné premenné, ktoré nás nezaujímajú: marginalizujeme ich (uvažujeme všetky ich možné hodnoty)

K-means clustering

Broňa Brejová

12.11.2020

Formulácia problému

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

Výstup: Rozdelenie vektorov do k zhlukov:

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

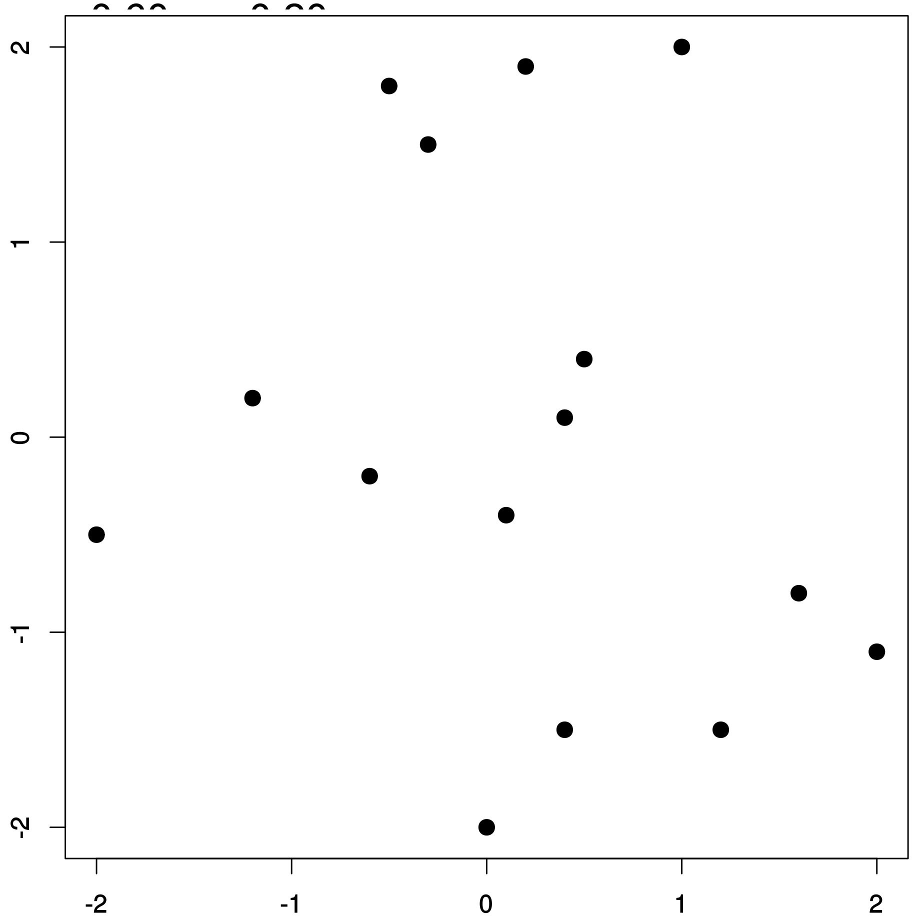
Hodnoty $c_1, \ldots, c_t$ a $\mu_1, \ldots, \mu_k$ volíme tak, aby sme minimalizovali súčet štvorcov vzdialeností od každého vektoru k centru jeho zhluku:

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

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

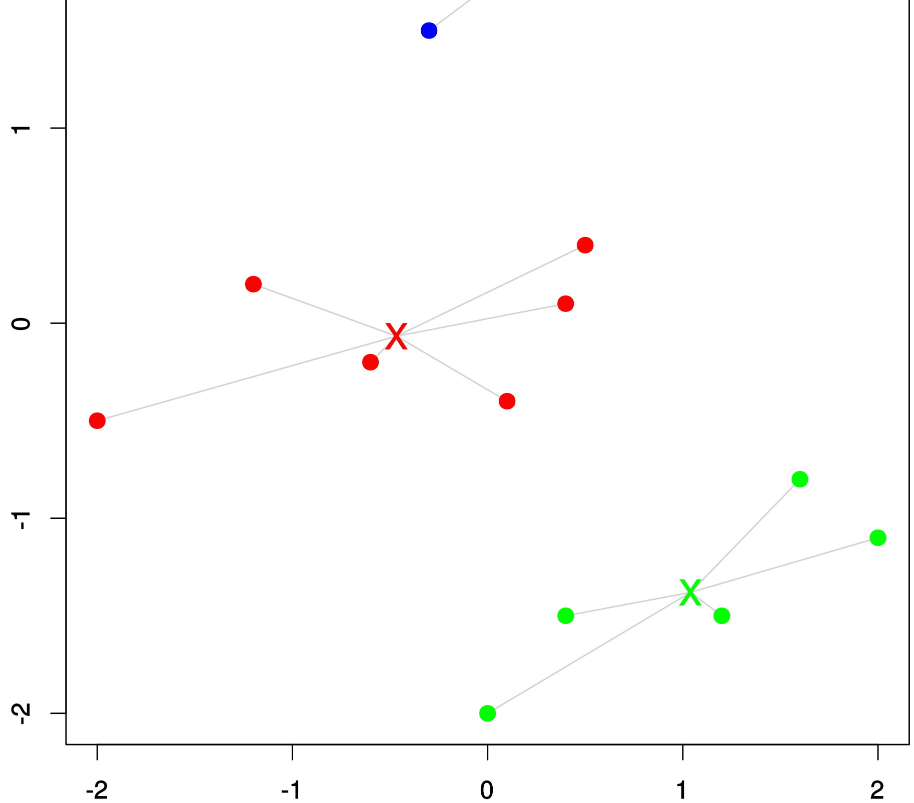
Príklad vstupu

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
μ_1	**-0.47**	**-0.07**	
μ_2	**1.04**	**-1.38**	
μ_3	**0.10**	**1.80**	



Sum of squared distances: 10.61

Algoritmus

Heuristika, ktorá nenájde vždy najlepšie zhlukovanie.

Začne z nejakého zhlukovania a postupne ho zlepšuje.

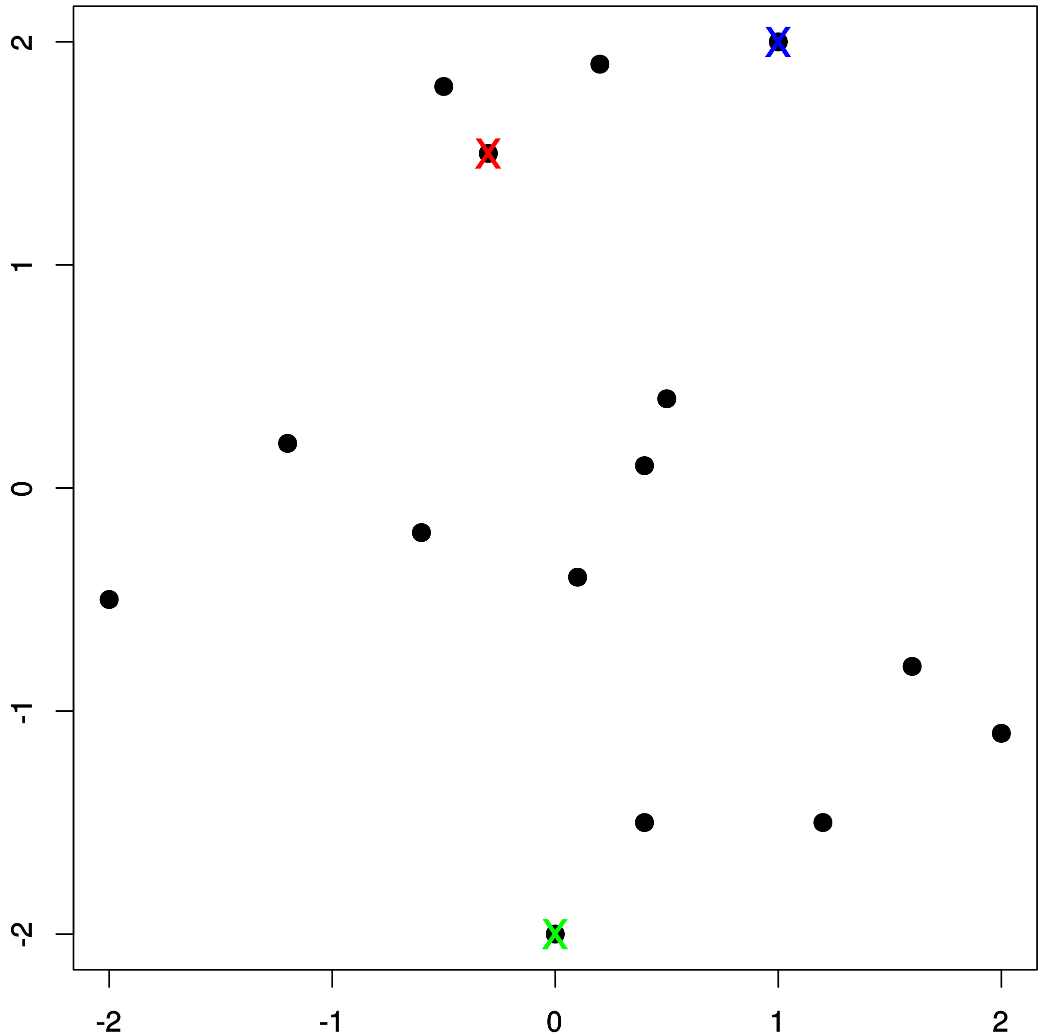
Inicializácia:

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

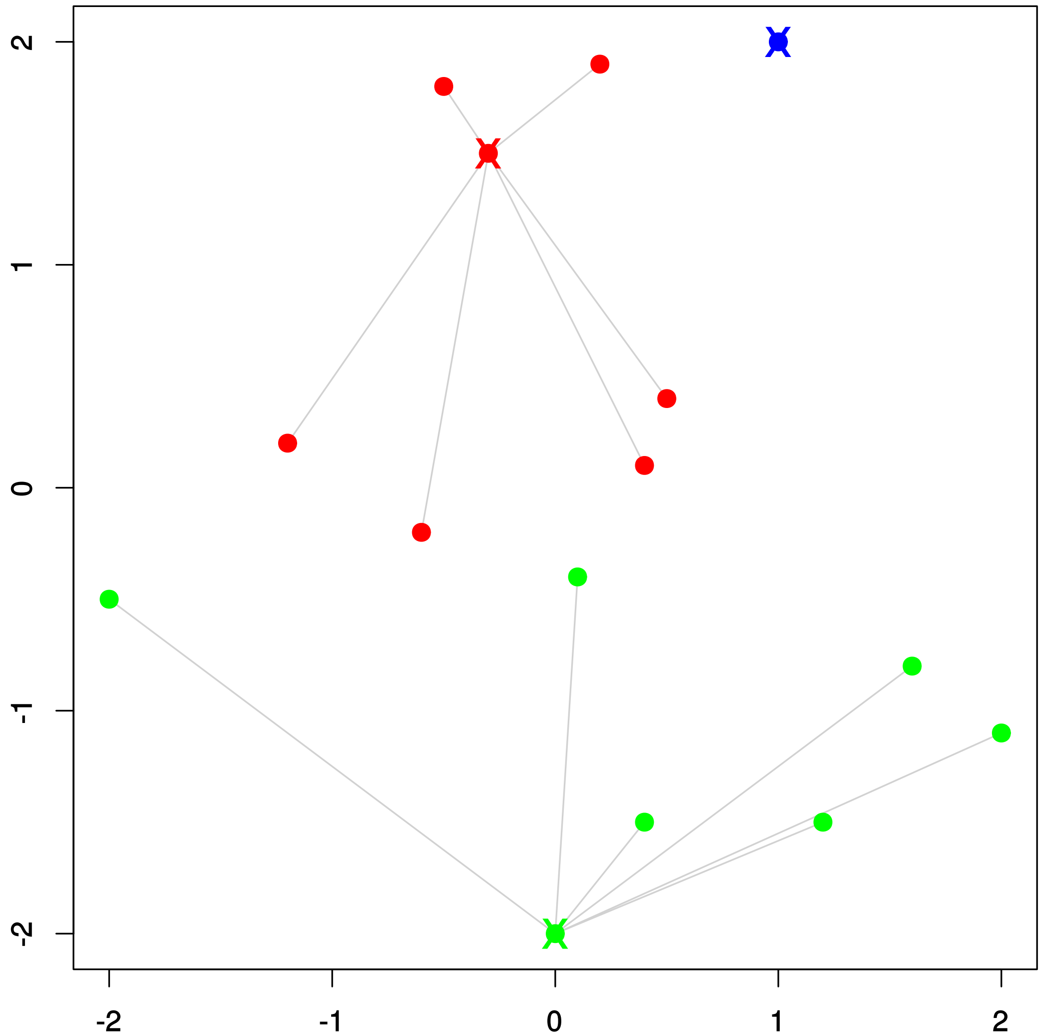
Opakuj, kým sa niečo mení:

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

Zvolíme náhodné centrá μ_i

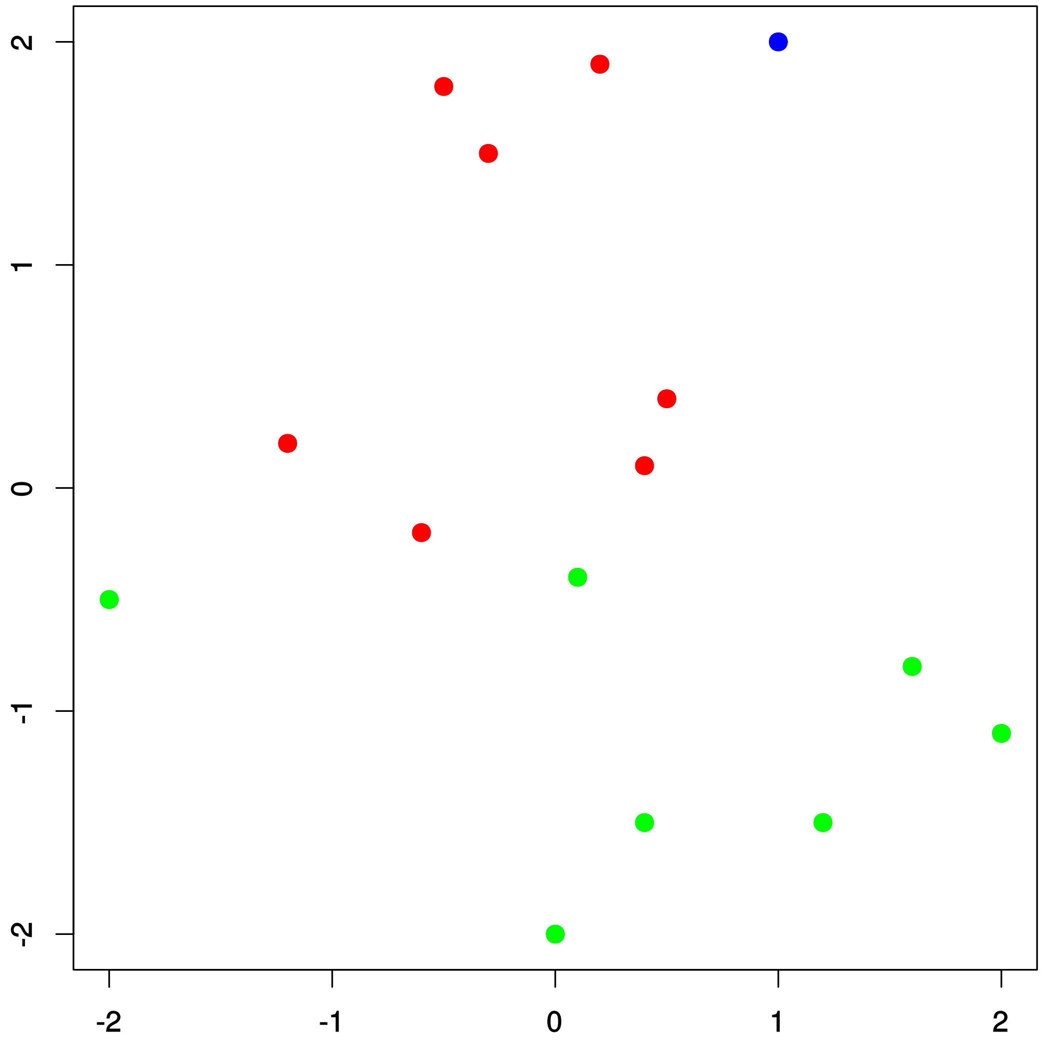


Vektory priradíme do zhlukov (hodnoty c_i)

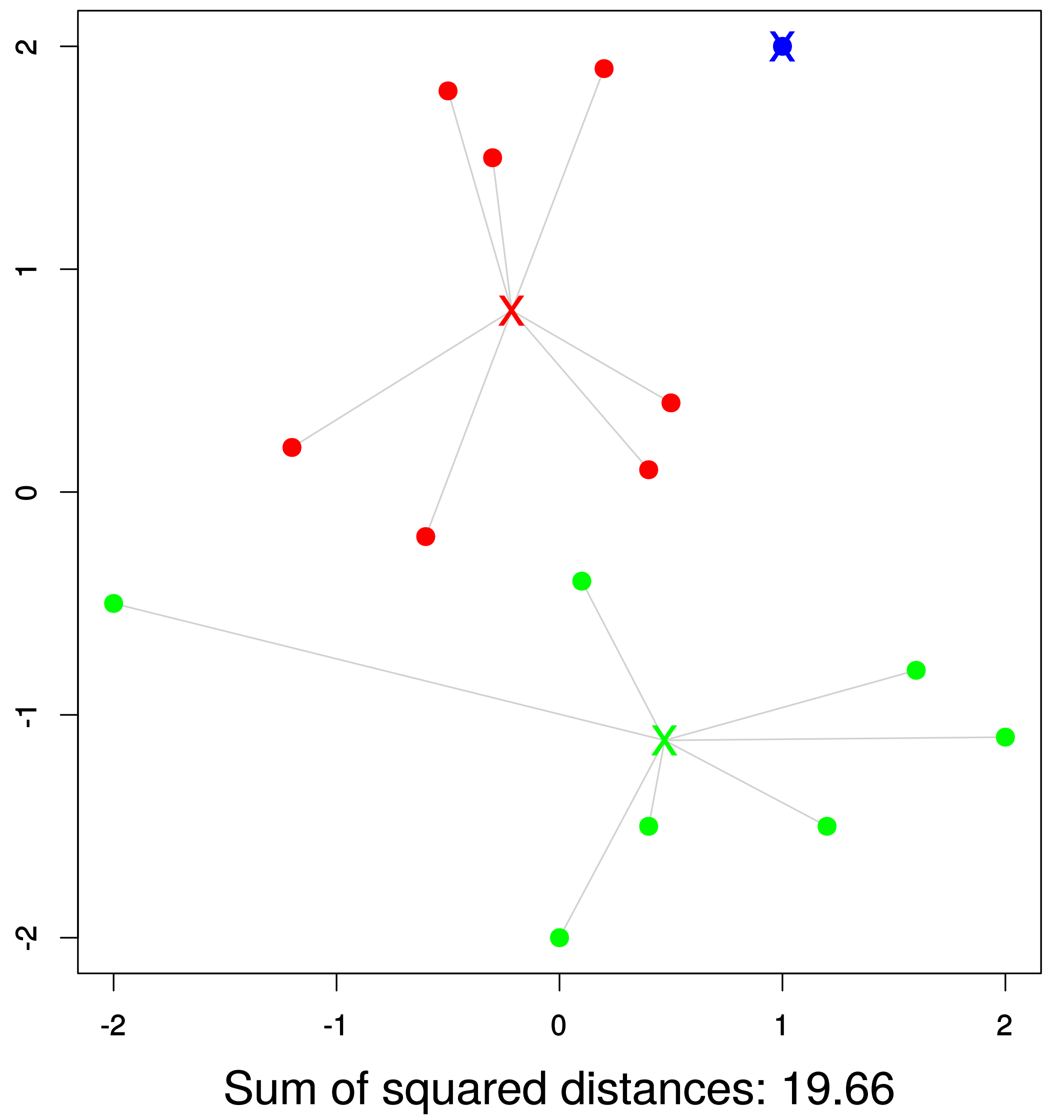


Sum of squared distances: 30.05

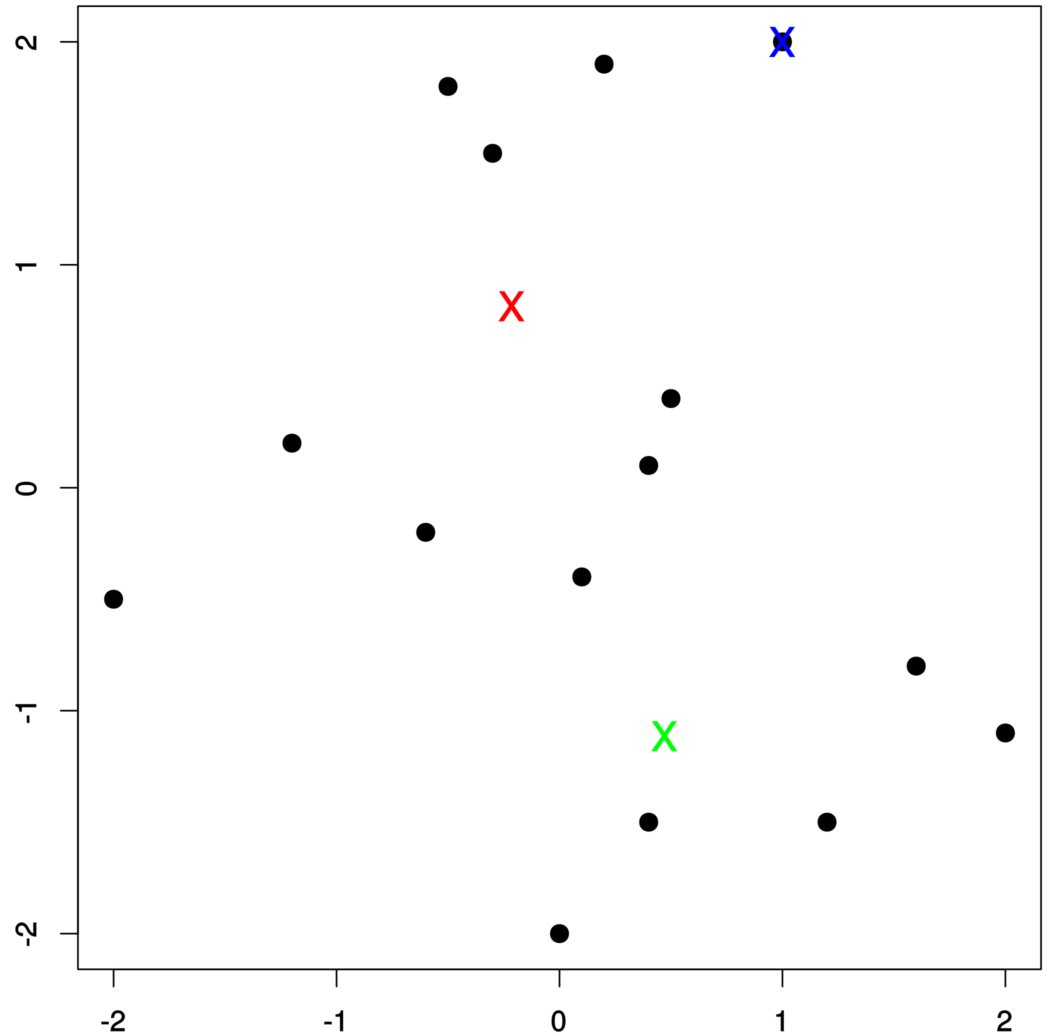
Zabudneme μ_i



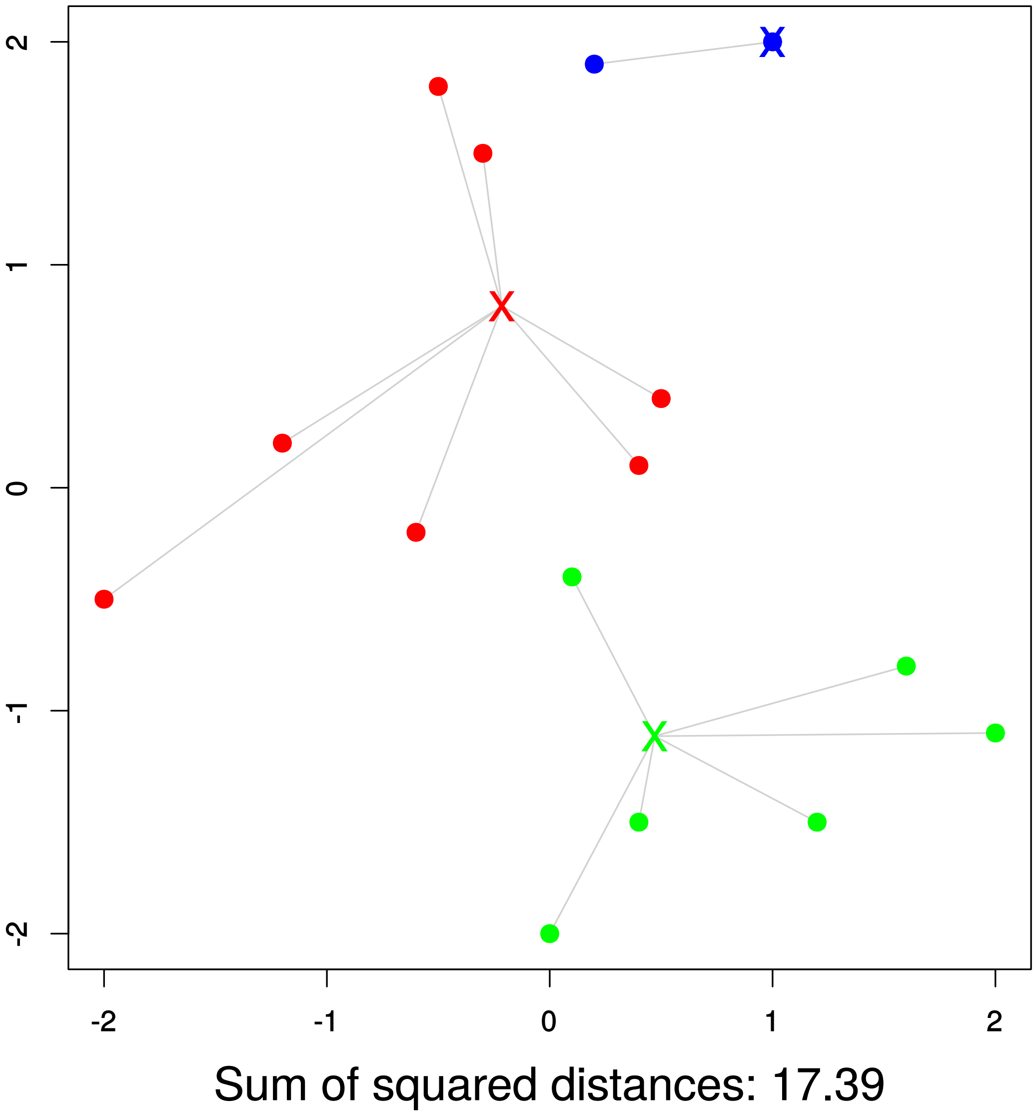
Dopočítame nové μ_i (suma klesla z 30.05 na 19.66)



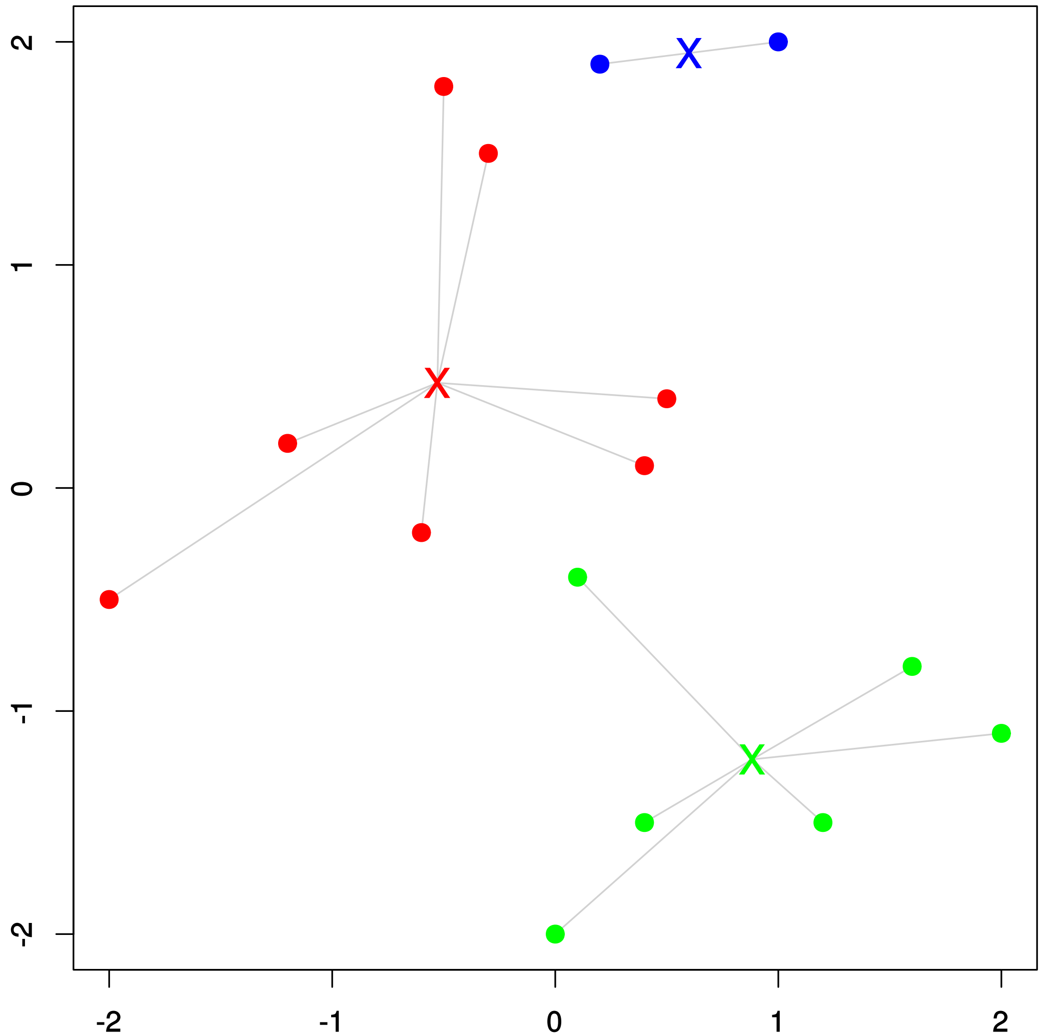
Zabudneme c_i



Dopočítame nové c_i (suma klesla z 19.66 na 17.39)

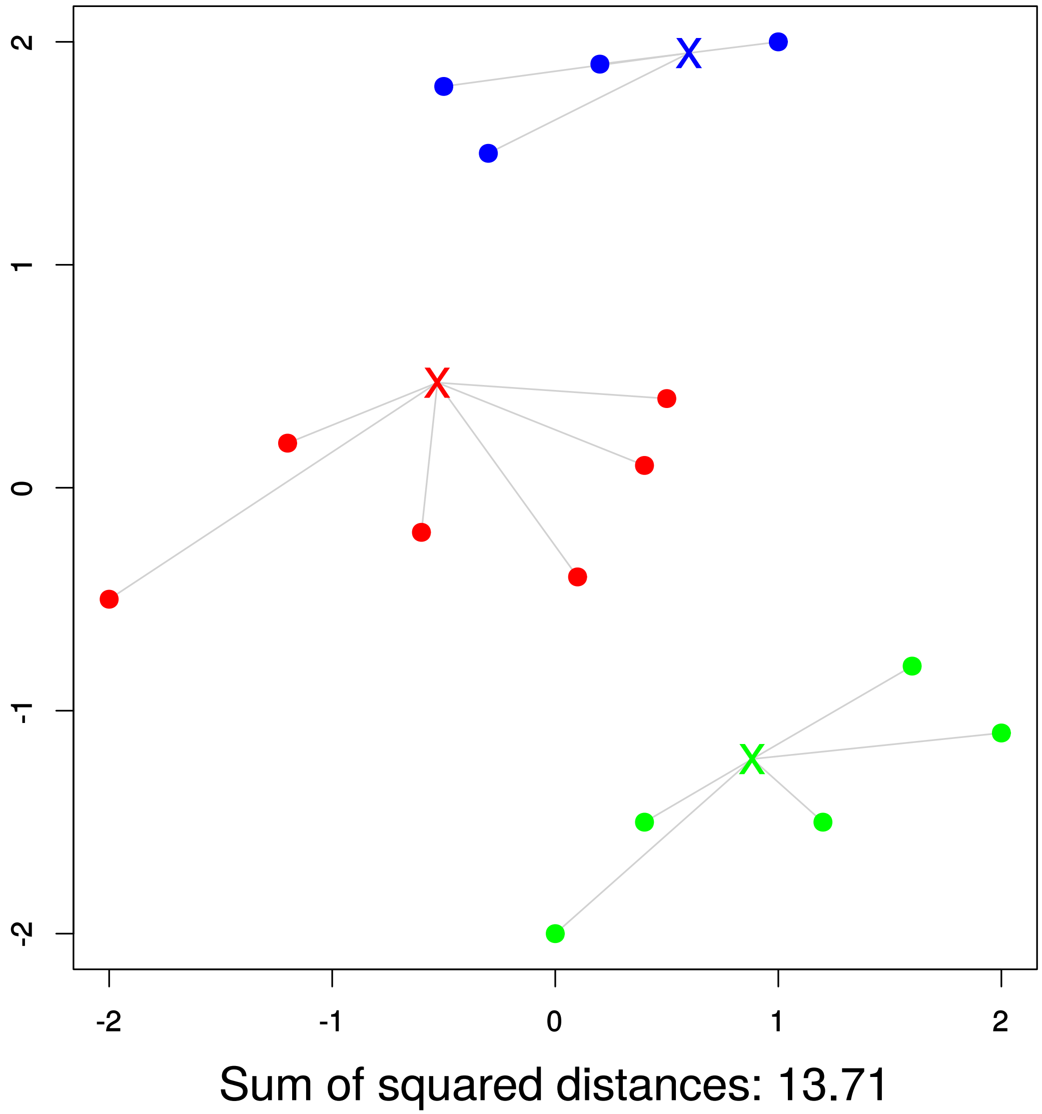


Prepočítame μ_i

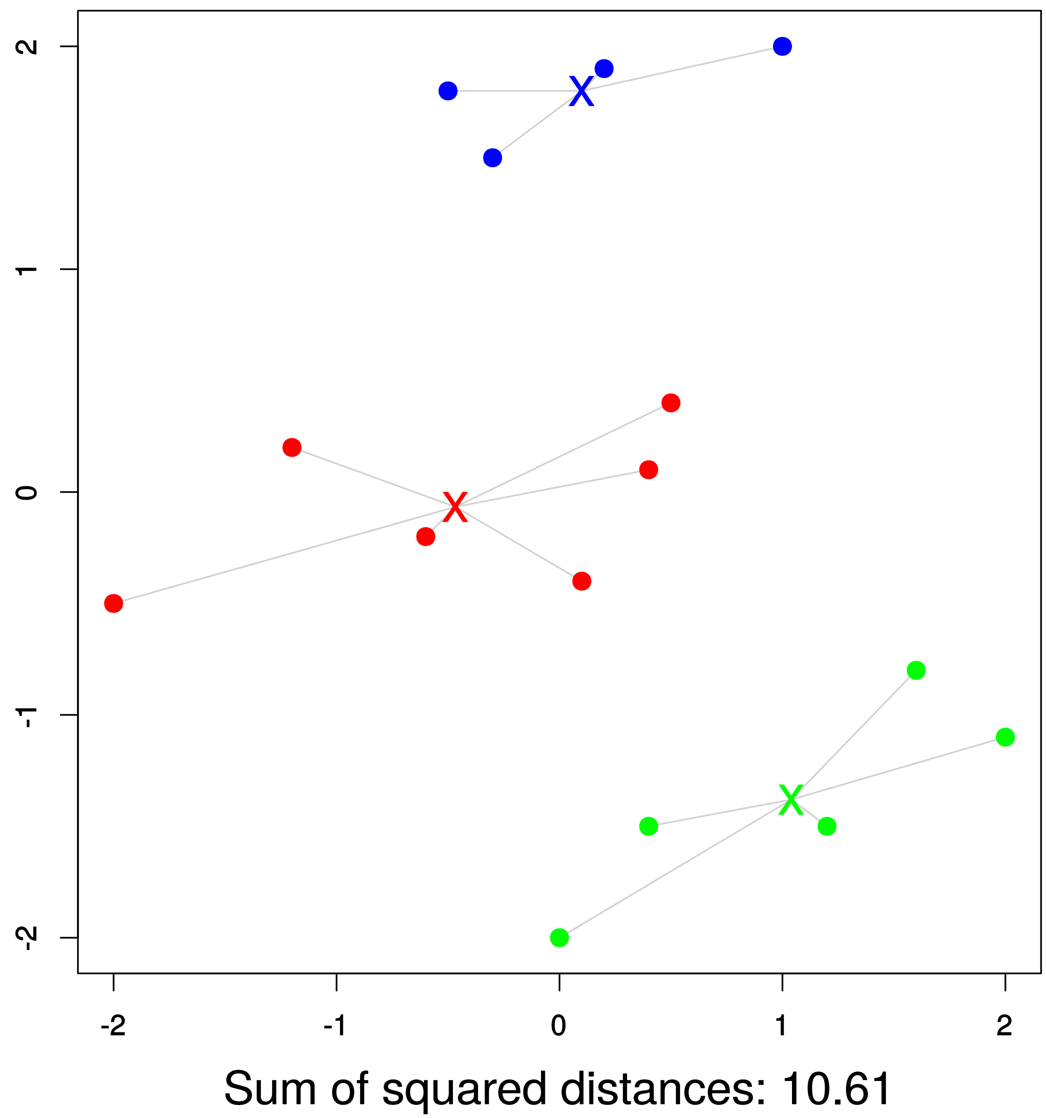


Sum of squared distances: 14.47

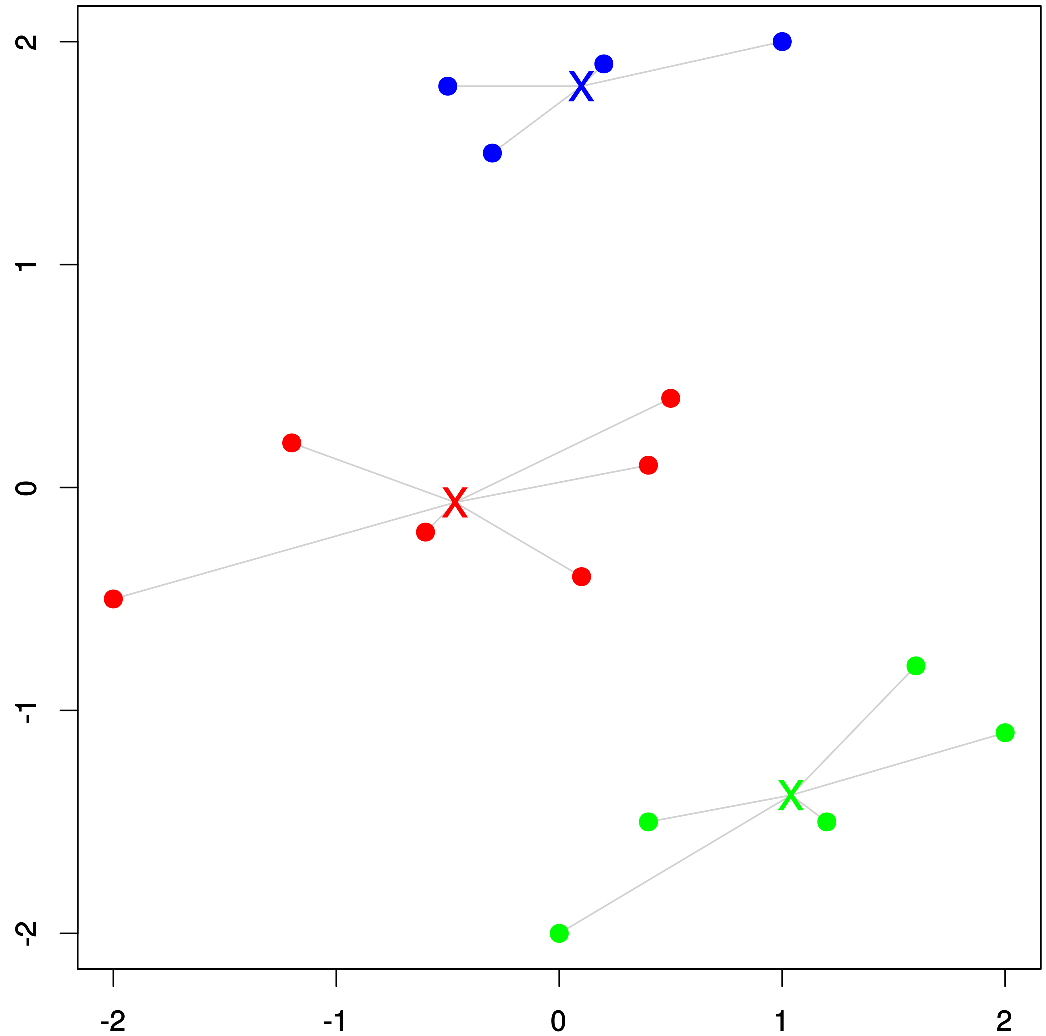
Prepočítame c_i



Prepočítame μ_i

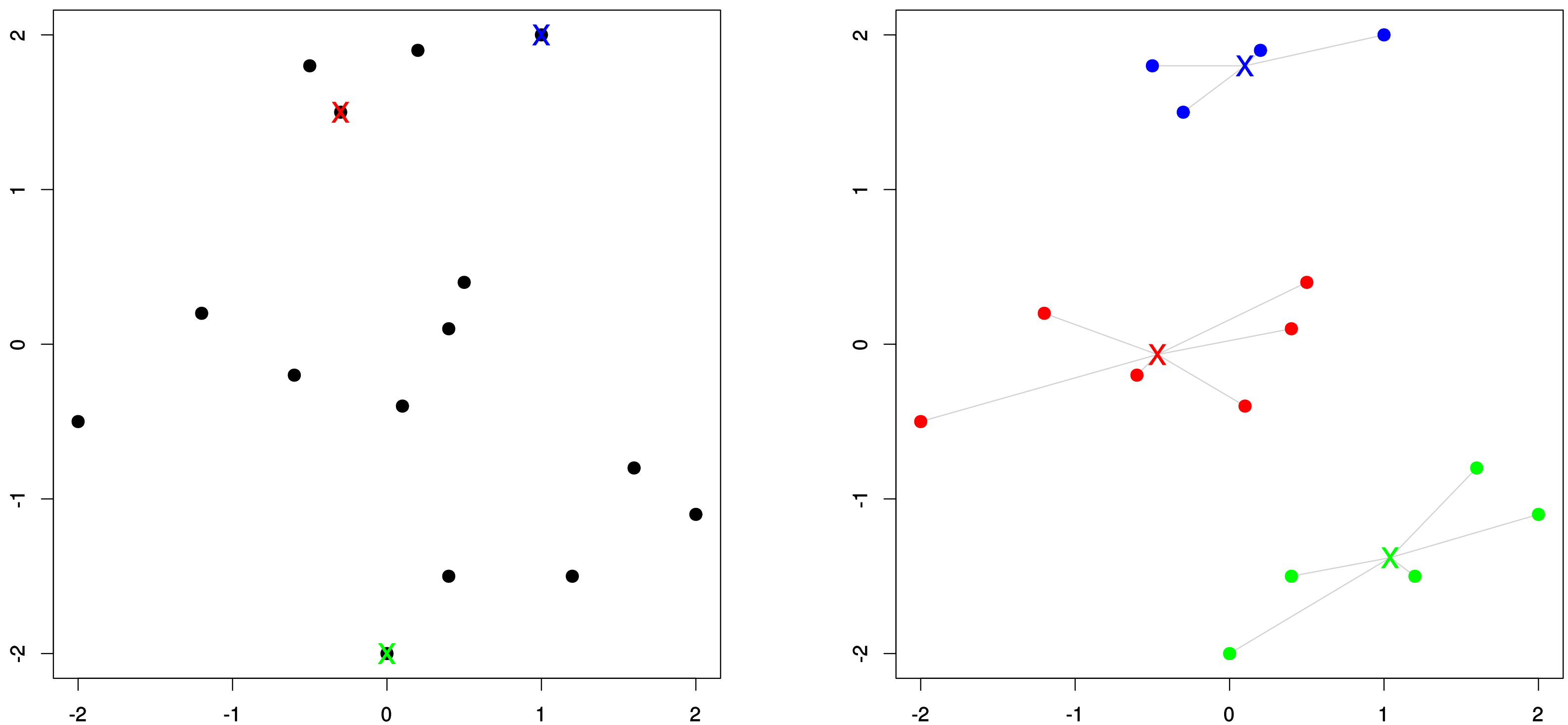


Prepočítame c_i (žiadna zmena, končíme)



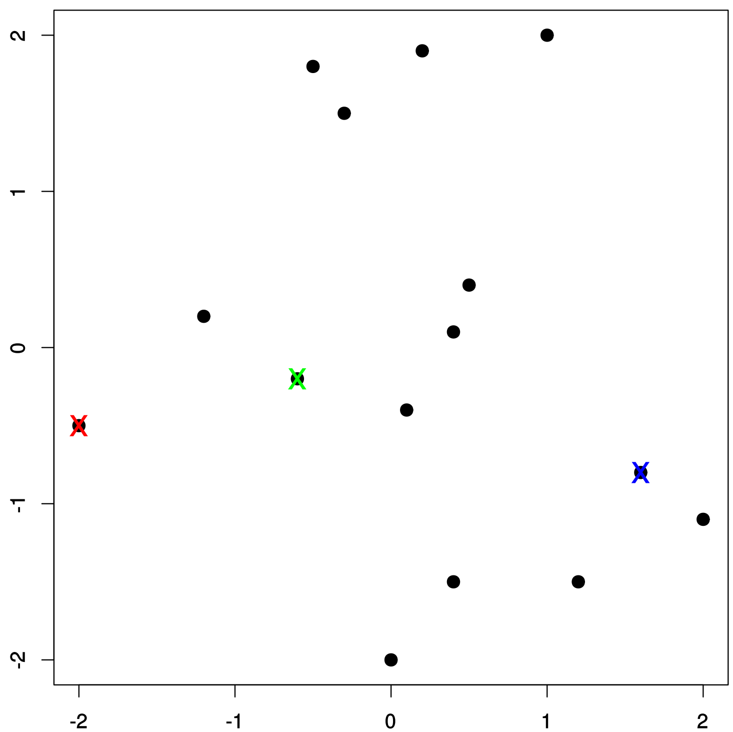
Sum of squared distances: 10.61

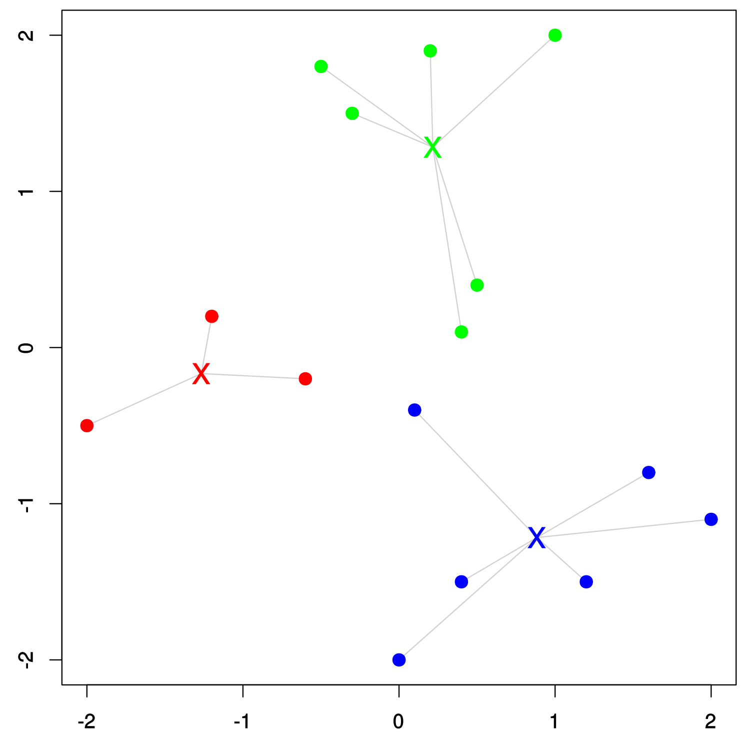
Príklady niekoľkých behov programu



Sum of squared distances: 10.61

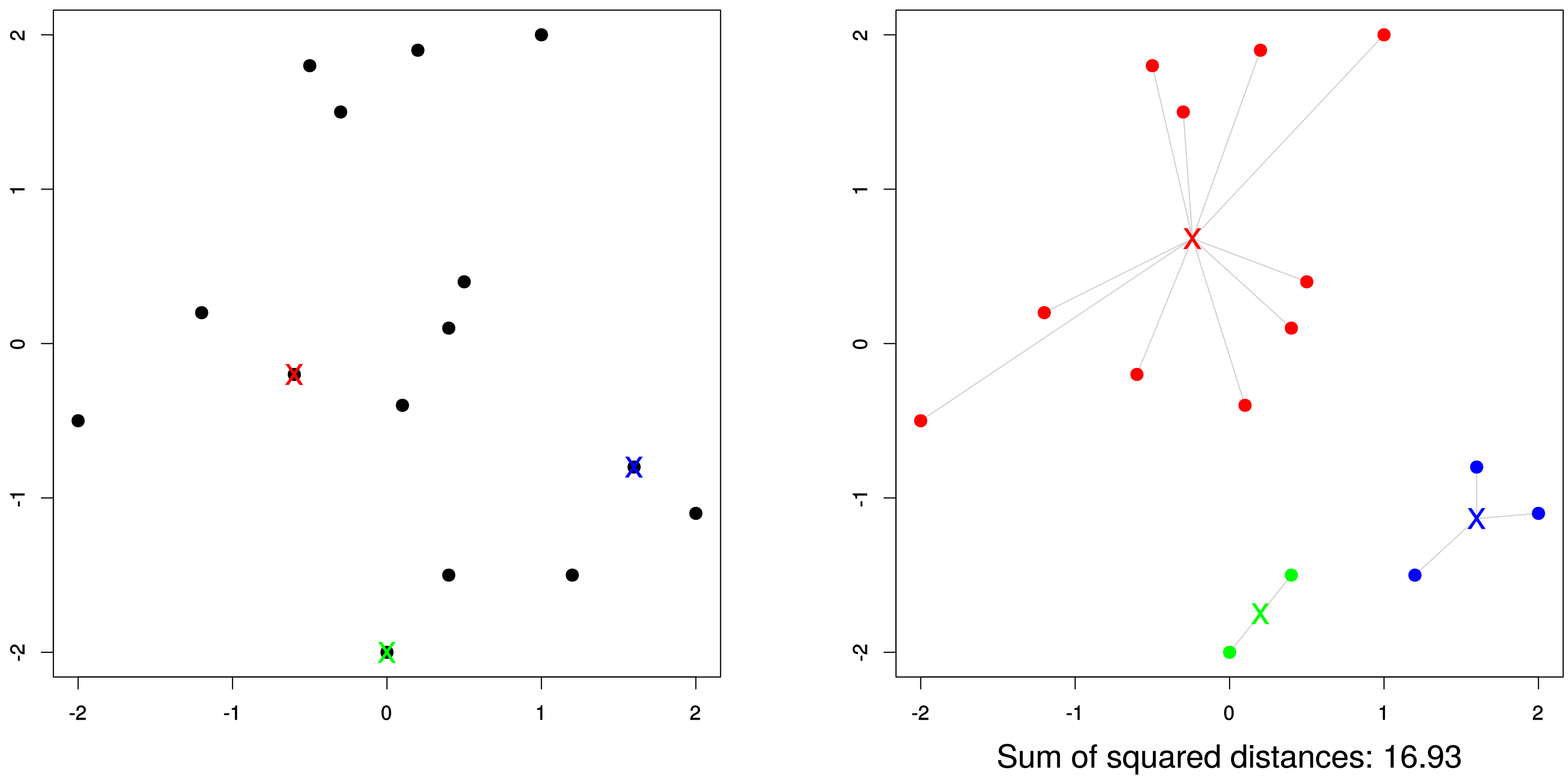
Príklady niekoľkých behov programu



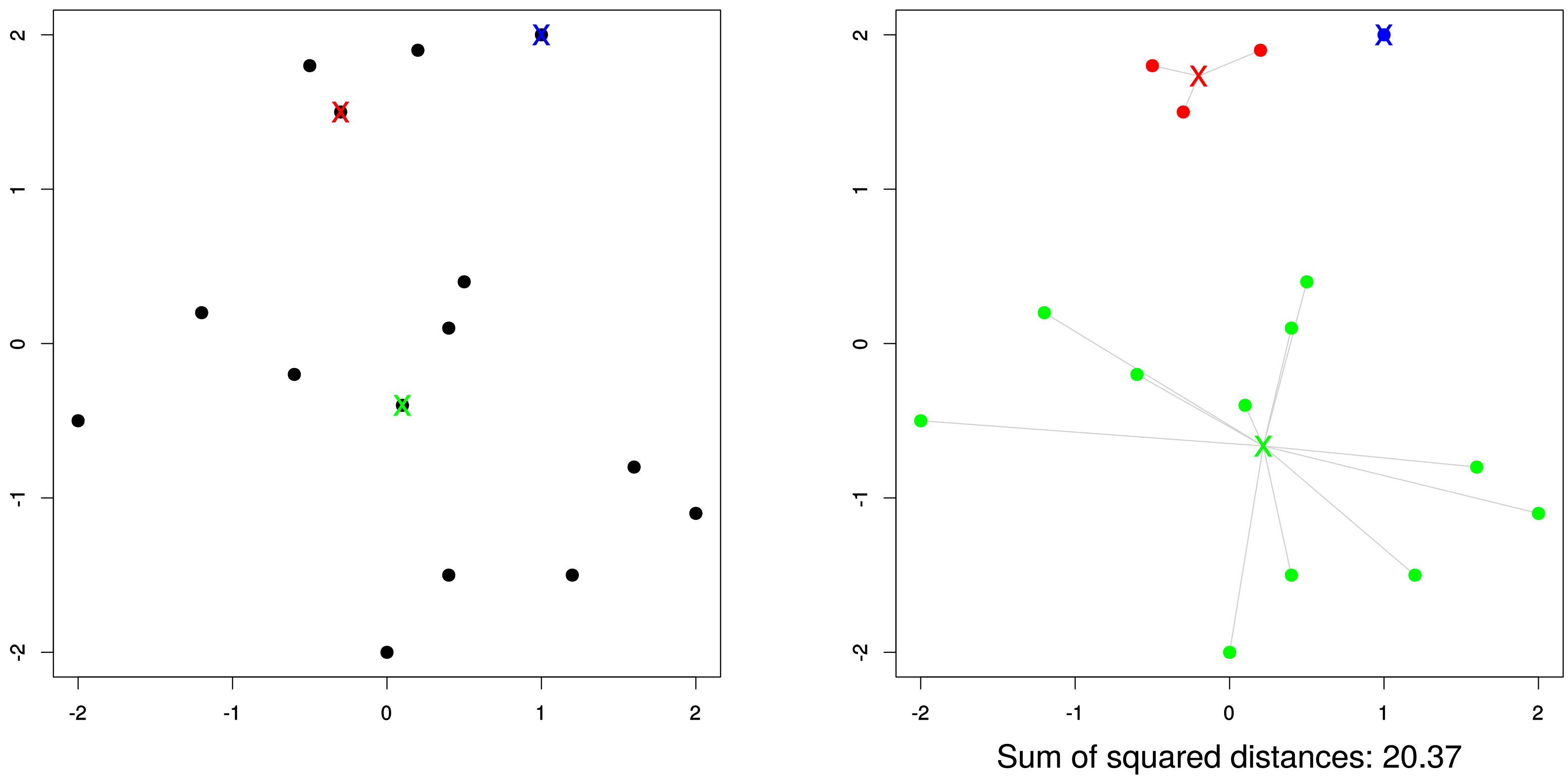


Sum of squared distances: 11.25

Príklady niekoľkých behov programu



Príklady niekoľkých behov programu



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 underexressed 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 enrichemnt 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 enrichemnt 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**:

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

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

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 χ^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 ≤ 0.05
- Test enrichment of 1000 functional categories in a list of genes, selectthose with p-value ≤ 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, 7.12.2023

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 dávaju 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)
- Zarovnania (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)
- **Linux pre používateľov** 1-AIN-500, Nagy (LS, 2K, 2kr) alebo
  Operačné systémy a počítačové siete 1-DAV-103 (ZS, 2P+2C, 5k)
- **Programovanie (1)** 1-MAT-130, Salanci (ZS, 2P+2C, 5kr) alebo
  Programovanie (1) 1-AIN-130 Blaho (ZS, 4P+4C, 9kr)

Pre pokročilejších

- **Seminár z bioinformatiky 1, 2** Brejová, Vinař (ZS/LS, 2S, 2kr)
  journal club o bioinformatických metódach
- **Úvod do bioštatistiky** 1-BMF-331 Waczulíková (LS, 2P+1C, 4kr)
  predmet pre biomedicínskych fyzikov a dátovú vedu
- **Vizualizácia dát** 1-DAV-105 Brejová, Bátorová (LS, 2P+2C, 5kr)
  vyžaduje základy Pythonu
- **Manažment dát** 1-DAV-105 Brejová, Boža, Vinař (LS, 1P+2C, 5kr)
  vyžaduje znalosť programovania, základy práce na príkazovom riadku

Teória grafov

Broňa Brejová

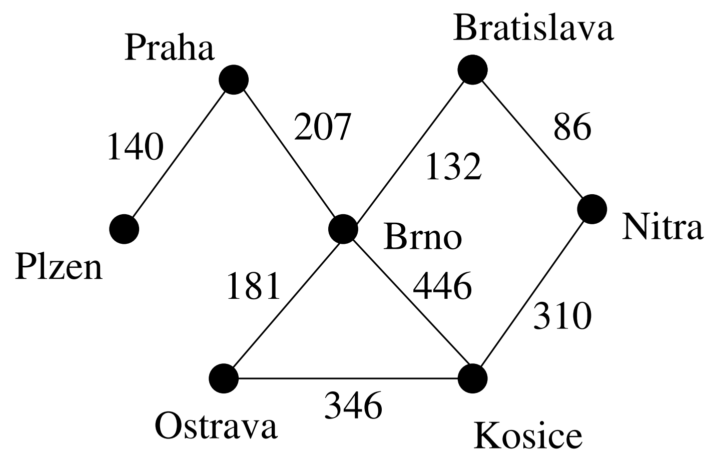
19.12.2020

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

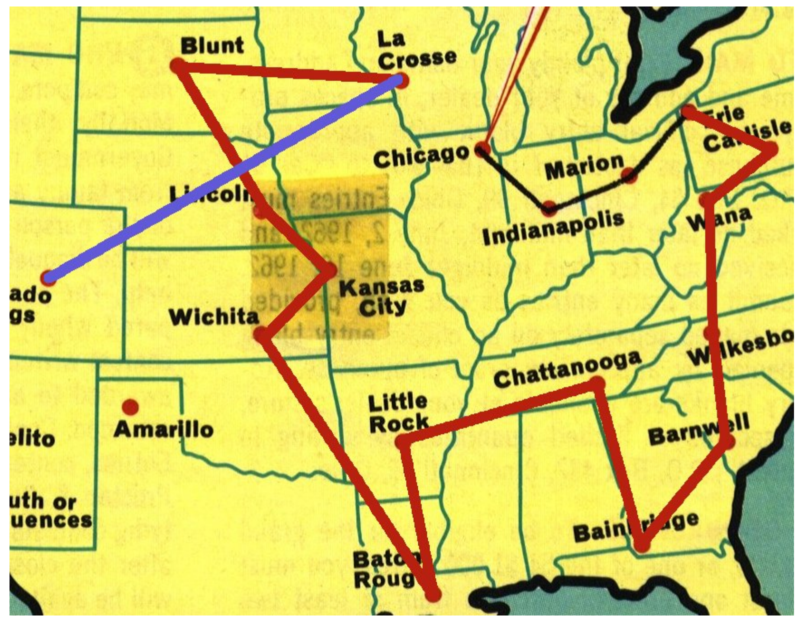
Problém obchodného cestujúceho

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

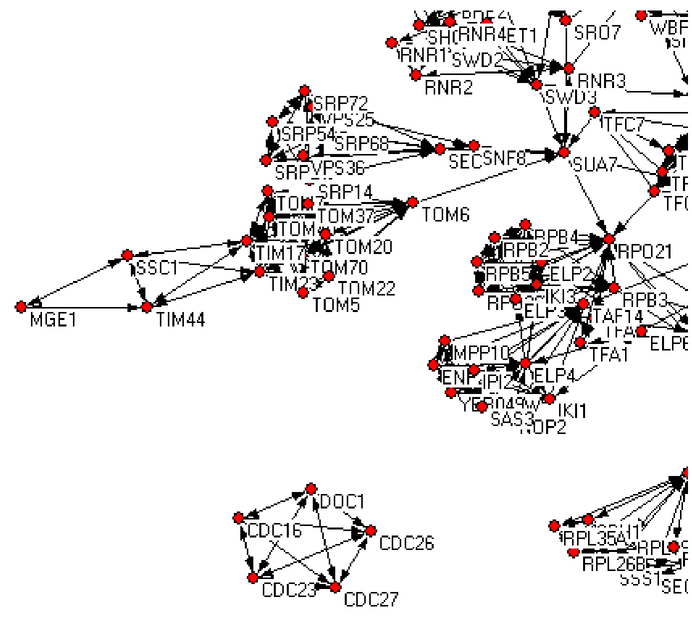
Metabolické dráhy zodp. **cestám**

Metabolické cykly zodp. **cyklom**

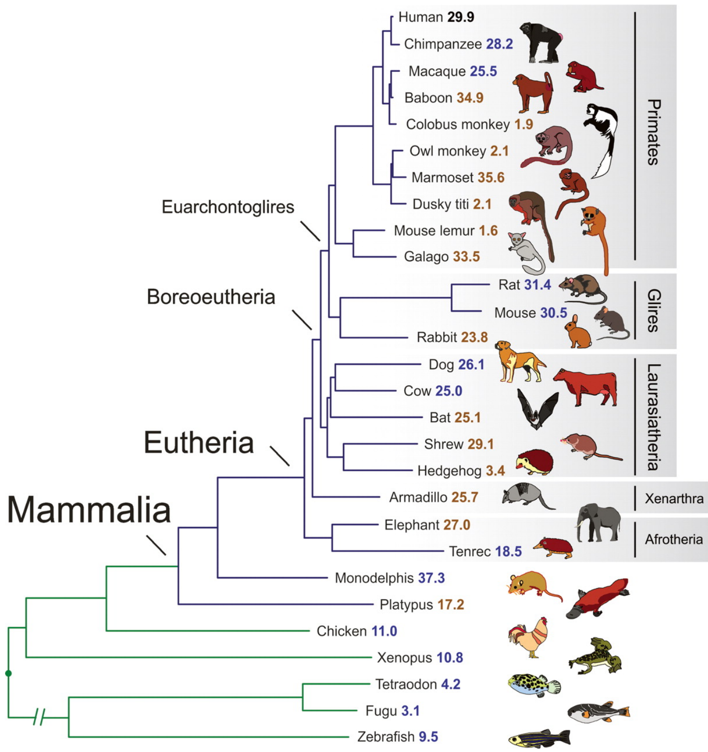
Kliky: Skupiny vrcholov priamo prepojené každý s každým

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)