Fakulta matematiky, fyziky a informatiky, Univerzita Komenského v Bratislave

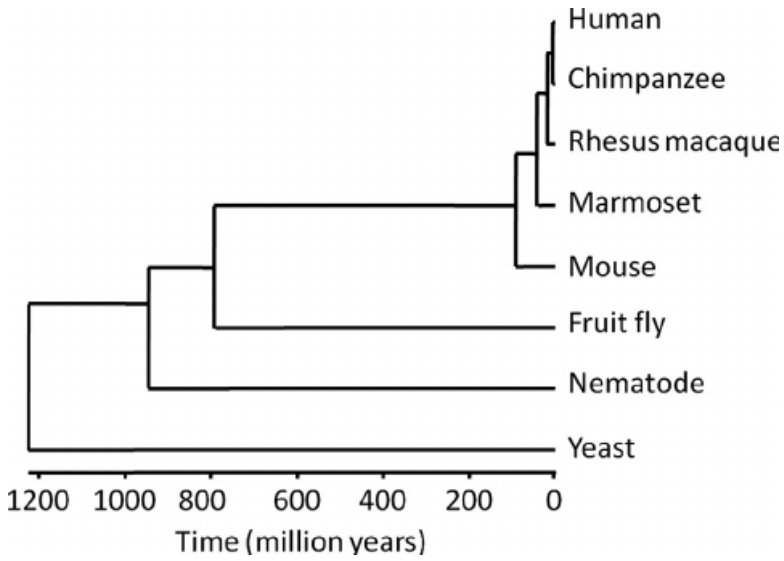
Algoritmy pre izometrickú rekonciliáciu génových stromov

Autor: Radoslav Chládek

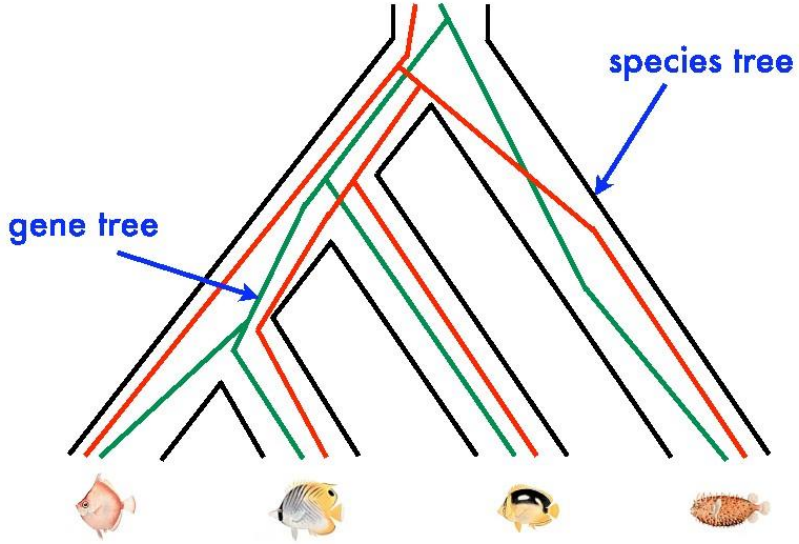
Školiteľka: Bronislava Brejová

Evolučné stromy

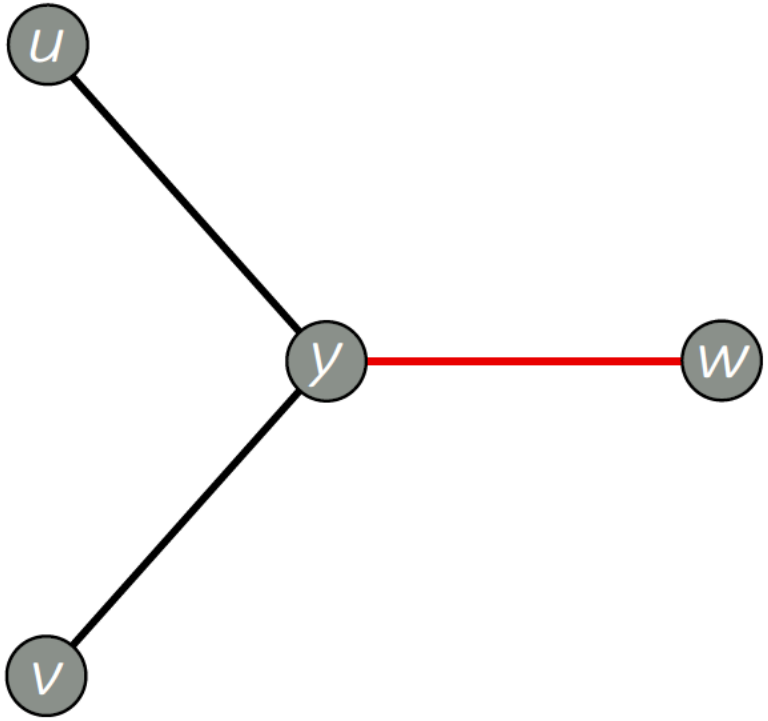
- druhový strom
  - zobrazuje evolučné vzťahy medzi biologickými druhmi
  - vnútorné vrcholy predstavujú speciácie

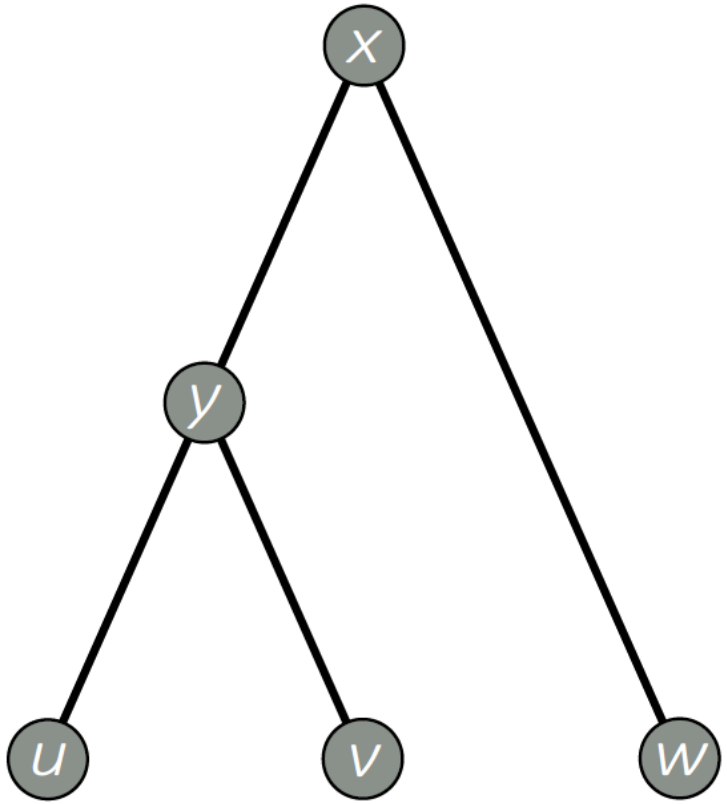


- génový strom
  - zobrazuje vývoj jedného génu
  - listy predstavujú rodinu génov v dnešných druhoch
  - vnútorné vrcholy sú duplikácie alebo speciácie



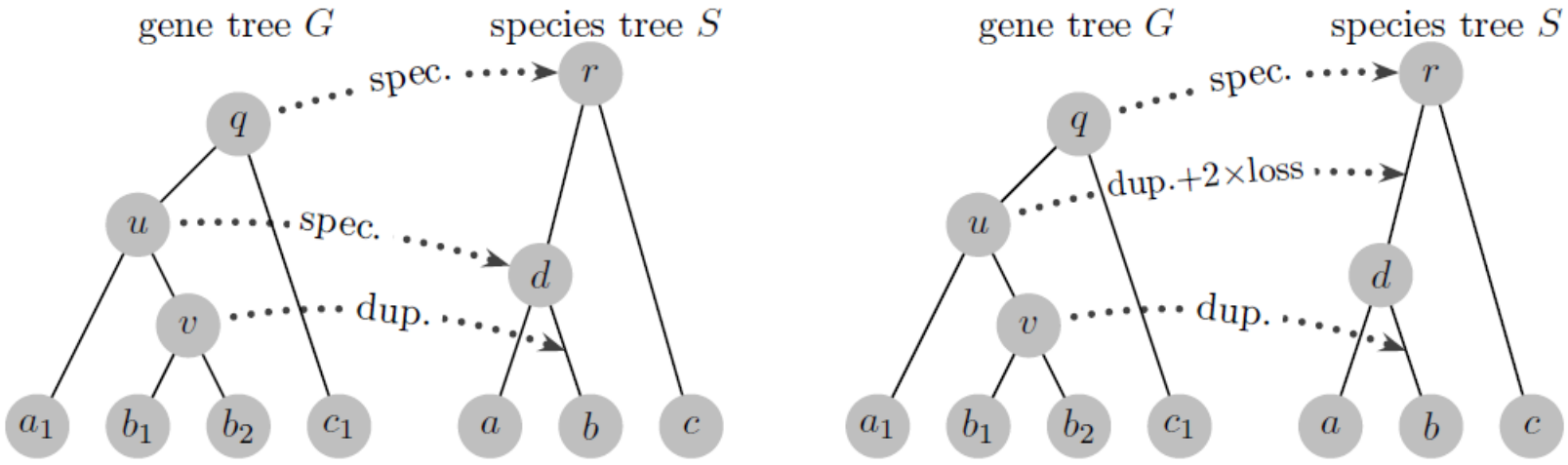
Nezakorenený/zakorenený strom





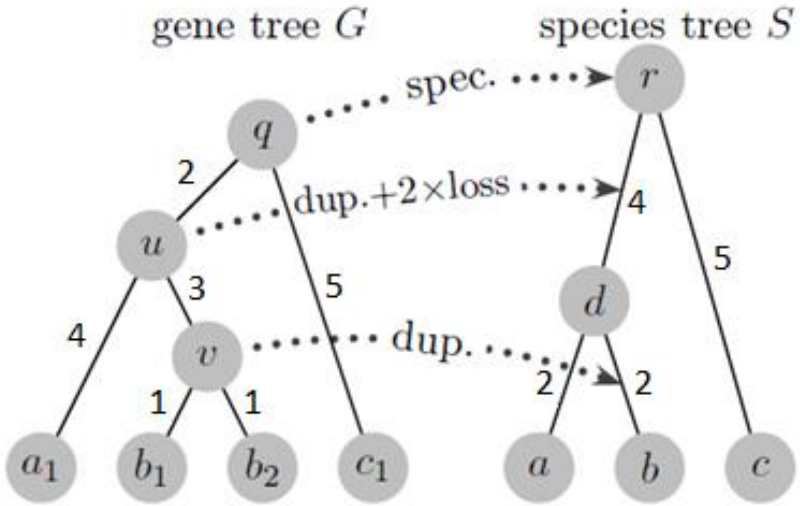
Rekonciliácia génového stromu

- vstup: druhový strom *S*, génový strom *G*, mapovanie listov G do listov S
- úloha: namapovať vnútorné vrcholy *G* do vrcholov alebo hrán *S*



Súčasný stav problematiky

- viacero algoritmov pre najúspornejšiu (parsimonious) rekonciliáciu (=len topológia)
  - problém sa študuje od r. 1979
- izometrická rekonciliácia = okrem topológie sa berú do úvahy aj dĺžky hrán
  - problém prvý krát zavedený v Ma et al. (2008)
  - Brejová et al. (2017)
    - poskytuje vylepšenie algoritmu z Ma et al., algoritmy pre ďalšie varianty problému



Cieľ práce

- navrhnúť algoritmy pre izometrickú rekonciliáciu
  - hrany vstupných stromov môžu mať svoje **dĺžky určené intervalom**
  - zakorenený/nezakorenený génový a druhový strom
  - nájsť všetky možné riešenia, najúspornejšie riešenie

- motivácia
  - v doterajších riešeniach bola dĺžka hrán fixná
  - skutočné dĺžky hrán nie je možné určiť presne, iba s určitou chybou
  - => doterajšie algoritmy na skutočných vstupoch často nenájdu riešenie

Výsledky

Vyjadrenie problému vo forme LP

$$
\begin{aligned}
X_v - X_u &\leq w(u,v)_{\max} && \forall (u,v) \in E(G) \\
X_v - X_u &\geq w(u,v)_{\min} && \forall (u,v) \in E(G) \\
Y_b - Y_a &\leq w(a,b)_{\max} && \forall (a,b) \in E(S) \\
Y_b - Y_a &\geq w(a,b)_{\min} && \forall (a,b) \in E(S) \\
X_u &\leq Y_{\Phi_{\mathrm{lca}}(u)} && \forall u \in V(G) \\
X_u &= Y_{\mu(u)} && \forall u \in \mathrm{leaves}(G) \\
Y_r &= 0
\end{aligned}
$$

Efektívnejšie algoritmy (1)

- zakorenený *S* s fixnými dĺžkami hrán, *G* nepresné dĺžky

- algoritmus pre zakorenený *G* s časovou zložitosťou *O(N):*
- prechod zdola nahor

$$x[u]_{\min} = \max_{v \in \text{children}(u)}(x[v]_{min} - w(u,v)_{\max})$$
$$x[u]_{\max} = \min(D(\Phi_{\text{lca}}(u)), \min_{v \in \text{children}(u)}(x[v]_{max} - w(u,v)_{\min}))$$

- prechod zhora nadol

$$X[u]_{\min} = \max(x[u]_{\min}, X[w]_{\min} + w(w,u)_{\min})$$
$$X[u]_{\max} = \min(x[u]_{\max}, X[w]_{\max} + w(w,u)_{\max})$$

- najúspornejšie riešenie

Efektívnejšie algoritmy (2)

- nezakorenený *G* ($O(N^2)$)
  - pre každú hranu z *G* vyrieši problém pre polo-zakorenený (semi-rooted) strom
  - používa predošlý *O(N)* algoritmus + lineárny program konštantnej veľkosti
  - nevie nájsť najúspornejšie riešenie

- počet duplikácií a strát spôsobených izometrickou rekonciliáciou - $O(NlogN)$
- najúspornejšia rekonciliácia pre nezakorenený *G* - $O(N^2h^2logN)$
  - opäť rieši problém pre každý polo-zakorenený strom
  - rozkladá intervaly mapovania vrcholov na podintervaly s rovnakou úspornosťou
  - vyberie najlepšiu možnú kombináciu riešení

Zhrnutie

- algoritmy pre izometrickú rekonciliáciu
  - dĺžky hrán určené intervalmi
- všeobecné riešenie - lineárny program
- najúspornejšie riešenie (strom S je zakorenený s presnými dĺžkami hrán)
  - $O(N)$ algoritmus pre zakorenený strom G
  - $O(N^2h^2logN)$ algoritmus pre nezakorenený strom G

Ďakujem za pozornosť

1. V kapitole 3.2 prípad rekonciliácie nezakoreneného génového stromu riešite v $O(n^2)$ redukovaním na n problémov, kde je strom polo-zakorenený. Zdá sa mi však, že ak by sme postupne prechádzali susedné hrany, vzniknuté inštancie sú veľmi podobné a netreba ich riešiť úplne od začiatku. Ak už mám riešenie, kde sa koreň nachádza na jednej hrane, prejsť na riešenie, kde je koreň na susednej hrane by sa malo dať v konštantnom čase, čím dostaneme lineárny algoritmus – je to pravda, alebo mi niečo uchádza?

- áno, je možné nájsť menej zložitý algoritmus
- algoritmus v nadväzujúcom článku
  - nájde všetky koreňové hrany – O(N)
  - konkrétne riešenia pre každý koreň – O(NlogN)

2. Na str. 24 tvrdíte, že dĺžka cesty medzi dvoma vrcholmi sa dá spočítať ako rozdiel ich výšok ($|path(a,b)|=h(a) - h(b) + 1$). To však nie je pravda (napr. nech *a* je koreň, *b* jeho ľavý syn je list, ale pravý podstrom je veľký; dĺžka cesty je 1, výška listu je 1, ale výška koreňa je veľká kvôli pravému podstromu).

   Pravda by to bola pre rozdiel hĺbok.

- súhlasím, chybu som si nevšimol

3. Na str. 25 a 26 a v Algoritme 4.1 potom ukazujete, ako počítať počet duplikácií a delécií pomocou rozdielu výšok (hĺbok?), ale v priloženej implementácií je použitý iný postup – prečo?

4. V texte na str. 25 píšete, že výpočet *lca* trvá O(*h*), čo je veľa – *lca* sa dá riešiť po predspracovaní v konštantnom čase. Čo je však horšie, vaša implementácia v priloženom kóde je kvadratická(!) kvôli volaniu *ArrayList.contains*, čo je lineárny prechod poľom. Túto funkciu používate v *countDL* a tým pádom aj v najvnútornejších cykloch, čo vám zhoršuje časovú zložitosť.

   - implementácia je zatiaľ prototyp – ak sa bude ďalej používať, upravím ju

   - query na *lca* sa v algoritme nakoniec nepoužíva, mohol som však spomenúť aj lepšie riešenia

5. Základom polynomiálneho algoritmu pre úspornú rekonciliáciu je rozloženie všetkých riešení na intervaly s rovnakým počtom duplikácií a delécií. V texte to spomínate, v priloženom kóde to je, ale v Algoritme 4.4 na str. 31 tento krok chýba. Navyše, namiesto spájania intervalov s rovnakou dvojicou (*D*,*L*), nebolo by lepšie spájať intervaly s rovnakým celkovým skóre, t.j. *D*+*L*?

- v algoritme pridávam intervaly do poľa $subint_u$
- dvojica (D,L) je oproti D+L informatívnejšia

18 **do** :
19 $I_{min} = max(I_{min}^{left}, I_{min}^{middle}, I_{min}^{right})$
20 $DL_u = countDL(u, \delta_I, \delta_{I^{left}}^{orig}, \delta_{I^{right}}^{orig})$
21 $DL_I = DL_u + DL_{I^{left}}^{orig} + DL_{I^{left}}^{orig}$
22 $subint_u.add(I)$
23 **if** $I_{min}^{left} == I_{min}$: $I^{left} = next(shiftedInts_v)$
24 **if** $I_{min}^{middle} == I_{min}$: $I^{middle} = next(splitInts)$
25 **if** $I_{min}^{right} == I_{min}$: $I^{right} = next(shiftedInts_w)$
26 $I_{max} = I_{min}$
27 **while** $I_{min}^{left}, I_{min}^{middle}, I_{min}^{right} \neq X[u]_{min}$

6. S tým súvisí aj odhad počtu intervalov $\text{subInt}_{max} = O(N_u + N_u h) = O(N_u h)$ na str. 29. Ak spájate iba intervaly, kde je rovnaké nie skóre, ale dvojica DL, nemalo by v odhade byť násobenie namiesto sčítania?

- ak by sme uvažovali všetky dvojice DL, odhad počtu je násobok
- pre každý podinterval (okrem najnižšieho) platí, že jeho D+L je väčšie ako D+L predošlého podintervalu
- => dokopy maximálny počet D+L=$O(N_u h)$ podintervalov

7. Na druhej strane, nie je odhad na počet delécií $O(N_u h)$ príliš pesimistický? Ak vezmeme odhad zo str. 25, že počet delécií <= $|path_S(B_u, B_v)| + |path_S(B_u, B_w)| - 2$, celkovo sa to v celom podstrome nasčíta na $O(N_u)$ – v tom prípade by bol počet intervalov $subInt_{max} = O(N_u)$. V prípade, že sa mýlim, viete ukázať príklad vstupu, na ktorom je počet intervalov veľký? Prípadne príklad vstupu, kde výpočet funkcie *countSubintervalDL* trvá dlho?

- *G* s tromi vrcholmi má koreň *u*, ktorý sa môže mapovať do celej hĺbky S => *u* ma O(*h)* intervalov

8. Dajú sa algoritmy rozšíriť aj na stromy, ktoré nie sú binárne, prípadne majú hrany nulovej dĺžky?

- okrem posledného (4.4) vedia algoritmy pracovať s nebinárnymi stromami
- algoritmus 4.4 pre najúspornejšiu rekonciliáciu možno rozšíriť, ale bude zložitejší o faktor *d* (max. počet detí)

- nulové dĺžky hrán -> zložitejšie algoritmy