SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2324

Orthobalancer: aplikacija za kreiranje skupova bioloških vrsta usporedive taksonomijske širine

Ivan Slijepčević

Zagreb, svibanj 2012.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da biste uklonili ovu stranicu obrišite naredbu `\izvornik`.

zahvale

SADRŽAJ

POPIS SLIKA

POPIS ALGORITAMA

POPIS TABLICA

1. Uvod

2. Teoretski uvod

kljucna uloga proteina

srodnost / evolucija / otkrivanje

2.1. Homologija proteina

Homologija u biološkom smislu predstavlja slične osobine među vrstama na različitim razinama organizacije života, poput organa, tkiva, stnice ili molekule. Homologne osobine uočene među jedinkama različitih vrsta obično upućuju na zajedničke pretke tih vrsta u evoluciji. Međutim, u molekularnoj biologiji termin homolog se često koristi i za naznačavanje sličnosti. bez obzira na genetsko srodstvo [1]

Za homologne sekvence proteina kažemo da su ortologne kad su direktni potomci neke sekvence u zajedničkom pretku, bez da su prošle duplikaciju gena. Drugim riječima, ortologne sekvence se mogu naći u jedinkama različitih vrsta, a obavljaju istu funkciju u svim tim vrstama. Paralogne sekvence su homologne sekvence koje su nastale od dvije različite kopije nekog gena koji je prošao kroz proces duplikacije gena u nekom zajedničkom evolucijskom pretku. Paralozi se mogu naći u jedinkama jedne ili više vrsta te obavljaju slične funkcije.

chart ortho-para

ideja...

3. Podaci

Orthobalancer radi sa primarnim proteinskim strukturama — sekvencama reziduuma, odnosno aminokiselina. Za zapis sekvenci koristi se standardizirani FASTA format. Orthobalancer za svoj rad koristi dvije NCBI-jevih[1] baza podataka od kojih jedna sadrži sekvence formatirane u FASTA format, a druga taksonomsko stablo živog svijeta.

3.1. FASTA format

3.2. Neredundantna baza

3.3. Taksonomsko stablo živog svijeta

3.4. Ulaz

Aplikacija kao ulaz prima nekolicinu paralognih proteina u FASTA formatu. Ako korisnik posjeduje samo sekvencu proteina, može ju zadati bez FASTA zaglavlja, no u tome je slučaju dužan dati ime unesenoj sekvenci. Nužno je imati ime za svaki ulazni paralog te je nužno da da su sva međusobno jedinstvena.

Dodatno, korisnik može specificirati čvorove taksonomskog stabla za čija podstabla smatra da sadrže zamjenjive vrste. Ponuđen je i osnovni skup zamjenskih čvorova za koje se vjeruje da bi mogli biti od koristi korisniku.

3.5. Izlaz

Web aplikacija za završetak izvođenja prikazuje tablicu balansiranih vrsta. Stupci tablice su imenovani po paralozima s ulaza. Retci su grupirani u zamjenske čvorove.

[1] National Center for Biotechnology Information

Svaki redak predstavlja jedan balansirani skup vrsta. U stupcu pod pojedinim paralogom nalazi se ortologna vrsta, a lijevo od svih vrsta je zapisan čvor na kojem su vrste tog retka balansirane. Primjer tablice se može vidjeti naslici 3.1.

Exchangeable nodes	Balanced node	CAL1	COF1
Mammalia	Ornithorhynchus anatinus	Ornithorhynchus anatinus	Ornithorhynchus anatinus
	Monodelphis domestica	Monodelphis domestica	Monodelphis domestica
	Equus caballus	Equus caballus	Equus caballus
	Canis lupus familiaris	Canis lupus familiaris	Canis lupus familiaris
	Ailuropoda melanoleuca	Ailuropoda melanoleuca	Ailuropoda melanoleuca
	Ovis aries	Ovis aries	Ovis aries
	Bos taurus	Bos taurus	Bos taurus
	Sus scrofa	Sus scrofa	Sus scrofa
	Callithrix jacchus	Callithrix jacchus	Callithrix jacchus
	Macaca fascicularis	Macaca fascicularis	Macaca fascicularis
	Macaca mulatta	Macaca mulatta	Macaca mulatta
	Nomascus leucogenys	Nomascus leucogenys	Nomascus leucogenys
	Pan troglodytes	Pan troglodytes	Pan troglodytes
	Homo sapiens	Homo sapiens	Homo sapiens
	Pongo abelii	Pongo abelii	Pongo abelii
	Primates	Otolemur garnettii	Papio anubis
	Oryctolagus cuniculus	Oryctolagus cuniculus	Oryctolagus cuniculus
	Mus musculus	Mus musculus	Mus musculus
	Rattus norvegicus	Rattus norvegicus	Rattus norvegicus
	Eutheria	Mustela putorius furo	Mus spretus
Sauropsida	Gallus gallus	Gallus gallus	Gallus gallus
	Taeniopygia guttata	Taeniopygia guttata	Taeniopygia guttata
	Squamata	Anolis carolinensis	Gekko japonicus
Actinopterygii	Salmo salar	Salmo salar	Salmo salar
	Osmerus mordax	Osmerus mordax	Osmerus mordax
	Esox lucius	Esox lucius	Esox lucius
	Epinephelus coioides	Epinephelus coioides	Epinephelus coioides
	Tetraodon nigroviridis	Tetraodon nigroviridis	Tetraodon nigroviridis
	Percomorpha	Sparus aurata	Anoplopoma fimbria
	Danio rerio	Danio rerio	Danio rerio
	Clupeocephala	Ictalurus punctatus	Oncorhynchus mykiss
	Ancestral node	**unbalanced**	
	Percomorpha	Dicentrarchus labrax	
	Clupeocephala	Ictalurus furcatus	
Amphibia	Xenopus (Silurana) tropicalis	Xenopus (Silurana) tropicalis	Xenopus (Silurana) tropicalis
	Xenopus laevis	Xenopus laevis	Xenopus laevis
unclassifiable homologues		synthetic construct	
		Saccoglossus kowalevskii	
		Trichoplax adhaerens	

Slika 3.1: Izlazna tablica sa web stranice Orthobanalcera

Također, završna stranica sadrži poveznice za preuzimanje generiranih datoteka tijekom izvođenja. Datoteke su opisane u nastavku.

4. Metode

pipeline neki dijagram za pipeline (sequence / activity / state)

tax dio pseudokod

neki sequence dijagram za sve

izlaz

(prebaciti u neki teex file za server) server slike (ulaz, zamjenjivi, izvršavanje, kraj, error)

5. Implementacija

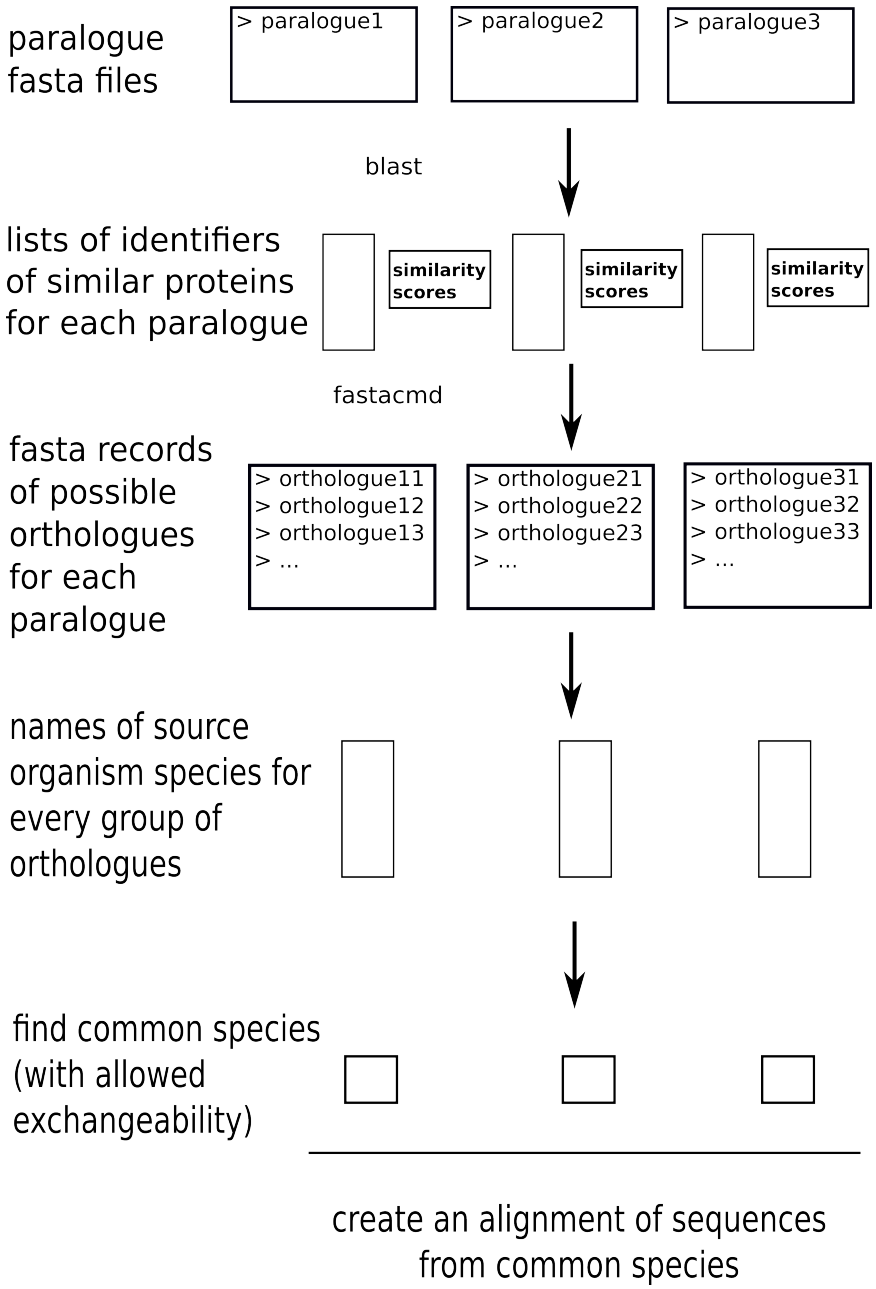
Aplikacija je pisana u programskom jeziku Python verzije 2.7. Aplikacija se dijeli u nekoliko zasebnih cjelina. U središtu aplikacije nalazi se cjovovod koji poziva alate poput BLAST-a i fastacmd-a za komuniciranje sa NCBI-jevom neredundantnom bazom, zatim dio aplikacije za odabir i balansiranje vrsta na taksonomskom stablu te alat mafft za poravnanje sekvenci. Pored cjevovoda implementirana je web aplikacija kao korisničko sučelje za cijeli program. Web aplikacija je implementirana koristeći Flask microframework, dok su operacije na klijentskoj strani implementirane u javascriptu uz korištenje biblioteke jQuery.

5.1. Cjevovod

Cjevovod je arhitektonski programski obrazac u kojem prolaze kroz filtre koji su postavljeni jedan za drugim. Time se simulira jedan tok koji ulazne podatke transformacijom kroz filtre generira izlazne podatke. U ovome projektu cjevovodna arhitektura je samo logički kostur koji se enkapsulira unutar razreda *Pipeline*. Iako je u začetku razvoja aplikacije svaki filter bio zaseban proces, vrlo ubrzo je ustanovljeno kako većina filtera generira podatke koji su potrebni na raznim mjestima u cijeloj aplikaciji te se činilo lakše imati sve podatke u memoriji pojedinog cjevovoda. To je omogućilo da razred *Pipeline* naslijedi razred *Thread* iz modula *threading* te se može pozivati kao zasebna dretva.

Tok cjevovoda se može vidjeti na slici 5.1. Ulaz u cjevovod predstavljaju paralogni proteini u FASTA formatu koje zadaje korisnik. Ti se podaci zadaju pri stvaranju objekta *Pipeline* kako bi se mogli zapisati na disk u direktorij vezan za instancu *Pipeline-a*. Stvarni objekt kojeg prima konstruktor *Pipeline-a* je rječnik prilagođen uporabi servera, što je detaljnije opisano u odjeljku 5.3.

Pri pokretanju cjevovoda za svaku se od unesenih sekvenci stvara objetk razreda *ProteinHolder* prilikom čega se obavljaju pozivi filtara nezavisnih za svaku pojedinu sekvencu. Prvi filter koji se koristi je alat *BLAST* te je izveden kao poziv zasebnog



Slika 5.1: Protok podataka kroz cjevovod. Prikazuju se podaci rađe nego filtri radi boljeg uvida u rad cjevovoda

izvršnog programa *blastall* na sljedeći način:

```
blastall -p blastp -i <ulaz-FASTA> -d <nr-baza> -m 8 -a <broj-dretvi>
```

Argument *p* s parametrom *blastp* označava programu da se koristi algoritam za uspoređivanje jedne ulazne sekvence amino kiselina sa bazom proteinskih sekvenci. S argumentom *i* se zadaje put do ulazne datoteke s FASTA sekvencom. Nadalje, argument *d* prima put na disku do NCBI-jeve nereduntantne baze sekvenci koja je prethodno

formatirana za pretraživanje sekvenci u FASTA formatu. Argument *m* određuje format ispisa koji generira *blastall*, a parametar 8 označava tabularni ispis bez dodatnih komentara koji je pogodan za parsiranje. Konačno, argument *a* upućuje *blastall* na broj dretvi koji treba koristiti kako bi se ubrzalo njegovo izvođenje. Broj dretvi se u Orthobalanceru može postaviti prilikom instalacije.

Izlaz koji generira *BLAST* predstavlja informacije o sekvencama sličnim ulaznoj sekvenci, odnosno informacije o potencijalnim ortolozima za ulazni paralog. Svaki redak, između ostalih, sadrži dvije bitne informacije: jedinstveni ključ sekvence u neredundantnoj bazi — gi-broj — te ocjenu sličnosti ulaznome paralogu. Nakon parsiranja tog izlaza ocjene sličnosti se spremaju za kasniju upotrebu, a gi-brojevi se zapisuju u privremenu datoteku za sljedeći korak u cjevovodu.

Sljedeći filter je izvršni program *fastacmd* koji za svaki gi-broj pronalazi i ispisuje sekvencu u FASTA formatu, također koristeći NCBI-jevu neredundantnu bazu. Program se poziva ovako:

```
fastacmd -i <ulaz> -d <nr-baza>
```

Budući da program *fastacmd* dolazi u paketu zajedno sa *BLAST* alatima, argumenti imaju sličnu konotaciju kao i za *blastall*: s argumentom *i* se zadaje ulazna datoteka, a s *d* put do neredundantne baze.

Izlaz *fastacmd*-a se parsira pomoću razreda FastaRecord. Svaka dobivena sekvenca kao kandidat za ortologa dobiva instancu razreda FastaRecord u kojoj je sadržana sekvenca te sve bitne informacije iz zaglavlja pojedinog FASTA zapisa. Dodatno, instanci FastaRecord se pridružuje ocjena sličnosti sekvence koju opisuje prema ulaznom paralogu. Kako bi bilo lakše objasniti što je preuzeto iz zaglavlja pojedinog FASTA zapisa, bitno je imati na umu strukturu neredundantne baze što je objašnjenu u poglavlju 3. Naime, ako je za nekoliko proteina zabilježeno da imaju identičnu sekvencu, tada će zaglavlje takve FASTA sekvence u neredundantnoj bazi sadržavati spojene podatke o navedenim proteinima. Zato objekt razreda FastaRecord sadrži listu elemenata zaglavlja, gdje se svaki od tih elemenata opisuje n-torkom (gi-broj elementa, ime vrste, ime proteina, zastavica: najbolja sekvenca za ovaj gi-broj). Iako je poznato da neredundantna baza sadrži sekvence sakupljene iz raznih baza što implicira činjenicu da zaglavlja pojedinih sekvenci ne moraju imati identičan oblik, uočeno je određeno pravilo po kojem *fastacmd* ispisuje sekvence te se ono koristi kao heuristika ugrađena u parser. Pretpostavljeni oblik pojedinog elementa zaglavlja je sljedeći:

```
>gi|"gi-broj"|nebitne-informacije IME PROTEINA
[IME VRSTE]nebitna-informacija.
```

Na primjer:

>gi|**344243907**|gb|EGW00011.1| **Cofilin-1** [**Cricetulus griseus**]

Na kraju obrade podataka za pojedini paralog s ulaza prolazi se listom svih objekata FastaRecord te se za svaku pronađenu vrstu odabire sekvenca s najboljom ocjenom sličnosti. Time nastaje jedan podskup sekvenci za koje možemo reći da predstavljaju grupu ortologa za dani paralog.

Nakon što je svaki paralog opisan jednim objektom razreda *ProteinHolder* potrebno je odrediti postoji li koji gi-broj koji se može pronaći u grupama ortologa različitih paraloga. Donesena je odluka kako to svojstvo nije poželjno jer jedan te isti protein ne želimo imati kao predstavnika neke vrste u različitim grupama ortologa. Za potrebe ove funkcionalnosti dodani su razredi *BestScore* i *BestScoreCollection*. Razred *BestScore* sadrži informaciju koja upućuje u kojoj se grupi ortologa, na kojem FASTA zapisu te sa kojim elementom zaglavlja FASTA zapisa nalazi najbolje ocijenjena sekvenca za dani gi-broj identifikator. Razred *BestScoreCollection* služi kao sučelje za korištenje rječnika najboljih sekvenci, a nudi metode za ažuriranje rječnika kandidatima za najbolje sekvence te metodu za dohvat trenutno postavljene najbolje instance razreda *BestScore*. Skica algoritma za pronalazak najbolje ocijenjenih dana je algoritmom 1. Nakon provedbe algoritma nepoželjni proteini imaju spuštenu zastavicu najbolje sekvence za svoj gi-broj te se ne razmatraju u nastavku programa.

Ovdje bi se još mogla dodati funkcionalnost kojom bi se za vrstu kojoj je protein odbačen pronašao sljedeći najbolji nezauzeti protein, no to nije razmatrano jer predstavlja rubni slučaj osnovne funkcionalnosti odbacivanja sekvenci, što je već samo po sebi rubni slučaj i vrlo se rijetko dešava.

Najbitniji korak u cjevovodu je pronalaženje i balansiranje zajedničkih vrsta na stablu taksonomije živog svijeta. Ovaj korak je detaljno opisan u odjeljku . Ovom se filtru kao ulaz predaju kompletni opisnici proteina, odnosno objekti razreda *ProteinHolder* iz kojih sam filter povlači sve potrebne informacije. Nakon što završi obrada, izlaz filtra je predočen višeslojnim rječnikom koji razdvaja podatke po sljedećim slojevima: grupe ortologa, grupe zamjenskih čvorova u taksonomskom stablu, grupe balansiranih čvorova u taksonomskom stablu te su zadnje vrijednosti balansirane vrste.

Zadnji korak u cjevovodu prije zapisivanja svih skupljenih podataka na disk je poravnanje sekvenci dobivenih kao izlaz programa *fastacmd*. Za poravnanje se koristi alat *mafft* te mu je ulaz datoteka sa zapisanim sekvencama u FASTA formatu, a izlaz datoteka sa istim, ali poravnatim sekvencama. *mafft* se poziva na sljedeći način:

```
mafft --auto <ulaz> > <izlaz>
```

Algorithm 1 Nalaženje najbolje ocijenjenih proteina

```
function PROTEINHOLDER::IZRAČUNAJNAJOCJENE
    ocjene ← BestScoreCollection()
    for all fastaRecord ∈ this.records do
        for all element ∈ fastaRecord.elementiZaglavlja do
            ocjena ← BestScore(element)
            ocjene.auriraj(ocjena, fastaRecord.ocjena)
        end for
    end forreturn ocjene
end function
function PROTEINHOLDER::PROPAGIRAJ(najOcjene)
    for all fastaRecord ∈ this.records do
        for all element ∈ fastaRecord.elementiZaglavlja do
            najOcjena ← najOcjene.dohvati(element['gid'])
            if ¬najOcjena.provjeriElement(element) then
                element['zastavicaNajbolji'] ← ⊥
            end if
        end for
    end for
end function
function PRONAĐINAJOCJENE(proteinHolders)        ▷ funkcija je isječak iz
cjevovoda
    najOcjene ← newBestScoreCollection()
    for all protein ∈ proteinHolders do
        ocjene ← protein.izraunajNajOcjene()
        najOcjene.aurirajKolekcijom(ocjene)
    end for
    for all protein ∈ proteinHolders do          ▷ propagacija najboljih ocjena
        protein.propagiraj(najOcjene)
    end for
end function
```

Argument –auto upućuje *mafft* da automatski odabere najbolju strategiju poravnavanja sekvenci, uzimajući u obzir veličinu podataka. Izlaz se direktno preusmjerava u novu datoteku na disk jer poravnate sekvence nisu potrebne u memoriji.

5.2. Nalaženje zajedničkih vrsta

Pronalazak zajedničkih vrsta najbitniji je modul Orthobalancera. U suštini, ovaj modul nije logički vezan za nalaženje ortologa, no za potrebe Ortobalancera je svojim sučeljima prilagođen njegovom cjevovodu. Nalaženje zajedničkih vrsta među skupovima ulaznih vrsta predstavlja jednu novu dimenziju u pronalasku ortolognih proteina. Konvencionalno traženja ortologa[**?** **?**] jest kvalitetnije jer se pretraživanje odvija na razini sekvence, ali je moguće samo za genome koji su u potpunosti istraženi i zapisani u baze podataka.[**?**] Podizanje potrage za ortolozima na razinu vrsta omogućava da se ortolozi pronađu i među vrstama čiji genomi nisu još zabilježeni.

Ovaj modul za svoj rad koristi podatke iz NCBI-jeve *Taxonomy* baze. Koristi se čvorovi taksonomskog stabla živog svijeta i znanstvena imena pridijeljena čvorovima. Čvorovi su identificirani svojim jedinstvenim identifikatorima *tax_id*, a svaki čvor ima pridruženo jedno ili više korištenih imena. Samo jedno od tih imena je označeno kao znanstveno i također je jedinstveno za svaki čvor, uz neke iznimke poput sintetički stvorenih vrsta koje dijele znanstveno ime *Synthetic construct*. Datoteke koje sadrže podatke iz ovih baza su prilično velike te njihovo učitavanje, prilagođavanje i korištenje najviše utječe na trajanje izvođenja programa.

Na početku rada modula svi podaci se prilagođavaju za potrebe modula. Iz cjevovoda se od svake instance *ProteinHolder*-a uzimaju zabilježene vrste, a iz popisa zamjenjivih čvorova koje zadaje korisnik se prikupljaju svi zadani čvorovi. Svim prikupljenim imenima se tada pridružuje *tax_id* iz baze.

Nakon toga se gradi taksonomsko stablo u memoriji. Stablo je izvedeno kao veliki rječnik kojem za ključeve koristi *tax_id*, a svaki čvor je objekt razreda *Node*. Razred *Node* sadrži sljedeće bitne podatke: *tax_id* roditeljskog čvora, listu djece, brojač za svaku grupu ortologa, ukupni brojač grupa, listu zamjenskih roditeljskih čvorova te zastavicu je li balansiran. Brojači se inicijaliziraju na nulu, a kasnije tijekom algoritma će koristiti kao broj vrsta pojedine grupe ortologa koje se nalaze pod danim čvorom. Ukupni brojač grupa govori koliko ortolognih grupa ima svoga predstavnika pod nekim čvorom. Lista zamjenskih roditeljskih čvorova se inicijalizira na praznu listu, a bit će popunjena svim čvorovima koji su od korisnika označeni kao zamjenski te se nalaze iznad trenutno razmatranog čvora. Zastavica za balansiranje koristi se kasnije tijekom postupka balansiranja podstabala ispod zamjenskih čvorova. Algoritam je opisan u nastavku, a dan je i njegov pseudokod 2.

Sljedeći koraci predstavljaju inicijalizaciju stabla. Najprije se za svaku ortolognu grupu prolazi kroz sve vrste. Za svaku vrstu pronalazi se njen čvor, odnosno list u

stablu. Brojač tog lista za trenutnu ortologlognu grupu se inicijalizira na 1. Istovremeno se ukupni brojač grupa u tome listu postavi na 1 te se pozove metoda koja propagira ukupni brojač grupa sve do korijena stabla, pazeći da ukupni brojač niti u jednom čvoru ne bude više od jednom povećan za neku grupu ortologa.

Nakon toga treba inicijalizirati zamjenske čvorove i pripadna podstabla. Za svaki zamjenski čvor poziva se rekurzivna funkcija koja se spušta do listova i svakome čvoru u njegovu listu zamjenskih roditeljskih čvorova dodaje *tax_id* zamjenskog čvora nad kojim je rekurzija pozvana. Time svaki čvor sadrži informaciju kojim zamjenskim podstablima pripada.

Slijedi algoritma 2.

Algorithm 2 Obilazak stabla — balansiranje vrsta

$ULAZ$: taksonomsko stablo, skup zamjenjivih čvorova, n skupova vrsta
$SREDINJESTRUKTUREPODATAKA$:
$IZLAZ$: višeslojni rječnik outkoji za svaki balansirani čvor sadrži listu vrsta predstavnica iz svake grupe g

function OBIDISTABLO($cvor, grupe, zamjenskiCvorovi$)
  if $vor.zamjenski()$ **then** ▷ metoda vraća istinu ako je čvor u zamjenskom podstablu
return $balansiraj(cvor, grupe, zamjenskiCvorovi)$
  else
    if $cvor.list()$ **then** ▷ metoda vraća istinu ako je čvor list
      for all $grupa \in grupe$ **do**
        $izlaz[grupa]['nezamjenski'] \leftarrow cvor.ime$
      end for
    else
      $izlaz \leftarrow$ new $izlaz$
      for all $dijete \in cvor.dijeca$ **do**
        if $dijete.ukupniBrojac = vor.ukupniBrojac$ **then**
          $izlaz \in izlaz \cup obidiStablo(dijete, grupe, zamjenskiCvorovi)$
        end if
      end for
    end if
  end ifreturn $izlaz$
end function

5.3. server

dio po dio

jQuery

komunikacija pipelinea i klijenta, log, dekoratori

automatizacija (posla kroz cjevovod->treba ići u neko uvodno poglavlje o namjeri)

robusnost

6. Rezultati

7. Zaključak

LITERATURA

[1] Andreas D. Baxevanis. *Bioinformatics and The Internet*. John Wiley & Sons, Inc., 2002. ISBN 9780471223924.

Orthobalancer: aplikacija za kreiranje skupova bioloških vrsta usporedive taksonomijske širine

Sažetak

Ključne riječi:

Orthobalancer: web application for creation of taxonomically balanced sets of orthologous protein sequences

Abstract

Keywords: