SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Bioinformatika

UPGMA algoritam

Studenti:
Matija Bagić
Jan Bzik
Tibor Šefček

Zagreb, siječanj 2014.

Sadržaj

1. Algoritam 1
2. Primjer algoritma 2
3. Rezultati testiranja 6
4. Zaključak 11
5. Literatura 12

1. Algoritam

UPGMA (Unweighted Pair-Group Method with Arithmetic mean) je najjednostavnija metoda za izgradnju stabla. UPGMA generira ultrametrična stabla s korijenom. Veliki nedostatak UPGMA metode je da se pretpostavlja jednaka evolucijska brzina na svim linijama, odnosno da je stopa mutacija konstantna tijekom vremena i za sve linije u stablu (hipoteza molekularnog sata). To bi značilo da su svi krajnji čvorovi jednako udaljeni od korijena. U stvarnosti je jako mala vjerojatnost da će pojedine grane imati jednaku brzinu mutacije. Prema tome, UPGMA često generira krive topologije stabla.

Pseudokod algoritma:

UPGMA počinje s matricom udaljenosti D[1..n, 1..m]. Krajnji čvorovi se grupiraju u klastere (grozdove).

1. Pronaći par klastera (klaster **i** i **j**) s najmanjom udaljenošću u matrici udaljenosti D[**i**, **j**].

2. Definirati novi klaster koji obuhvaća klastere **i** i **j**:
   Klaster **i** je spojen granom na zajednički čvor koji je predak, isto vrijedi i za klaster **j**. Dakle, udaljenost D[**i**, **j**] je podijeljena na dvije grane, pa svaki od dva kraka ima duljinu D[**i**, **j**] / 2.

3. Ako su **i** i **j** zadnja dva klastera, stablo je završeno. Ako nisu, algoritam nalazi novi klaster **u**.

4. Odrediti udaljenost klastera **u** do svih ostalih klastera (**k**, gdje je **k** <> **i** ili **j**) da se dobije prosjek udaljenosti d(**k**, **i**) i d(**k**, **j**).
   -d(**k**, **u**) = (d(**k**, **i**) + d(**k**, **j**)) / 2

5. Vratiti se na korak 1 s jednim klasterom manje. Klasteri **i** i **j** više ne postoje, a dodan je klaster **u**.

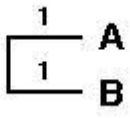
2. Primjer algoritma

1. CIKLUS

Neka je dana matrica udaljenosti koja predstavlja parove evolucijske udaljenosti (6 krajnjih čvorova koji predstavljaju 6 ulaznih nizova):

	A	B	C	D	E	F
A	0	2	4	6	6	8
B	2	0	4	6	6	8
C	4	4	0	6	6	8
D	6	6	6	0	4	8
E	6	6	6	4	0	8
F	8	8	8	8	8	0

Prvo je potrebno grupirati par krajnjih čvorova s najmanjom udaljenošću, a to su čvorovi A i B. Njihova udaljenost iznosi 2, što znači da duljina njihovih grana iznosi 2 / 2 = 1. Sada se konstruira podstablo:



Nakon grupiranja čvorova A i B, kojim se dobiva novi čvor AB, izračunava se nova matrica udaljenosti prema formuli:

dist (A, B), C = (distAC + distBC) / 2 = 4
dist (A, B), D = (distAD + distBD) / 2 = 6
dist (A, B), E = (distAE + distBE) / 2 = 6
dist (A, B), F = (distAF + distBF) / 2 = 8

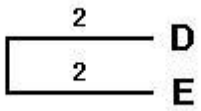
Drugim riječima, udaljenost između novog čvora AB i čvora C je prosjek udaljenosti između čvorova A i C (4) i B i C (4), što je (4 + 4) / 2 = 4. Isto vrijedi i za druge čvorove, a nakon izračuna se crta nova matrica udaljenosti s izračunatim udaljenostima. Taj se proces ponavlja sve dok u matrici ne ostane samo jedna udaljenost.

2. CIKLUS

	AB	C	D	E	F
AB	0	4	6	6	8
C	4	0	6	6	8
D	6	6	0	4	8
E	6	6	4	0	8
F	8	8	8	8	0

U ovoj matrici postoje dva para najkraćih udaljenosti, D i E (4), te AB i C (4). Izabire se par D i E, ali se je isto tako mogao izabrati i drugi par.

Duljina grana: 4 / 2 = 2

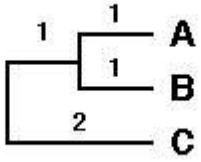


dist (D, E), AB = (distABD + distABE) / 2 = 6
dist (D, E), C = (distCD + distCE) / 2 = 6
dist (D, E), F = (distDF + distEF) / 2 = 8

3. CIKLUS

	AB	C	DE	F
AB	0	4	6	8
C	4	0	6	8
DE	6	6	0	8
F	8	8	8	0

Duljina grana: 4 / 2 = 2 i 4 / 2 - 1 = 1

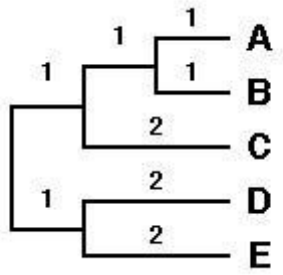


dist (AB, C), DE = (distABDE + distCDE) / 2 = 6
dist (AB, C), F = (distABF + distCF) / 2 = 8

4. CIKLUS

	ABC	DE	F
ABC	0	6	8
DE	6	0	8
F	8	8	0

Duljina grana: 6 / 2 - 2 = 1



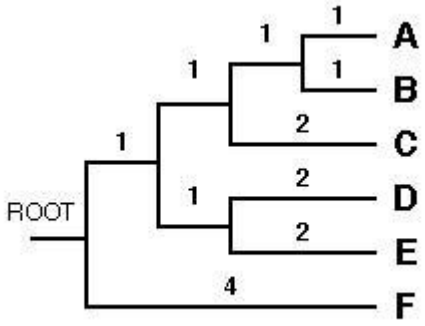
dist (ABC, DE), F = (distABCF + distDEF) / 2 = 8

5. CIKLUS

Zadnji korak sastoji se od grupiranja dva čvora koja su ostala, ABCDE i F.

	ABCDE	F
ABCDE	0	8
F	8	0

Duljina grana: 8 / 2 = 4 i 8 / 2 - 3 = 1



Iako ova metoda vodi do stabla bez korijena, UPGMA pretpostavlja jednake stope mutacija preko svih grana. Stoga teoretski korijen mora biti na jednakoj udaljenosti od svakog krajnjeg čvora, a u ovom slučaju je to dist (ABCDE), F / 2 = 4.

3. Rezultati testiranja

Rezultati testiranja za programski jezik Java

UPGMA algoritam implementiran u Javi sa Jukes Cantor i Kimura80 modelima DNA evolucije testiran je na duljinama sekvenci od 2000 znakova, 100000 znakova i 3 000 000 znakova. Vremena i maksimalni utrošak memorije prikazan je na sljedećim slikama.

Primjer 1

Duljina sekvence : 12755 znakova
Broj sekvenci : 60
Ukupno znakova: 765300
JukesCantor vrijeme izvođenja: 108 ms
Kimura vrijeme izvođenja :803ms
UPGMA vrijeme izvođenja: 46 ms
Korištena memorija: *24.53 MB*

Primjer 2.

Duljina sekvence: 12755
Broj sekvenci: 120
Ukupno znakova:1530600
JukesCantor vrijeme izvođenja:360ms
Kimura vrijeme izvođenja:2983ms
UPGMA vrijeme izvođenja: 156ms
Korištena memorija: *28.69 MB*

Primjer 3.

Duljina sekvence: 12755
Broj sekvenci: 241
Ukupno znakova: 3073955
JukesCantor vrijeme izvođenja: 1360ms
Kimura vrijeme izvođenja: 12108ms
UPGMA vrijeme izvođenja: 859ms
Korištena memorija: *33.21 MB*

Rezultati testiranja za programski jezik Python

UPGMA algoritam implementiran u Pythonu 2.7 sa modelima JC69 i K80 testiran je duljinama sekvenci od 12755 znakova, broj sekvenci je 241.

Primjer 1

Duljina sekvence : 12755 znakova
Broj sekvenci : 60
Ukupno znakova: 765300
JukesCantor vrijeme izvođenja: 5.55 sec
Kimura vrijeme izvođenja: 21.89 sec
UPGMA vrijeme izvođenja: 3.54 sec
Korištena memorija: *3.89 MB*

Primjer 2.

Duljina sekvence: 12755
Broj sekvenci: 120
Ukupno znakova: 1530600
JukesCantor vrijeme izvođenja: 21.87 sec
Kimura vrijeme izvođenja: 88.76 sec
UPGMA vrijeme izvođenja: 3.85 sec
Korištena memorija: *4.01 MB*

Primjer 3.

Duljina sekvence: 12755
Broj sekvenci: 241
Ukupno znakova: 3073955
Jukes-Cantor vrijeme izvođenja: 88.65 sec
Kimura vrijeme izvođenja: 351.40 sec
UPGMA vrijeme izvođenja: 4.86 sec
Korištena memorija: *4.89 MB*

Rezultati testiranja za programski jezik Ruby

UPGMA algoritam implementiran u Rubyu 1.9.3 sa modelima JC69 i K80 testiran je duljinama sekvenci od 12755 znakova, broj sekvenci je 241.

Primjer 1

Duljina sekvence : 12755 znakova
Broj sekvenci : 60
Ukupno znakova: 765300
JukesCantor vrijeme izvođenja: 12.660 sec
Kimura vrijeme izvođenja :17.600 sec
UPGMA vrijeme izvođenja: 0.049 sec
Korištena memorija: *4.03 MB*

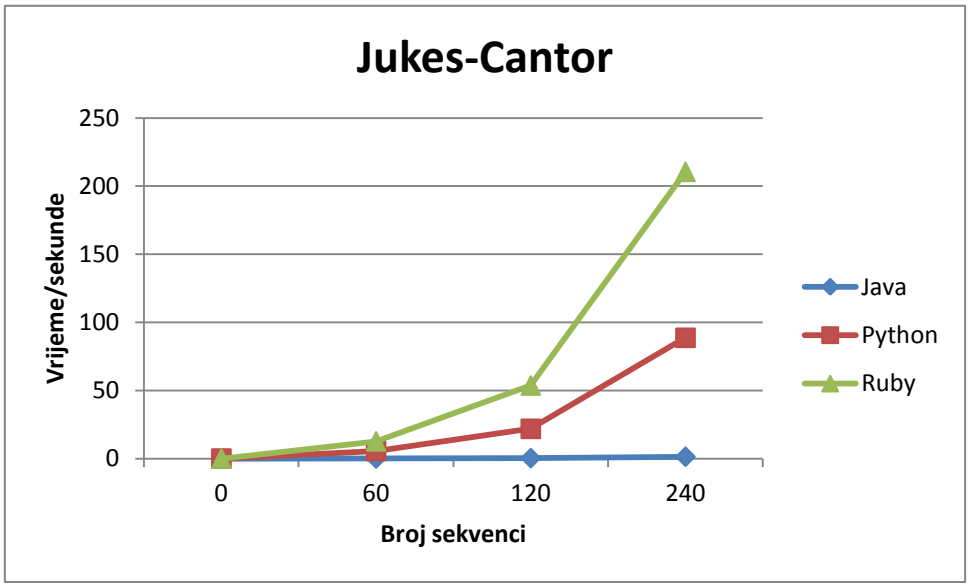
Primjer 2.

Duljina sekvence: 12755
Broj sekvenci: 120
Ukupno znakova:1530600
JukesCantor vrijeme izvođenja:53.645 sec
Kimura vrijeme izvođenja:68.733 sec
UPGMA vrijeme izvođenja: 0.382 sec
Korištena memorija: *4.15 MB*

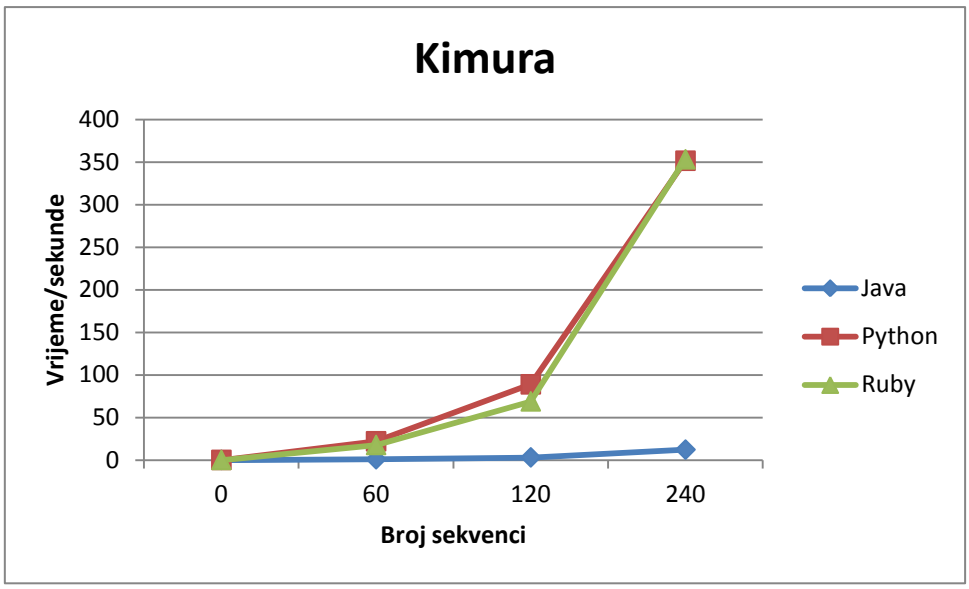
Primjer 3.

Duljina sekvence: 12755
Broj sekvenci: 241
Ukupno znakova: 3073955
Jukes-Cantor vrijeme izvođenja: 210.516 sec
Kimura vrijeme izvođenja: 353.101 sec
UPGMA vrijeme izvođenja: 2.922 sec
Korištena memorija: *4.48 MB*

Na temelju rezultata testiranja generirani su grafovi kako bi se lakše vidjele razlike u vremenima izvođenja pojedinog algoritma u svakom od tri gore navedena programska jezika.

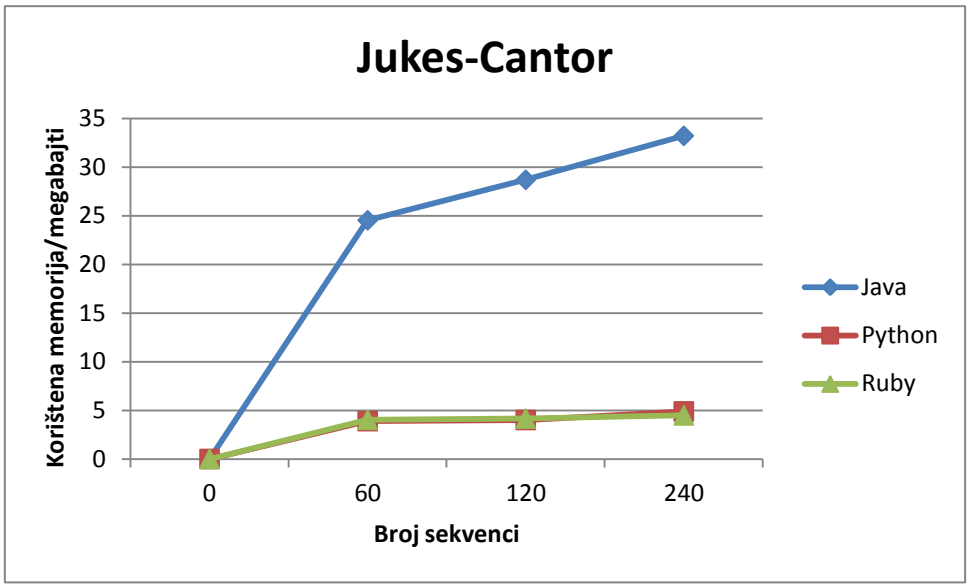


Grafički prikaz 1 - Vremena izvođenja za Jukes-Cantor model DNA evolucije



Grafički prikaz 2 - Vremena izvođenja za Kimura model DNA evolucije

Količina utrošene memorije je skoro pa jednaka i za Jukes-Cantor i za Kimura model DNA evolucije, što je i očekivano jer se modeli ne razlikuju jako puno, stoga je gornji rezultat vremena izvođenja za Kimura model DNA evolucije ponešto neočekivan. Java koristi najviše memorije, no rast memorije pada s povečanjem sekvenci, dok je za Python i Ruby utrošena memorija nešto manja.



Grafički prikaz 3 - Korištena memorija za Jukes-Cantor model DNA evolucije

4. Zaključak

Na temelju testiranja UPGMA algoritma na istim poravnatim sekvencama u programskim jezicima: Java, Ruby i Python jasno je da je izvođenje algoritma UPGMA u Javi višestruko brže nego u Rubyu i Pythonu, što smo i očekivali.

Također smo primijetili da "bottleneck" nije sam UPGMA algoritam već računanje matrica distanci između sekvenci putem Jukes Cantor i Kimura80 modela DNA evolucije. Iterira se po velikom broju znakova te se obavljaju operacije uspoređivanja što je vremenski poprilično zahtjevno. Ovdje se iz rezultata testiranja da zaključiti da je Jukes Cantor višestruko brži od Kimura80 modela no i najjednostavniji. Jukes Cantor i Kimura80 rade samo za sekvence koje nisu jako udaljene. Ukoliko su udaljene više od granice određene samom domenom prirodnog logaritma koji se koristi u obje formule oba modela DNA evolucije su neupotrebljiva. Postoje i drugi modeli poput T92 modela, GTR modela, HKY85 modela no oni u ovom seminaru nisu implementirani.

Krajnji zaključak je da se skriptni jezici poput Rubya i Pythona baš i ne mogu koristiti za zahtjevnije probleme analiziranja velikih Stringova što je jedan od osnovnih zahtjeva bioinformatike. Java je dorasla ovom zadatku te je čak i ovakva neoptimizirana implementacija dorasla zadatku.

5. Literatura

1) UPGMA Method,
   URL: http://www.sequentix.de/gelquest/help/upgma_method.htm

2) Construction of a distance tree using clustering with the Unweighted Pair Group Method with Arithmatic Mean (UPGMA), 9. rujan 1997.,
   URL: http://www.icp.ucl.ac.be/~opperd/private/upgma.html