Univerzita Komenského, Bratislava
Fakulta Matematiky, Fyziky a Informatiky
Katedra Informatiky

Vzdialenosti na stromoch

(Bakalárska práca)

Matúš Fedák

Študijný odbor: Informatika 9.2.1
Vedúci: Prof. RNDr. Branislav Rovan, PhD.

Bratislava, 2009

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

Poďakovanie

Chcel by som poďakovať vedúcemu bakalárskej práce, Prof. RNDr. Branislavovi Rovanovi, PhD. za pomoc pri výbere témy práce a za cenné rady pri písaní práce.
Tiež by som chcel poďakovať svojej rodine za podporu, obzvlášť svojej mame.
Nakoniec by som sa chcel poďakovať svojej priateľke za jej trpezlivosť a kritické pripomienky ku práci.

Abstrakt

Autor:	Matúš Fedák
Názov bakalárskej práce:	Vzdialenosti na stromoch
Škola:	Univerzita Komenského v Bratislave
Fakulta:	Fakulta matematiky, fyziky a informatiky
Katedra:	Katedra informatiky
Vedúci bakalárskej práce:	Prof. RNDr. Branislav Rovan, PhD.
Rozsah práce:	34 strán
Bratislava, jún 2009	

Práca predstavuje algoritmy na výpočet vzdialenosti stromov. Venuje sa najmä editačnej vzdialenosti stromov a obmedzenej editačnej vzdialenosti stromov. Prezentuje doteraz známe algoritmy na výpočet editačnej vzdialenosti usporiadaných stromov a ukazuje dôkaz NP úplnosti problému výpočtu editačnej vzdialenosti neusporiadaných stromov. Pre obmedzenú editačnú vzdialenosť popisuje algoritmy na výpočet vzdialenosti usporiadaných aj neusporiadaných stromov. Práca vyzdvihuje hlavné myšlienky jednotlivých algoritmov, zjednocuje ich prezentáciu a ilustruje ich na príkladoch. Okrem prezentácie algoritmov sa práca venuje rozboru efektívnosti (časovej zložitosti) jednotlivých algoritmov.

Kľúčové slová: Editačná vzdialenosť stromov, Obmedzená editačná vzdialenosť stromov.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Prvýkrát sa problémom vzdialenosti dvoch stromov môžeme stretnúť v práci autora Tai [Tai79]. Jeho vzdialenosť bola zovšeobecnením vtedy dobre známej vzdialenosti na slovách. Dodnes je táto vzdialenosť založená na zmene stromu pomocou operácií premenovania, pridania a zmazania vrcholu jednou z najpoužívanejších vzdialeností.

V dnešnej dobe je problém vzdialenosti stromov stále skúmanou oblasťou. Je to najmä vďaka využitiu v počítačovej biológii (vzdialenosť sekundárnych štruktúr RNA[BAS90]), porovnávaní štrukturovaných textov (XML databázy [ZCZ03]), analýze obrázkov [Shi91], alebo optimalizácii kompilátorov.

Okrem vzdialenosti stromov vzniklo viac problémov, ktoré súvisia s touto problematikou. V literatúre sa preto môžeme stretnúť s problémami ako inklúzia stromov (Tree inclusion), zarovnanie stromov (Tree alignment distance), najväčší spoločný podstrom (Largest common subtree) a najmenší spoločný nadstrom (Smallest common supertree).

Táto práca sa zaoberá len problémom výpočtu vzdialenosti dvoch stromov. Predstavuje dve definície vzdialenosti (editačnú a obmedzenú editačnú vzdialenosť) a algoritmy na výpočet týchto vzdialeností.

V práci prezentujeme jednotným spôsobom algoritmy, ktoré boli v literatúre prezentované rôznorodo. Je preto ľahšie vidieť v čom sa líšia a v čom spočíva ich zložitosť.

Prvá kapitola predstavuje problém vzdialenosti dvoch stromov a definuje editačnú vzdialenosť [Tai79]. V druhej kapitole predstavujeme štyri algoritmy (jednoduchý algoritmus [Bil05], algoritmus Zhang and Shasha [ZS89], Kleinov algoritmus [Kle98] a Demainov algoritmus [DMRW07]) na výpočet editačnej vzdialenosti usporiadaných stromov. Na záver druhej kapitoly ukážme dôkaz REF, že pre neusporiadané stromy je problém výpočtu ich editačnej vzdialenosti NP úplný. V tretej kapitole zadefinujeme obmedzenú editačnú vzdialenosť [Zha95] a predstavíme algoritmus na výpočet vzdialenosti dvoch usporiadaných

[Zha95] alebo neusporiadaných stromov[Zha96].

1.1 Stromy

V práci sa budeme zaoberať vzdialenosťou stromov. Predpokladáme, že čitateľ má aspoň základné znalosti z teórie grafov a pozná pojmy ako strom, les, predchodca, nasledovník alebo prehľadávanie stromu.

Pracovať budeme s usporiadanými (majú dané usporiadanie medzi súrodencami), ale aj neusporiadanými stromami(súrodencov stromu môžeme ľubovoľne prehadzovať). Základné značenie bude pre stromy T_1, T_2 a pre lesy F_1, F_2. Podstrom stromu T_1, ktorého koreň je vrchol $v \in T_1$, budeme označovať $T_1(v)$. Les, ktorý vznikne odstránením vrcholu v zo stromu $T_1(v)$, budeme označovať $F_1(v)$.

Pre usporiadané stromy je vrchol v naľavo od vrcholu w, ak pri postorder a preorder prechode stromom narazíme na vrchol v skôr ako na vrchol w. Ak je vrchol v naľavo od vrcholu w, potom vrchol w je napravo od vrcholu v.

Budeme sa zaoberať stromami, ktorých vrcholy majú priradené návestia z konečnej abecedy Σ.

1.2 Vzdialenosť

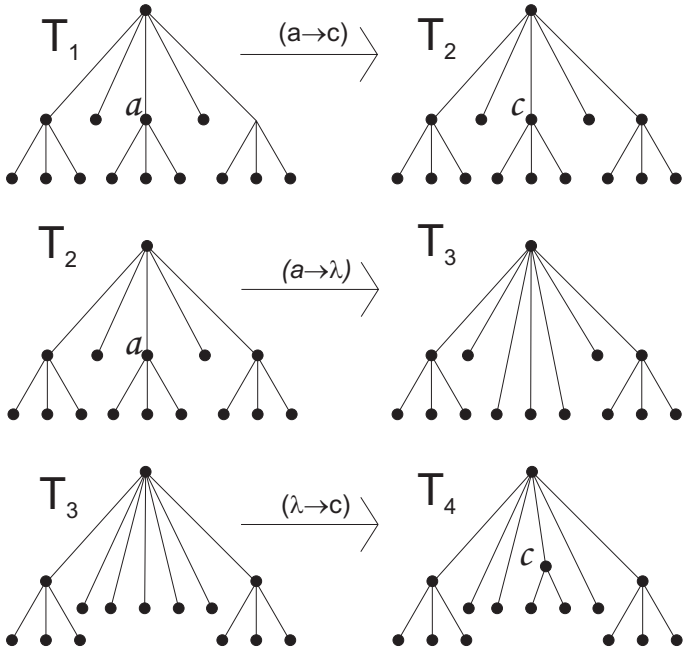
Predtým ako začneme hovoriť o vzdialenosti, najprv ju zadefinujeme:

Definícia 1.2.1 *Nech M je ľubovoľná množina a $\mathbb{R}$ množina reálnych čísel, nech $u, v, w \in M$. Nech $d : M \times M \to \mathbb{R} \cup \{\infty\}$ je funkcia spĺňajúca nasledovné podmienky:*

1. $d(u, v) \geq 0$
2. $d(u, v) = 0 \Leftrightarrow u = v$ *(reflexívnosť)*
3. $d(u, v) = d(v, u)$ *(symetria)*
4. $d(u, v) + d(v, w) \geq d(u, w)$ *(trojuholníková nerovnosť)*

Potom funkciu d nazývame vzdialenosťou na množine M.

Teraz definujeme základné editačné operácie s vrcholmi stromu. Vďaka týmto operáciam budeme môcť meniť štruktúru stromu a návestia vrcholov.



Obr. 1.1: Príklad editačných operácií slúžiacich na zmenu stromu.

- *Operáciou premenovania* zmeníme návestie vrcholu v.
- *Operáciou zmazania* odstránime zo stromu vrchol v a všetkých jeho synov dáme jeho rodičovi w.
- *Operácia vloženia* bude inverzná k operácii zmazania, teda vložíme vrchol v ako rodiča súvislej podpostupnosti synov vrcholu w. Teda po tejto operácii stratí w niektorých zo svojich synov, avšak získa syna v.

Aby sme si zjednodušili zápis týchto operácií, zavedieme symbol imaginárneho vrcholu λ s prázdným návestím. Abecedu Σ rozšírenú o prázdny symbol λ budeme označovať Σ_λ. Každú operáciu zapíšeme pomocou stavu pred a po jej vykonaní. Nech teda l_1, l_2 sú návestia vrcholov $l_1, l_2 \in \Sigma$. Zmenu návestia vo vrchole v zapíšeme ako $(l_1 \to l_2)$. Operáciou zmazania odstránime vrchol, a pri zápise si pomôžeme imaginárnym vrcholom $(l_1 \to \lambda)$. Pridanie vrcholu, naopak, z ničoho vytvorí nový vrchol a túto operáciu budeme označovať $(\lambda \to l_1)$. Toto označenie bude používať pre vrcholy rovnako ako pre návestia teda zápisom $(v \to w)$ budeme rozumieť zmenu návestia vrcholu v na návestie vrcholu w. Zmazanie vrcholu v zapíšeme $(v \to \lambda)$, pridanie vrcholu $(\lambda \to v)$.

Teraz už môžeme definovať vzdialenosť dvoch stromov ako počet zmien, ktoré musíme vykonať, aby sme zo stromu T_1 dostali strom T_2 za použitia vyššie definovaných editačných operácií.

Definícia 1.2.2 *Jednotková vzdialenosť dvoch stromov $\delta(T_1, T_2)$ je najmenší počet editačných operácií, ktoré musíme vykonať, aby sme zo stromu T_1 dostali strom T_2.*

Dôkaz. Teraz musíme ukázať, že táto vzdialenosť spĺňa všetky podmienky vzdialenosti. Počet operácií je vždy kladný, preto prvú podmienku spĺňa. Reflexívnosť platí, lebo na zmenu stromu T_1 na strom T_1 nepotrebujeme žiadnu operáciu. Naopak, ak nepotrebujeme žiadnu operáciu na zmenu stromu T_1 na T_2, tak musia byť tieto stromy totožné. Symetriu dostávame ako dôsledok symetrie operácií[1]. Trojuholníkovú nerovnosť si môžeme predstaviť ako proces transformácie stromu T_1 na strom T_3, pričom raz chceme dostať ako medzikrok T_2. Je zrejmé, že pri transformácii s medzikrokom musíme určite použiť viac operácií ako pri transformácii bez medzikroku.

Takto definovaná vzdialenosť priradzuje každej operácii rovnakú váhu. V niektorých prípadoch môže byť výhodné ohodnotiť jednotlivé operácie rôzne. Funkciu, ktorá bude ohodnocovať jednotlivé editačné operácie, nazveme cenovou funkciou γ. V tejto práci sa budeme zaoberať iba cenovými funkciami, ktoré spĺňajú podmienky vzdialenosti.

Definícia 1.2.3 *Funkciu $\gamma : (\Sigma_\lambda \times \Sigma_\lambda)\backslash(\lambda, \lambda) \to \mathbb{R}$, ktorá spĺňa nasledujúce podmienky, nazveme cenovou funkciou. Nech $l_1, l_2 \in \Sigma_\lambda$:*

1. $0 \leq \gamma(l_1, l_2) \leq 1, \gamma(l_1, l_1) = 0$
2. $\gamma(l_1, l_2) = \gamma(l_2, l_1)$
3. $\gamma(l_1, l_3) \leq \gamma(l_1, l_2) + \gamma(l_2, l_3)$

Príklad: Majme symboly $\Sigma = \{a, b, c\}$

$$\gamma(a \to \lambda) = 1, \gamma(a \to b) = 0.5, \gamma(a \to c) = 0.5$$

$$\gamma(b \to \lambda) = 1, \gamma(b \to a) = 0.5, \gamma(b \to c) = 0.5$$

$$\gamma(c \to \lambda) = 1, \gamma(c \to a) = 0.5, \gamma(c \to b) = 0.5$$

Pri takto definovanej cenovej funkcii, bude mať operácia zmazania alebo pridania vrcholu dvojnásobnú cenu oproti operácii premenovania.

[1] Premenovanie je symetrické samo so sebou. Zmazanie a pridanie vrcholu sú navzájom symetrické.

Teraz už môžeme pozmeniť jednotkovú vzdialenosť pre usporiadané stromy, a to tak, že namiesto počtu operácií vezmeme sumu ich cien z cenovej funkcie.

Definícia 1.2.4 *Nech $S = s_1, \ldots, s_n$ je postupnosť operácií transformujúcich T_1 na T_2. Pre vzdialenosť dvoch stromov platí:*

$$\delta(T_1, T_2) = \min\{\sum_{s_i \in S} \gamma(s_i) | S \text{ transformuje } T_1 \text{ na } T_2\}$$

1.3 Editačné zobrazenie

V predchádzajúcej časti sme definovali vzdialenosť dvoch stromov ako sumu cien operácií, transformujúcich jeden strom na druhý. Ako môžeme vidieť na obrázku 1.2, poradie týchto operácií nemusí byť jednoznačné. Výslednú cenu poradie operácií neovplyvňuje, preto tento pohľad na vzdialenosť nám nevyhovuje. V tejto časti teda ukážeme, ako sa dá na editačnú vzdialenosť pozerať z iného uhla.

Vezmime si ľubovoľnú minimálnu transformáciu stromu T_1 na strom T_2. Počas nej vrcholy mažeme, pridávame alebo meníme ich návestia. Ak naša transformácia je minimálna, potom vieme povedať, že vrcholy, ktoré zmažeme sa musia[2] nachádzať v strome T_1. Podobne vrcholy, ktoré sme počas transformácie pridali, musíme nájsť v strome T_2.

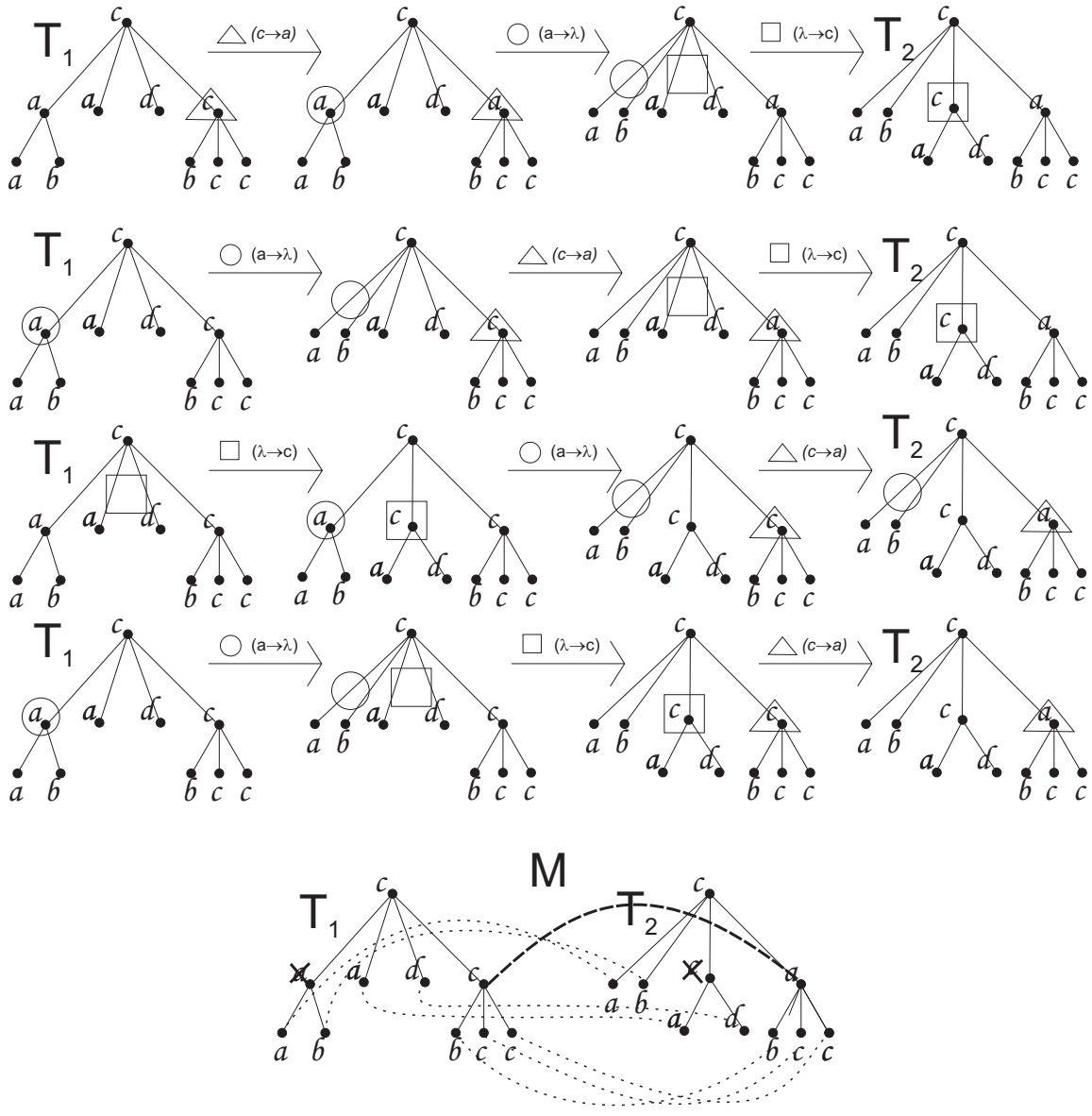
To, čo nám ostáva, sú vrcholy, ktoré počas transformácie nezmeníme alebo zmeníme ich návestie, a teda sa nachádzajú v oboch stromoch. Preto medzi nimi môžeme vytvoriť páry. Vrchol zo stromu T_1 po zmene návestia tvorí pár s daným vrcholom stromu T_2 . Rovnako vrchol stromu T_1, na ktorý neaplikujeme žiadnu operáciu[3], automaticky tvorí pár s príslušným vrcholom stromu T_2. Takúto množinu párov vrcholov stromov T_1, T_2 budeme nazývať editačné zobrazenie.

Vrcholy stromu T_1, ktoré nemajú pár v editačnom zobrazení, musíme počas transformácie zmazať. Vrcholy stromu T_2 bez páru v editačom zobrazení musíme počas transformácie pridať. A vrcholom v pároch pri transformácii zmeníme návestie.

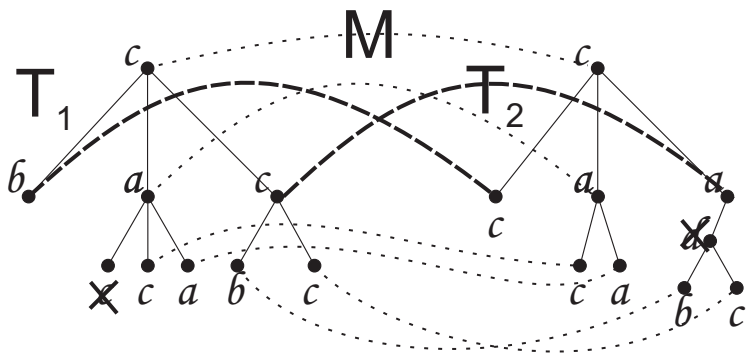
V predchádzajúcich odsekoch sme ukázali, ako z transformácie vytvoriť editačné zobrazenie. Teraz definujeme podmienky, ktoré nám zaručia, že množina dvojíc vrcholov bude editačným zobrazením. Teda bude existovať transformácia, z ktorej vieme vytvoriť toto editačné zobrazenie.

[2] Ak by naša transformácia nebola minimálna, tak sa môže stať, že zmažeme vrchol, ktorý sme do stromu T_1 pridali. Tým však spravíme dve zbytočné operácie, ktoré sa v minimálnej transformácii nachádzať nemôžu.

[3] Keďže zmena návestia vrcholu na to isté návestie má nulovú cenu, môžeme túto operáciu aplikovať na vrchol, s ktorým sme nič nerobili, čím na každý vrchol aplikujeme nejakú operáciu.



Obr. 1.2: Transformácie stromu T_1 na strom T_2.



Obr. 1.3: Príklad editačného zobrazenia.

Definícia 1.3.1 *Editačné zobrazenie je usporiadaná trojica* (M, T_1, T_2)*, kde* $M \subseteq T_1 \times T_2$ *a pre všetky páry* $(v_1, w_1), (v_2, w_2) \in M$ *platí:*

1. $v_1 = v_2$ *práve vtedy, keď* $w_1 = w_2$ *(jedno-jednoznačnosť)*
2. v_1 *je predchodca* v_2 *práve vtedy, keď* w_1 *je predchodca* w_2 *(zachovanie predchodcov)*
3. v_1 *je naľavo od* v_2 *práve vtedy, keď* w_1 *je naľavo* w_2 *(zachovanie usporiadania synov)*

Prvá podmienka nám zabezpečí, že každý vrchol stromu T_1 môže byť v páre s maximálne jedným vrcholom stromu T_2. Druhá bude slúžiť na to, aby sme v editačnom zobrazení neprehadzovali poradie vrcholov od koreňa k listom, keďže operáciami zmazania, pridania a zmeny návestia toto nedosiahneme. Posledná podmienka zaručuje, že sa nám nezmení usporiadanie synov ľubovoľného vrcholu. Táto tretia podmienka má význam vtedy, ak chceme počítať vzdialenosť usporiadaných stromov, keďže tie majú pevne dané usporiadanie medzi synmi. Pre neusporiadané stromy môžeme meniť usporiadanie synov, a teda túto podmienku musíme z editačného zobrazenia vynechať.

Pre jednoduchosť budeme editačné zobrazenie označovať len symbolom M namiesto (M, T_1, T_2), ak je jasné, medzi akými stromami je toto zobrazenie definované.

Definícia 1.3.2 *Cena editačného zobrazenia* $\gamma(M)$ *je:*

$$\gamma(M) = \sum_{(v,w)\in M} \gamma(v, w) + \sum_{v\in D} \gamma(v, \lambda) + \sum_{w\in I} \gamma(\lambda, w),$$

kde D *je množina vrcholov stromu* T_1*, ktoré nemajú pár v editačnom zobrazení* M*. Množinu* I *tvoria vrcholy stromu* T_2 *bez páru v editačnom zobrazení.*

Jednotlivé editačné zobrazenia môžeme spolu skladať . Nech M_1 je editačné zobrazenie medzi stromami T_1 a T_2. M_2 je editačné zobrazenie medzi stromami T_2 a T_3. Potom $M_3 = M_1 \circ M_2$ bude editačné zobrazenie medzi stromami T_1 a T_3 definované takto:

$$M_1 \circ M_2 = \{(v, w) | \exists u \in T_2; (v, u) \in M_1 \ a(u, w) \in M_2\}$$

.

Teraz už môžeme definovať vzdialenosť pomocou editačného zobrazenia.

Definícia 1.3.3 *Pre vzdialenosť dvoch stromov platí:*

$$\delta(T_1, T_2) = \min\{\gamma(M) | (M, T_1, T_2) \textit{ je editačné zobrazenie}\}$$

Je ľahké ukázať, že takto definovaná vzdialenosť je ekvivalentná s Definíciou 1.2.4. Preto budeme pri výpočte vzdialenosti dvoch stromov hľadať minimálne editačné zobrazenie namiesto minimálnej transformácie.

Kapitola 2

Editačná vzdialenosť stromov

V tejto kapitole predstavíme algoritmy na výpočet editačnej vzdialenosti stromov. V prvej podkapitole ukážeme lemu, ktorá bude kľúčovým prvkom jednoduchého algoritmu [Bil05] na výpočet vzdialenosti usporiadaných stromov. V ďalšej podkapitole zovšeobecníme jednoduchý algorimus, čím definujeme triedu algoritmov „stratégie pokrytia“. Takéto zovšeobecnenie nájdeme prvýkrát v práci [DTI03]. V nasledujúcich kapitolách predstavíme tri algoritmy z triedy stratégie pokrytia, ktoré počítajú vzdialenosť usporiadaných stromov. Posledný predstavený Demainov algortimus [DMRW07] dosahuje pre triedu algoritmov stratégie pokrytia optimálnu časovú zložitosť. Na záver kapitoly ukážeme, že pre neusporiadané stromy je problém výpočtu editačnej vzdialenosti NP úplný [ZWS95].

2.1 Jednoduchý algoritmus

V tejto časti ukážeme rekurziu, ktorá nám umožní napísať jednoduchý algoritmus na výpočet vzdialenosti usporiadaných stromov. Tento algoritmus možno nájsť v [ZS89], avšak my ukážeme jeho formu z [Bil05].

Z prvej kapitoly vieme, že na výpočet editačnej vzdialenosti dvoch usporiadaných stromov nám stačí nájsť minimálne editačné zobrazenie. Pri jeho hľadaní budeme postupovať tak, že si vyberieme jeden vrchol z každého stromu a rozoberieme všetky možnosti, ktoré môžu nastať pri ľubovoľnom editačnom zobrazení. Z týchto možností si vyberieme vždy tú s najnižšou cenou, čo nám zaručí nájdenie minimálneho editačného zobrazenia.

Na výpočet vzdialenosti stromu od prázdneho stromu vieme, že jeho koreň sa v editačnom zobrazení nenachádza. Preto ho môžeme zmazať a počítať vzdialenosť rekurzívne bez neho. Pre dva neprázdne stromy si za vrcholy, ktoré budeme rozoberať, vyberieme korene. Tie musia v editačnom zobrazení tvoriť pár, alebo jeden z nich sa v editačnom

zobrazení nebude nachádzať. Možnosť, že obidva majú iný pár v editačnom zobrazení nemôže nastať, pretože takéto zobrazenie by nespĺňalo podmienku zachovanie predchodcov z definície editačného zobrazenia. Ak teda vieme, že korene tvoria spolu pár, stačí nám vypočítať vzdialenosť bez týchto vrcholov a pripočítať cenu za zmenu jedného na druhý. A ak sa v editačnom zobrazení jeden z vrcholov nenachádza, vypočítame vzdialenosť bez neho a pripočítame cenu za jeho zmazanie.

Lema 2.1.1 *Nech T_1 a T_2 sú usporiadané stromy a γ je cenová funkcia definovaná na symboloch. Nech v, w sú korene stromov T_1, T_2 a symbol θ predstavuje prázdny podstrom, potom platí:*

$$\begin{aligned}
\delta(\theta,\theta) &= 0 \\
\delta(T_1,\theta) &= \delta(T_1 - v,\theta) + \gamma(v \to \lambda) \\
\delta(\theta,T_2) &= \delta(\theta,T_2 - w) + \gamma(\lambda \to w) \\
\delta(T_1,T_2) &= \min \begin{cases} \delta(T_1 - v,T_2) + \gamma(v \to \lambda) \\ \delta(T_1,T_2 - w) + \gamma(\lambda \to w) \\ \delta(T_1 - v,T_2 - w) + \gamma(v \to w) \end{cases}
\end{aligned}$$

Zápis $\delta(T_1 - v, T_2 - w)$ môžme nahradiť zápisom $\delta(F_1(v), F_2(w))$. Pri predchádzajúcej Leme 2.1.1 sa nám môže stať, že po zmazaní niektorého vrcholu musíme vypočítať vzdialenosť dvoch lesov.

Pri výpočte vzdialenosti lesov F_1, F_2 budeme postupovať rovnako ako pri stromoch, avšak výber vrcholov, s ktorými budeme pracovať, bude zložitejší. Najvýhodnejšie pre nás budú korene v, w najľavejších, prípadne najpravejších stromov týchto lesov T_1, T_2. Výhodné sú vďaka tomu, že ak tvoria páry v editačnom zobrazení, tak nemôžu vytvoriť dva nezávislé páry, ale tvoria pár spolu, teda $(v, w) \in M$.

Lema 2.1.2 *Nech F_1 a F_2 sú usporiadané lesy a γ je cenová funkcia definovaná na symboloch. Nech v a w sú korene najľavejších (najpravejších) stromov $T_1(v), T_2(w)$ v lesoch F_1 a F_2, potom platí:*

$$\begin{aligned}
\delta(\theta,\theta) &= 0 \\
\delta(F_1,\theta) &= \delta(F_1 - v,\theta) + \gamma(v \to \lambda) \\
\delta(\theta,F_2) &= \delta(\theta,F_2 - w) + \gamma(\lambda \to w) \\
\delta(F_1,F_2) &= \min \begin{cases} \delta(F_1 - v,F_2) + \gamma(v \to \lambda) \\ \delta(F_1,F_2 - w) + \gamma(\lambda \to w) \\ \delta(F_1(v),F_2(w)) + \delta(F_1 - T_1(v),F_2 - T_2(w)) + \gamma(v \to w) \end{cases}
\end{aligned}$$

Dôkaz. Rozoberieme prípad s výberom najľavejších koreňov, keďže pre najpravejšie korene je situácia symetrická. Možnosti, ako môže vyzerať editačné zobrazenie pre korene v, w najľavejších stromov $T_1(v), T_2(w)$ lesov F_1, F_2 sú tri.

(a) Vrchol v nemá pár v editačnom zobrazení. Tento vrchol zmažeme a vypočítame vzdialenosť bez neho.

(b) Vrchol w nemá pár v editačnom zobrazení. Táto situácia je symetrická s prvým prípadom.

(c) Obidva vrcholy v, w majú pár v editačnom zobrazení. Sporom ukážeme, že v tomto prípade tvoria pár spolu. Predpokladajme, že majú obidva iný pár v editačnom zobrazení, teda existujú vrcholy $v' \in F_1$, $w' \in F_2$, pričom $(v, w'), (v', w) \in M$. Keďže v je najľavejší koreň, musí byť v' jeho potomok, alebo napravo od neho. Z definície editačného zobrazenia dostávame, že aj w je potomok w', alebo musí byť napravo od w'. V oboch prípadoch sa dostávame do sporu s výberom w ako najľavejšieho koreňa lesa F_2.

   Ak teda platí $(v, w) \in M$ vrcholy stromu $T_1(v)$ môžu tvoriť pár jedine s vrcholmi stromu $T_2(w)$ kvôli podmienke zachovania predchodcov. Vrcholy lesa $F_1 - T_1(v)$ môžu tvoriť páry s vrcholmi lesa $F_2 - T_2(w)$, vďaka podmienke zachovania usporiadania súrodencov. To rozdeľuje náš problém vzdialenosti lesov F_1, F_2 na dva menšie problémy $\delta(F_1(v), F_2(w))$ a $\delta(F_1 - T_1(v), F_2 - T_2(v))$.

Teraz už máme dostatok informácií, aby sme zostrojili jednoduchý algoritmus na vypočítanie editačnej vzdialenosti dvoch usporiadaných stromov. Priamym aplikovaním Lemy 2.1.2 dostaneme rekurzívny algoritmus. Pri tomto algoritme budeme počítame každý podproblém len raz a jeho výsledok si zapamätáme, čím zlepšíme časovú zložitosť.

Lema 2.1.3 *Časová zložitosť jedoduchého algoritmu je* $O(|T_1|^2|T_2|^2)$.

Dôkaz. V každom kroku sa náš algoritmus rekurzívne zavolá na menšie podproblémy. Následne tieto menšie podprobémy spracuje, čo predstavuje niekoľko súčtov a výber minima. Keďže spracovanie jedného podproblému trvá konštantný čas, môžeme zložitosť algoritmu ohraničiť počtom podproblémov, na ktoré počas rekurzie narazíme.

Najprv si definujeme podles $F^{(i,j)}, 0 < i + j < |F|$ ako les, ktorý získame z F postupným zmazávaním najľavejších a najpravejších koreňov i a j krát. Pre analógiu so slovami nazveme $F^{(0,j)}$ prefix lesa F a $F^{(i,0)}$ suffix lesa F.

Teraz ukážeme, že všetky podproblémy, na ktoré počas rekurzie narazíme, budú tvaru $\delta(T_1^{(i,j)}, T_2^{(k,l)})$. Počas rekurzie môžu nastať tri prípady:

(a) Zmažeme najľavejší koreň prvého stromu, teda zväčšíme i.

(b) Zmažeme najľavejší koreň druhého stromu, teda zväčšíme k.

(c) Spárujeme najľavejšie korene oboch stromov a musíme vypočítať dva menšie podproblémy. Prvým je vzdialenosť najľavejších podstromov, ktorú dostaneme zväčšením j a l tak, aby nám ostali len najľavejšie podstromy. Druhým problémom je vzdialenosť lesov bez najľavejších podstromov. Vtedy nám stačí zväčšiť i a k o veľkosti najľavejších podstromov.

Pre najpravejšie korene je situácia symetrická.

Tým sme teda ukázali, že všetky podproblémy, na ktoré môžeme naraziť, sú tvaru $\delta(T_1^{(i,j)}, T_2^{(k,l)})$. Keďže i, j vyberáme z inervalu $\langle 0, |T_1| \rangle$ a k, l z intervalu $\langle 0, |T_2| \rangle$, počet podproblémov je ohraničený číslom $|T_1|^2|T_2|^2$. Týmto zároveň dostávame výslednú zložitosť, keďže spracovanie jedného podproblému trvá konštantný čas.

2.2 Stratégia pokrytia

V predošlej podkapitole sme dokázali Lemu 2.1.2, pomocou ktorej môžeme vypočítať vzdialenosť dvoch usporiadaných stromov. Pri tejto leme sme si vybrali jeden vrchol z každého stromu a rozobrali sme všetky možnosti, ako môže vyzerať editačné zobrazenie. Pre dva stromy boli tieto vrcholy korene. Pri výpočte vzdialenosti lesov sme si museli zvoliť, či budeme pracovať s najľavejšími alebo najpravejšími koreňmi. Jednoduchý algoritmus predstavený v predošlej časti výber koreňov neobmedzoval. Postup, akým si vyberáme, s ktorými koreňmi budeme pracovať, má vplyv na počet podproblémov, ktoré budeme musieť vypočítať. Preto v tejto časti definujeme triedu algoritmov, ktoré budú popisovať práve tento výber s ktorými vrcholmi máme pracovať.

Ak chceme zostrojiť algoritmus na výpočet vzdialenosti usporiadaných stromov za použitia Lemy 2.1.2, musíme sa vedieť pri každom rekurzívnom volaní rozhodnúť, či použijeme verziu lemy s pravými alebo ľavými koreňmi. Tento výber nazveme „stratégiou pokrytia“.

Definícia 2.2.1 *Nech T_1, T_2 sú stromy. Stratégiou pokrytia nazveme zobrazenie z dvojice (F_1', F_2') podlesov stromov T_1, T_2 do množiny {vľavo,vpravo}.*

Prvýkrát sa s takýmto prístupom k algoritmom na výpočet vzdialenosti dvoch stromov môžeme stretnúť v práci autorov Dulucqata and Touzetu [DTI03], kde môžeme nájsť dolný odhad časovej zlozitosti $\Omega(|T_1||T_2| \log |T_1| \log |T_2|)$ tejto triedy algoritmov.

V nasledujúcich podkapitolách podrobne rozoberieme tri stratégie pokrytia, spolu s rozborom ich časovej zložitosti.

2.3 Algoritmus Zhang and Shasha

Algoritmus autorov Zhang a Shasha [ZS89] je prvým z algoritmov používajúcich stratégiu pokrytia, ktorým sa budeme v tejto práci venovať. Pre zopakovanie, pri jednoduchom algoritme sme neovplyvňovali, ktorý variant Lemy 2.1.2 použijeme. Ani pri tomto algoritme nebude výber variantu lemy náročný. Na začiatku si zvolíme, či budeme pracovať s ľavými, alebo pravými koreňmi, a počas výpočtu tento výber nebudeme meniť. Fungovanie algoritmu nebudeme ďalej popisovať, keďže je totožné s jednoduchým algoritmom a zameriame sa len na rozbor jeho časovej zložitosti. Tento rozbor časovej zložitosti spravíme pre prípad, že stratégia bude vyberať vždy najpravejšie korene.

Označenie 2.3.1 *Vrchol v stromu T nazveme významným, ak je koreňom stromu T, alebo má súrodenca, ktorý je od neho naľavo.*

Na základe významných vrcholov teraz definujeme dosiahnuteľné podlesy. O nich ukážeme, že počas výpočtu nám stačí spočítať vzdialenosti všetkých dvojíc dosiahnuteľných podlesov.

Označenie 2.3.2 *Nech v je významný vrchol stromu T. Potom všetky prefixy*[1] *lesa $F(v)$ nazveme dosiahnuteľnými.*

Lema 2.3.3 *Pre každý vrchol $v \in T$, je $F(v)$ dosiahnuteľný podles.*

Dôkaz. Ak je vrchol v významný, $F(v)$ je dosiahnuteľný z definície. Zaujímavejšie je, ak v nie je významný, teda je najľavejším synom svojho rodiča. Na ceste od vrchola v ku koreňu nájdeme prvý vrchol w, ktorý je významný. Všetky prefixy lesa $F(w)$ sú podľa definície dosiahnuteľné. Postupným zmazávaním pravých koreňov z lesa $F(w)$ dostaneme les F(v). Preto $F(v)$ je prefix lesa $F(w)$, teda je dosiahnuteľný.

Teraz ukážeme, že na výpočet vzdialenosti $\delta(T_1, T_2)$ nám stačí poznať vzdialenosti dosiahnuteľných podlesov T_1, T_2. Nech S_1, S_2 sú dosiahnuteľné podlesy a vrcholy v, w sú korene najpravejších stromov podlesov S_1, S_2. Podľa Lemy 2.1.2 tieto korene spárujeme, alebo

[1] Podľa definície prefix lesa dostaneme niekoľkonásobným zmazávaním najpravejších koreňov.

jeden z nich v editačnom zobrazení nebude. Ak sa v editačnom zobrazení jeden nenachádza, zmažeme ho, čím vytvoríme prefix lesa $S_1(S_2)$. Keďže $S_1(S_2)$ je dosiahnuteľný, aj jeho prefixy sú dosiahnuteľné. Druhá možnosť je, že vrcholy v, w spárujeme. Potom musíme vypočítať dve vzdialenosti, $\delta(S_1 - T_1(v), S_2 - T_2(w))$ a $\delta(F_1(v), F_2(w))$. Odstránením najpravejšieho stromu dostaneme znova prefix $S_1(S_2)$, teda dosiahnuteľný podles a lesy $F_1(v))$ a $F_2(w)$ sú dosiahnuteľné vďaka Leme 2.3.3.

Definícia 2.3.4 *Redukovanou hĺbkou vrcholu v vzhľadom na významné vrcholy nazveme počet významných vrcholov na ceste od vrcholu v ku koreňu. Označovať ju budeme* $\mathrm{rh}(v)$. *Redukovaná hĺbka stromu bude maximum z hĺbok jeho vrcholov* $\mathrm{rh}(T) = \max\{\mathrm{rh}(v) | v \in T\}$.

Lema 2.3.5 *[ZS89] Pre strom T platí* $\mathrm{rh}(T) \leq \min\{D, L\}$, *kde D predstavuje hĺbku stromu a L počet listov v strome T.*

Teraz sa už môžeme pustiť do ohraničenia počtu dosiahnuteľných podlesov stromu T.

$$\sum_{v \text{ je významný}} |F(v)| \quad < \quad \sum_{v \text{ je významný}} |T(v)| \quad = \sum_{v \in T} \mathrm{rh}(v) \leq \sum_{v \in T} \mathrm{rh}(T) \leq \sum_{v \in T} \min\{D, L\}$$

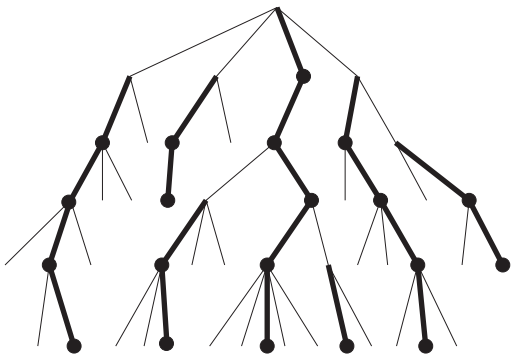
Začneme sumovať cez významné vrcholy a počet ich prefixov. Najdôležitejším krokom pre výpočet sumy bude zámena sumácie. Sumovanie cez významné vrcholy a súčet ich potomkov zmeníme na sumu cez všetky vrcholy s počtom ich významných predkov. Ten ohraničíme konšantou pre celý strom a použijeme Lemu 2.3.5.

Veta 2.3.6 *[ZS89] Problém výpočtu editačnej vzdialenosti dvoch usporiadaných stromov T_1 a T_2 sa dá vyriešiť v čase $O(|T_1||T_2| \min\{D_1, L_1\} \min\{D_2, L_2\})$ a pamäti $O(|T_1||T_2|)$, kde D_1, D_2 sú hĺbky a L_1, L_2 počty listov stromov T_1, T_2 .*

Pamäťová zložitosť pri priamej rekurzívnej implementácii bude rovnaká ako počet podproblémov teda $O(|T_1||T_2| \min\{D_1, L_1\} \min\{D_2, L_2\}$. S použitím dynamického programovania(výpočet vzdialenosti od menších podproblémov k väčším) a postupným zahadzovaním nepotrebných výsledkov je možné získať pamäťovú zložitosť $O(|T_1||T_2|)$.

2.4 Kleinov algoritmus

V tejto časti predstavíme Kleinov algoritmus [Kle98], v poradí druhý, ktorý používa stratégiu pokrytia. Predošlý algoritmus Zhang a Shasha dosahuje v najhoršom prípade zložitosť $O(N^4)$ pre stromy s lineárnou hĺbkou a počtom listov. Kleinov algoritmus zlepšuje



Obr. 2.1: Príklad rozdelenia vrcholov a hrán stromu na ľahké a ťažké.

túto hornú hranicu na $O(N^3 \log N)$. Základom Kleinovho algoritmu je rozdelenie vrcholov stromu na ťažké a ľahké. S takýmto rozdelením vrcholov sa môžeme stretnúť prvýkrát v práci Tarjana [HT84].

Definícia 2.4.1 *Koreň stromu nazveme ľahkým. Pre každý vrchol stromu vyberieme spomedzi jeho synov toho, ktorý má pod sebou najviac vrcholov a nazveme ho ťažkým. Ostatných synov tohoto vrcholu nazveme ľahkými. Hranu k ľahkému vrcholu nazveme ľahkou hranou a hranu k ťažkému vrcholu nazveme ťažkou hranou.*

Teraz popíšeme stratégiu, ktorú bude Kleinov algoritmus používať. Na začiatku si vyberieme väčší zo stromov a menší strom už nebudeme pri rozhodovaní brať do úvahy. Pri výbere variantu Lemy 2.1.2 sa budeme snažiť najprv pracovať s menšími podstromami v lese a ten najväčší si necháme na koniec. Teraz ukážeme, ako sa budeme rozhodovať pri výpočte vzdialenosti lesa $F_1(v)$ od lesa F_2. Nech vrchol u je ťažkým synom vrchola v. S najväčším stromom $T(v)$ lesa $F_1(v)$ budeme pracovať nakoniec. Na začiatku preto budeme používať Lemu 2.1.2 pre najľavejšie korene. Tú budeme používať až dovtedy, kým sa najľavejším koreňom nestane vrchol u. Vtedy začneme používať Lemu 2.1.2 pre najpravejšie korene. Tým postupne odstránime aj vrcholy napravo od u a na výpočet vzdialenosti nám ostane strom $T(u)$.

Takýto prístup je výhodný preto, že počet dosiahnuteľných podlesov pre väčší strom sa zmenší. Nevýhodou však je, že u menšieho stromu musíme uvažovať všetky jeho podlesy, tak ako pri jednoduchom algoritme.

Definícia 2.4.2 *Dosiahnuteľné podlesy stromu T budú tvoriť všetky dosiahnuteľné podlesy jeho ľahkých vrcholov.*

Teraz rekurzívne definujeme dosiahnuteľné podlesy pre vrcholy stromu. Pre vrchol v je $F(v)$ dosiahnuteľný podles. Nech u je ťažký syn vrcholu v a nech l, r označuje počet vrcholov

naľavo, napravo od vrcholu u v lese $F(v)$. Podlesy tvaru $F(v)^{(i,0)}, 0 \leq i \leq l$, $F(v)^{(l,j)}, 0 \leq j \leq r$ budú dosiahnuteľné pre vrchol v. Nakoniec všetky dosiahnuteľné podlesy vrchola u budú dosiahnuteľné aj pre vrchol v. Počet dosiahnuteľných podlesov je pre vrchol v rovný počtu jeho synov $|F(v)|$.

Je vidno, že dosiahnuteľné podlesy sme definovali podľa toho, ako Kleinov algoritmus bude pri výpočte postupovať. Teda počas výpočtu budeme počítať jedine vzdialenosť dosiahnuteľného podlesa jedného stromu a ľubovoľného podlesa druhého. Pre druhý podstrom teda máme $O(|T_2|^2)$ podlesov, ktoré musíme brať do úvahy. Teraz ukážeme ohraničenie počtu dosiahnuteľných podlesov prvého stromu, na ktoré počas výpočtu narazíme.

Definícia 2.4.3 *Redukovanou hĺbkou vrchola v vzhľadom na ľahké vrcholy nazveme počet ľahkých vrcholov na ceste od vrcholu v ku koreňu. Označovať ju budeme* $\mathrm{lh}(v)$.

Lema 2.4.4 *[HT84] Pre strom T_1 a ľubovoľný vrchol $v \in T_1$ platí* $\mathrm{lh}(v) \leq \log |T_1| + O(1)$.

Dôkaz. Ak chceme dokázať túto lemu, stačí si uvedomiť, že ak na ceste od vrcholu v ku koreňu narazíme na ľahký vrchol, musí existovať jeho ťažký súrodenec, ktorý má pod sebou viac vrcholov, ako tento ľahký vrchol. Stačí teda počítať vrcholy stromu T na ceste od vrcholu v. Keďže každým ľahkým vrcholom sa doterajší počet vrcholov stromu zdvojnásobí, redukovaná hĺbka vzhľadom na ľahké vrcholy bude logaritmická od veľkosti stromu.

Lema 2.4.5 *Pre strom T_1 je počet dosiahnuteľných podlesov ohraničený číslom$O(|T_1|\log|T_1|)$.*

Dôkaz. Pri dôkaze budeme postupovať podobne ako pri Leme 2.3.5. Začneme sumovať cez ľahké vrcholy. Z definície dosiahnuteľných podlesov vieme, že vrchol v má $|F_1(v)|$ dosiahnuteľných podlesov. Najdôležitejším krokom pri ohraničovaní je zámena sumácie. Namiesto počtu synov ľahkých vrcholov budeme sumovať počet ľahkých predkov všetkých vrcholov. Nakoniec už len ohraničíme redukovananú hĺbku vrcholu pomocou predošlej Lemy 2.4.4.

$$\sum_{v \text{ je lahky}} |F_1(v)| < \sum_{v \in T_1} \mathrm{lh}(v) = \sum_{v \in T_1} (\log |T_1| + O(1)) = O(|T_1| \log |T_1|)$$

Veta 2.4.6 *[Kle98] Problém výpočtu editačnej vzdialenosti dvoch usporiadaných stromov T_1 a T_2 sa dá vyriešiť v čase $O(|T_2|^2|T_1| \log |T_1|)$ a pamäti $O(|T_1||T_2|)$.*

Pri rekurzii uvažujeme dosiahnuteľné podproblémy lesa T_1 a všetky podproblémy lesa T_2. Spracovanie podroblému trvá konštantný čas, a preto je zložitosť algoritmu závislá len na počte podproblémov. Pamäťová zložitosť je pri priamej rekurzívnej implementácii $O(|T_2|^2|T_1| \log |T_1|)$, dá sa však zlepšiť použitím dynamického programovania.

Poslednou z výhod tohoto algoritmu je možnosť rozšíriť ho na stromy bez koreňov, bez zmeny časovej zložitosti [Kle98].

2.5 Demainov algoritmus

V tejto časti ukážeme Demainov algoritmus [DMRW07], ktorý bude vylepšením Kleinovho algoritmu. Rovnako pôjde o algoritmus používajúci stratégiu pokrytia a bude založený na rekurzii z Lemy 2.1.2. Dôležitým krokom v tomto algoritme je predpočítanie niektorých podproblémov.

Najprv ukážeme lemu, ktorá identifikuje podproblémy, na ktoré narazíme v každom algoritme používajúcom stratégiu pokrytia.

Lema 2.5.1 *Majme algoritmus využívajúci stratégiu pokrytia S. Pri výpočte vzdialenosti $\delta(T_1, T_2)$ bude podproblém $\delta(F_1(v), F_2(w)), v \in T_1, w \in T_2$ vždy dosiahnuteľný.*

Dôkaz. Stačí ukázať, že v rekurzii existuje cesta, ktorou sa dostaneme až k podproblému $\delta(F_1(v), F_2(w))$. Najprv si musíme uvedomiť, že s vrcholmi $v' \in T(v)$ a $w' \in T(w)$ budeme pracovať až potom, ako spracujeme v, w. V rekurzii budeme postupovať tak, že na začiatku budeme mazať vrcholy zo stromu T_1, až kým sa vrchol v nestane najľavejším alebo najpravejším koreňom. Následne budeme mazať vrcholy zo stromu T_2, kým sa w nestane najľavejším alebo najpravejším koreňom. Tým dospejeme k jednej z týchto možností:

(a) v a w sú najpravejšie (najľavejšie) korene F_1', F_2' a stratégia je vpravo (vľavo).
Teraz spárujeme v a w, čím musíme vypočítať podproblém $\delta(F_1(v), F_2(w))$.

(b) v a w sú najpravejšie (najľavejšie) korene F_1', F_2' a stratégia je vľavo (vpravo).
Určite aspoň jeden z lesov F_1',F_2' nemôže byť strom, inak nastáva zároveň prvý prípad. Preto budeme pokračovať v zmazávaní vrcholov z lesov F_1', F_2', kým sa z nich nestanú stromy, alebo sa nezmení smer, ktorý nám určuje stratégia.

(c) v je najpravejší koreň F_1' a w je najľavejší koreň F_2'.
Kým platí, že $S(F_1', F_2')$ je vľavo, budeme zmazávať vrcholy z F_1'. V opačnom prípade budeme mazať vrcholy z F_2'. Týmto sa dopracujeme k vyššie spomenutým prípadom.

(d) v je najľavejší koreň F_1' a w je najpravejší koreň F_2'.
Tento prípad je symetrický k prípadu (c).

Rozoberme si teraz Kleinov algoritmus, v čom spočíva jeho výhoda oproti predošlým algoritmom a čo je možné zlepšiť. Základným krokom stratégie bolo rozhodovanie sa na základe väčšieho stromu. Väčší strom sme si vybrali na začiatku a rozhodovali sme sa vždy podľa tohoto stromu. Počas rekurzie sa nám teda mohlo stať, že druhý strom sa stal väčším a my sme sa museli rozhodovať podľa menšieho podstromu. Nemôžeme však v ľubovoľnom kroku výpočtu zmeniť strom, podľa ktorého sa rozhodujeme, pretože by nám mohol vzrásť počet dosiahnuteľných podproblémov a teda by sa zhoršila časová zložitosť. Zmenu stromu, podľa ktorého sa rozhodujeme, môžeme spraviť iba v niektorých podproblémoch. My si tieto podproblémy predpočítame, pričom pri každom sa budeme rozhodovať podľa aktuálne väčšieho podstromu a nakoniec použijeme Kleinov algorimus, ktorý sa bude rozhodovať iba podľa pôvodne väčšieho podstromu a využije predpočítané výsledky.

Predpokladajme, že vieme hodnotu $\delta(F_1(v), F_2(w))$ pre $v \in T_1, w \in T_2$. To sú podproblémy, o ktorých vďaka Leme 2.5.1 vieme, že sú pre každý algoritmus stratégie pokrytia dosiahnuteľné. Na základe týchto informácii vypočítame $\delta(T_1, T_2)$. Na výpočet použijeme Kleinov algoritmus a zameriame sa na to, ako nám dané výsledky zmenšia zložitosť. Počas rekurzie vrcholy buď mažeme, alebo spárujeme. Výsledky, ktoré sme dostali na začiatku môžeme priamočiaro využiť pri spárovaní vrcholov a miesto dvoch podproblémov budeme počítať len jeden. To nám zmenší počet dosiahnuteľných podlesov väčšieho stromu T_1 z $|T_1| \log |T_1|$ na $|T_1|$. Počet dosiahnuteľných podlesov stromu T_2 ostane nezmenený $|T_2|^2$.

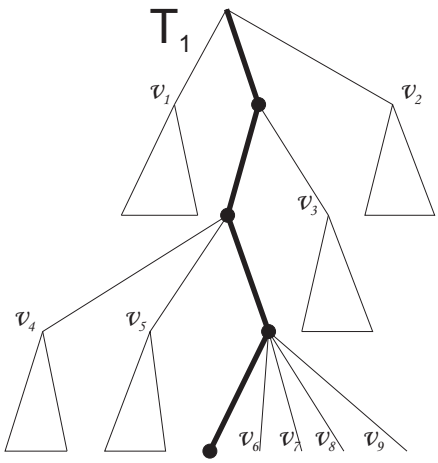
Definícia 2.5.2 *Ťažkú cestu stromu T_1 nazveme cestu od koreňa stromu vedúcu cez ťažké vrcholy.*

Dôležité pozorovanie, ktoré musíme spraviť je, že počas výpočtu vzdialenosti $\delta(T_1, T_2)$ nemusíme využiť vzdialenosti $\delta(F_1(u), F_2(w))$ pre u ležiace na ťažkej ceste stromu T_1. To preto, že podproblémy, ktoré potrebujeme na vyriešenie týchto vzdialeností musíme vypočítať znova kvôli vzdialenosti $\delta(T_1, T_2)$. Bez zmeny zložitosti teda môžeme vypočítať aj $\delta(F_1(u), F_2(w))$ pre u ležiace na ťažkej ceste stromu T_1.

Definícia 2.5.3 *Definujme* LKorene(T_1) *ako množinu koreňov stromov lesa, ktorý dostaneme po odstránení ťažkej cesty zo stromu T_1.*

Ak vypočítame vzdialenosti $\delta(T_1(v), T_2)$ pre $v \in$ LKorene(T_1) podľa Lemy 2.5.1, počas tohoto výpočtu vypočítame aj $\delta(F_1(v'), F_2(w))$ pre v' neležiace na ťažkej ceste stromu T_1. Týmto krokom vypočítamé podproblémy, o ktorých sme predpokladali, že ich dostaneme predpočítané.

Na tieto podproblémy narazíme aj počas rekurzie v Kleinovom algoritme. Kleinov algorimus sa však pri výpočte $\delta(T_1(v), T_2), v \in$ LKorene(T_1) rozhoduje vždy podľa stromu



Obr. 2.2: Ťažká cesta stromu T_1 a vrcholy v_i tvoriace množinu LKorene(T_1).

$T_1(v)$, pretože strom T_1 bol na začiatku výpočtu väčší. Môže teda nastať situácia, že $|T_1(v)| < |T_2|$ a Kleinov algoritmus sa rozhoduje podľa menšieho podstromu. V Demainovom algoritme sa pri výpočte vzdialenosti pre každý vrchol $v \in$ LKorene(T_1) znova rozhodneme, podľa ktorého stromu budeme vzdialenosť počítať. Ak teda $|T_1(v)| < |T_2|$, bude sa naša stratégia rozhodovať podľa väčšieho podstromu T_2.

Algoritmus 2.5.4 *Rekurzívny popis Demainovho algoritmu pre výpočet $\delta(T_1, T_2)$:*

(1.) Ak $|T_1| < |T_2|$, namiesto $\delta(T_2, T_1)$ vypočítaj $\delta(T_2, T_1)$.

(2.) Rekurzívne vypočítaj $\delta(T_1(v), T_2)$ pre $v \in$ LKorene(T_1).

(3.) Vypočítaj $\delta(T_1, T_2)$ pomocou stratégie pokrytia $S(F_1', F_2') =$ vľavo, ak najľavejší koreň lesa F_1' nie je ťažkým synom svojho rodiča. Inak $S(F_1', F_2') =$ vpravo. Počas výpočtu nebudeme rekurzívne počítať podproblémy, ktoré sme vypočítali v druhom kroku výpočtu.

Prvý krok výpočtu nám zaručí, že T_1 bude vždy väčší zo stromov. V druhom kroku predpočítame vzdialenosti $\delta(F_1(v), F_2(w))$ pre v neležiace na ťažkej ceste stromu T_1 a $w \in T_2$. Tretí krok je už totožný s Kleinovým algoritmom, iba nebudeme počítať podproblémy vypočítané v druhom kroku.

Teraz rozoberieme časovú zložitosť tohoto algoritmu. Keďže je to algoritmus používajúci stratégiu pokrytia, bude jeho zložitosť závisieť iba od počtu podproblémov. Tieto podproblémy budeme počítať po jednotlivých krokoch algoritmu. Triviálnym nazveme podproblém,

ak jeden zo stromov je prázdny. Počet triviálnych podproblémov môžeme ľahko ohraničiť $O(|T_1|^2 + |T_2|^2)$. Ďalej už tieto podproblémy nebudeme započítavať.

Označme $R(T_1, T_2)$ počet netriviálnych podproblémov, na ktoré narazíme pri výpočte vzdialenosti $\delta(T_1, T_2)$. V druhom kroku algoritmu počítame vzdialenosť $\delta(T_1(v), T_2)$ pre $v \in \text{LKorene}(T_1)$. Preto počet podproblémov, na ktoré narazíme, bude:

$$\sum_{v \in \text{LKorene}(T_1)} R(T_1(v), T_2)$$

Lema 2.5.5 *Pre tretí krok algoritmu je počet dosiahnuteľných podproblémov* $|T_1||T_2|^2$.

Dôkaz. V treťom kroku používame Kleinov algoritmus, avšak nepočítame podproblémy z druhého kroku. Pre strom T_2 musíme uvažovať všetkých $O(|T_2|^2)$ podproblémov, ktoré dostaneme ľubovoľným zmazávaním najľavejších a najpravejších koreňov. Pre strom T_1 nám stratégia presne určuje postupnosť vrcholov na zmazanie. Preto pri zmazávaní narazíme na $O(|T_1|)$ podproblémov. Ďalšie podproblémy stromu T_1, ktoré musíme započítať, vzniknú pri spárovaní vrcholov. Ak spárujeme vrchol u stromu T_1, ktorý neleží na ťažkej ceste, nemusíme započítať podproblémy z výpočtu $\delta(F_1(u), F_2')$, pretože sme ich započítali pri druhom kroku algoritmu. Ak spárujeme vrchol u, ktorý leží na ťažkej ceste, vďaka našej stratégii, ktorá pracuje s ťažkými vrcholmi nakoniec, vieme, že tento vrchol tvorí koreň stromu. Preto pri spárovaní dostávame rovnaký podproblém, ako keby sme tento vrchol zmazali. Celkový počet dosiahnuteľných podproblémov pre strom T_1 bude $|T_1|$.

Teraz už môžeme ohraničiť počet podproblémov, na ktoré narazíme v závislosti od veľkosti stromov. Preto ak $|T_1| \geq |T_2|$ platí:

$$R(T_1, T_2) \leq |T_2|^2|T_1| + \sum_{v \in \text{LKorene}(T_1)} R(T_1(v), T_2)$$

inak:

$$R(T_1, T_2) \leq |T_1|^2|T_2| + \sum_{v \in \text{LKorene}(T_2)} R(T_2(v), T_1)$$

Lema 2.5.6 $R(T_1, T_2) \leq 4(|T_1||T_2|)^{3/2}$

Dôkaz. Postupovať budeme indukciou vzhľadom na $|T_1| + |T_2|$. Ak $|T_1| + |T_2| = 0$, potom sú obidva stromy prázdne a platí, že $R(T_1, T_2) = 0$. Pri indukčnom kroku sú dva symetrické

prípady. Ak $|T_1| > |T_2|$, potom platí

$$\begin{aligned} R(T_1,T_2) &\leq |T_2|^2|T_1| + \sum_{v\in \mathrm{LKorene}(T_1)} 4(|T_1(v)||T_2|)^{3/2} \\ &= |T_2|^2|T_1| + 4|T_2|^{3/2} \sum_{v\in \mathrm{LKorene}(T_1)} |T_1(v)|^{3/2} \\ &= |T_2|^2|T_1| + 4|T_2|^{3/2} \sum_{v\in \mathrm{LKorene}(T_1)} |T_1(v)| \max_{v\in \mathrm{LKorene}(T_1)} \sqrt{|T_1(v)|} \\ &= |T_2|^2|T_1| + 4|T_2|^{3/2}|T_1|\sqrt{\frac{|T_1|}{2}} \\ &= |T_2|^2|T_1| + \sqrt{8}(|T_1||T_1|)^{3/2} \\ &\leq 4(|T_1||T_2|)^{3/2} \end{aligned}$$

V prvom kroku využijeme indukčný predpoklad na $R(T_1(v), T_2)$. V druhom vyberieme pred sumu to, čo sa nám nemení. V treťom kroku nahradíme odmocninu $\sqrt{|T_1(v)|}$ maximom zo všetkých vrcholov, cez ktoré sumujeme. Pre $v \in \mathrm{LKorene}(T_1)$ platí, že $|T_1(v)| < \frac{|T_1|}{2}$, inak by bol vrchol v ťažkým a nemohol by patriť do množiny $\mathrm{LKorene}(T_1)$. Ostala nám teda suma $\sum_{v\in \mathrm{LKorene}(T_1)} |T_1(v)|$, ktorú môžeme ohraničiť počtom vrcholov stromu $|T_1|$, keďže vrcholy v množine $\mathrm{LKorene}(T_1)$ tvoria navzájom nezávislé podstromy. Posledné ohraničenie môžeme spraviť vďaka predpokladu $|T_1| > |T_2|$.

Dôkaz indukčného kroku pre $|T_1| < |T_2|$ by bol symetrický.

Veta 2.5.7 *[DMRW07] Problém výpočtu editačnej vzdialenosti dvoch usporiadaných stromov T_1 a T_2 o počte vrcholov rádovo N sa dá vyriešiť v čase $O(|N^3|)$ a pamäti $O(N^2)$.*

Ako si môžeme všimnúť, predošlá veta platí pre stromy, ktoré majú rádovo rovnako vrcholov. Teraz rozoberieme časovú zložitosť presne na základe počtu vrcholov jednotlivých stromov. Budeme uvažovať, že $|T_1| = n, |T_2| = m$, pričom $n > m$. Môžeme si všimnúť, že v druhom kroku Demainovho algoritmu 2.5.4 robíme len rekurzívne volania, ale nevytvárame dosiahnuteľné podproblémy. Dosiahnuteľné podproblémy vznikajú v treťom kroku každého rekurzívneho volania.

Označenie 2.5.8 *V nasledujúcich úvahách budeme používať množiny $A, B \subseteq T_1$ definované takto:*

$$\begin{aligned} A &= \{a \textit{ je ľahký vrchol stromu } T_1 : |T_1(a)| < m\} \\ B &= \{b \in T_1 - A : b \in \mathrm{LKorene}(T_1(a)) \textit{ pre nejaké } a \in A\} \end{aligned}$$

Koreň stromu T_1 patrí určite do množiny A. Vrcholy patriace do množín A a B sú vrcholy, pre ktoré robíme rekurzívne volania s celým podstromom T_2, teda musíme vypočítať $\delta(T_1(v), T_2)$ pre $v \in A \cup B$. Taktiež si treba uvedomiť, že vrcholy z množiny B sú posledné, pre ktoré ešte robíme tieto rekurzívne volania s celým stromom T_2.

Teraz spočítame osobitne:

(i) dosiahnuteľné podproblémy z tretieho kroku Demainovho algoritmu, ktoré vznikajú pri rekurzívnych volaniach $\delta(T_1(a), T_2)$, pre všetky $a \in A$

(ii) všetky dosiahnuteľné podproblémy z rekurzívnych volaní v druhom kroku a dosiahnuteľné podproblémy pri treťom kroku algoritmu, na ktoré narazíme pri výpočte $\delta(T_1(b), T_2)$, pre $b \in B$ (teda $\sum_{b \in B} R(T_1(b), T_2)$)

Musíme ukázať, že toto bude tvoriť všetky dosiahnuteľné podproblémy pri výpočte vzdialenosti $\delta(T_1, T_2)$. Aby sme to mohli ukázať, rozoberieme, čo sa robí pri výpočte $\delta(T_1, T_2)$. Keďže koreň stromu patrí do množiny A, všetky podproblémy, ktoré vzniknú v treťom kroku algoritmu, započítame v (i). Teraz uvažujme rekurzívne volania v 2. kroku algoritmu pre $\delta(T_1, T_2)$. Tie tvoria podproblémy $\delta(T_1(v), T_2)$ pre $v \in \text{LKorene}(T_1)$.

Pre vrcholy $v \in \text{LKorene}(T_1)$ ale platí, že sú buď v množine A, alebo v množine B. Ak platí $|T_1(v)| \geq |T_2|$, potom $v \in A$, inak $v \in B$. Ak teda $v \in B$ vieme, že všetky podproblémy, na ktoré narazíme pri výpočte $\delta(T_1(v), T_2)$ sú započítané v (ii). Pri vrchole $v \in A$ spočítame dosiahnuteľné podproblémy $\delta(T_1(v), T_2)$, rovnako ako sme to spravili s koreňom stromu T_1. Teda spočítame osobitne počet podproblémov z tretieho kroku výpočtu $\delta(T_1(v), T_2)$ a osobitne počet dosiahnuteľných podproblémov z rekurzívnych volaní z druhého kroku $\delta(T_1(u), T_2), u \in \text{LKorene}(T_1(v))$. Týmto sme ukázali, že (i) a (ii) tvoria spolu všetky podproblémy, na ktoré počas výpočtu narazíme.

Definícia 2.5.9 *Redukovanou hĺbkou vrcholu v vzhľadom na množinu A nazveme počet vrcholov z množiny A na ceste od tohoto vrchola ku koreňu. Označovať ju budeme* $\text{rh}_\text{A}(v)$.

Lema 2.5.10 *[DMRW07]Pre vrchol v stromu T_1 platí, že* $\text{rh}_\text{A}(v) \leq 1 + \log \frac{n}{m}$.

Teraz už môžeme ohraničiť počet podproblémov typu (i) a typu (ii). Z Lemy 2.5.5 vieme, že v treťom kroku výpočtu $\delta(T_1(a), T_2)$ je počet dosiahnuteľných podproblémov $|T_1(a)||T_2|^2$. Preto je počet dosiahnuteľných podproblémov typu (i) rovný $\sum_{a \in A} |T_1(a)||T_2|^2$. Pre výpočet tejto sumy použijeme rovnaký postup ako v Kleinovom algoritme.

$$|T_2|^2 \sum_{a \in A} |T_1(a)| = m^2 \sum_{v \in T_1} 1 + \text{rh}_\text{A}(v) \leq m^2 \sum_{v \in T_1} (2 + \log \frac{n}{m}) = m^2 n (2 + \log \frac{n}{m})$$

Podproblémy typu (ii) tvoria sumu $\sum_{b\in B} R(T_1(b), T_2)$. Na túto sumu použijeme Lemu 2.5.6. Potom už len využijeme, že $|T_1(b)| < m$ a že $\sum_{b\in B} |T_1(b)| \leq |T_1|$, pretože stromy $T_1(b)$ sú pre rôzne b nezávislé.

$$\begin{aligned}\sum_{b\in B} R(T_1(b), T_2) &\leq 4|T_2|^{3/2} \sum_{b\in B} |T_1(b)|^{3/2} \leq 4|T_2|^{3/2} \sum_{b\in B} |T_1(b)| \max_{b\in B} \sqrt{|T_1(b)|} \\ &\leq 4|T_2|^{3/2}|T_1|\sqrt{m} \leq 4m^2 n\end{aligned}$$

Celkový počet dosiahnuteľných podproblémov pre $\delta(T_1, T_2)$ je najviac $m^2 n(2+\log \frac{n}{m}) + 4m^2 n = O(m^2 n(1 + \log \frac{n}{m}))$.

Veta 2.5.11 *[DMRW07] Problém výpočtu editačnej vzdialenosti dvoch usporiadaných stromov T_1 a T_2, kde $|T_1| > |T_2|$ sa dá vyriešiť v čase $O(|T_2|^2|T_1| \log \frac{|T_1|}{|T_2|})$ a pamäti $O(|T_1||T_2|)$.*

Pamäťová zložitosť je pri jednoduchej rekurzívnej implementácii $O(|T_2|^2|T_1| \log \frac{|T_1|}{|T_2|})$, dá sa však zlepšiť použitím dynamického programovania a výpočtom podprobémov v presnom poradí tak, aby nám stačilo zapamätať si $\delta(F_1(v), F_2(w))$ [DMRW07].

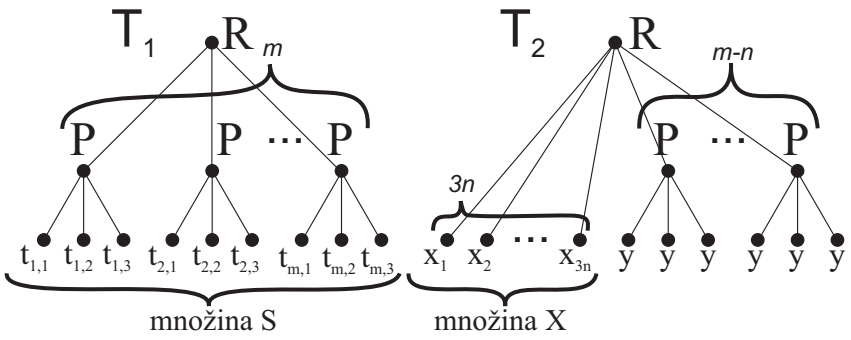
Časová zložitosť Demainovho algoritmu je zároveň spodnou hranicou pre algoritmy z triedy stratégie pokrytia. Teda existujú stromy T_1, T_2, pre ktoré bude výpočet $\delta(T_1, T_2)$ ľubovoľným algoritmom z triedy stratégie pokrytia trvať $O(|T_2|^2|T_1| \log \frac{|T_1|}{|T_2|})$[DMRW07].

2.6 Dôkaz NP úplnosti pre neusporiadané stromy

Tento dôkaz je možné nájsť v [ZWS95]. Na dokázanie, že problém vzdialenosti dvoch neusporiadaných stromov je NP úplný, použijeme redukciu zo známeho problému presného pokrytia trojprvkovými množinami (Exact Cover by 3-Sets Problem), o ktorom je dokázané, že je NP úplný.

Problém 2.6.1 *Presné pokrytie trojprvkovými množinami (X3C): Majme konečnú množinu X takú, že $|X| = 3n$ a množinu S, ktorá obsahuje trojprvkové podmnožiny množiny X. Otázkou je, či existuje presné pokrytie množiny X množinou S, teda či existuje podmnožina $S' \subseteq S$ taká, že každý prvok X sa nachádza práve v jednej trojici z množiny S'.*

Majme inštanciu X3C problému $X = \{x_1, x_2, \ldots, x_{3n}\}$, $S = \{S_1, S_2, \ldots, S_m\}$, pričom $S_i = \{t_{i,1}, t_{i,2}, t_{i,3}\}$, kde $t_{i,j} \in X, 1 \leq j \leq 3$. Ak $m < n$, tak presné pokrytie neexistuje. Pre $m = n$ stačí zistiť, či $\bigcup_{1\leq i\leq m} S_i = X$. Preto jediná možnosť, o ktorej má zmysel ďalej uvažovať je $m > n$.



Obr. 2.3: Neusporiadané stromy T_1, T_2.

Veta 2.6.2 *Problém výpočtu vzdialenosti dvoch neusporiadaných stromov je NP úplný.*

Dôkaz. Na základe inštancie X3C vytvoríme dva stromy T_1, T_2 (viď. obr. 2.3). Strom T_1 bude reprezentáciou množiny S a strom T_2 bude predstavovať množinu X. Potom vypočítame vzdialenosť týchto dvoch stromov a na základe toho budeme vedieť rozhodovať, či inštancia X3C obsahuje alebo neobsahuje presné pokrytie.

Strom T_1 zostrojíme nasledovne. Jeho koreň bude vrchol so symbolom R. Pod tento koreň budeme dávať prvky množiny S. Pre každý prvok množiny S vytvoríme nový vrchol so symbolom P a pod tento vrchol dáme jednotlivé prvky trojice, teda vrcholy $t_{i,1}, t_{i,2}, t_{i,3}$.

Koreň stromu T_2 bude tiež vrchol so symbolom R. Pod tento koreň dáme všetky prvky množiny X, teda vrcholy $x_1, x_2, \ldots, x_{3n}$. Okrem nich pod koreň dáme $m - n$ podstromov tvorených vrcholom so symbolom P s tromi synmi so symbolmi y.

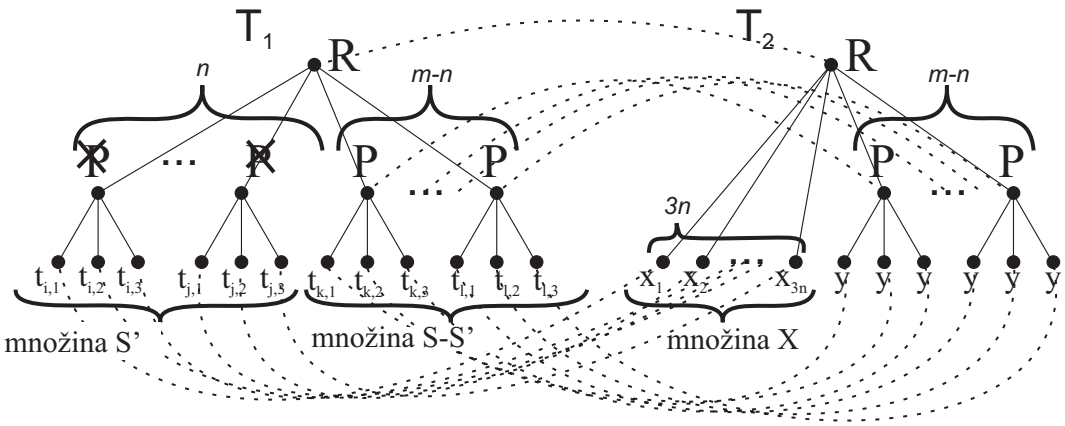
Lema 2.6.3 *Ak $\delta(T_1, T_2) = 3m - 2n$, potom S obsahuje presné pokrytie X.*

Dôkaz. Nech M je minimálne editačné zobrazenie jednotkovej vzdialenosti stromov T_1 a T_2. Označme l_d počet zmazaní, l_i počet vložení a l_m počet premenovaní v tomto editačnom zobrazení. Keďže počet vrcholov menia iba operácie zmazania a vloženia, musí platiť $|T_1| - l_d + l_i = |T_2|$. Keďže $|T_1| = 1 + 4m$ a $|T_2| = 1 + 4m - n$, dostávame rovnosť $l_d - l_i = n$.

Strom T_2 obsahuje $3(m - n)$ vrcholov so symbolom y, ktoré musia vzniknúť operáciou premenovania alebo vloženia, preto platí $l_c + l_i \geq 3(m - n)$. Ak k tejto nerovnosti pripočítame rovnosť $l_d - l_i = n$, dostaneme nerovnosť:

$$l_d + l_i + l_m \geq 3m - 2n + l_i$$

My ale vieme, že vzdialenosť je presne $3m - 2n$, teda $l_i = 0$, $l_d = n$, $l_m = 3m - 3n$. Keďže nepoužijeme operáciu pridania, vrcholy y stromu T_2 museli vzniknúť operáciou



Obr. 2.4: Minimálne editačné zobrazenie zostrojené z presného pokrytia množiny X.

premenovania, preto všetky operácie premenovania musíme aplikovať na vrcholy y. Už si stačí všimnúť, že strom T_1 obsahuje o n viac vrcholov so symbolom P ako strom T_2. Na tieto vrcholy musíme použiť operáciu zmazania a budú to práve tie vrcholy, ktorých synovia budú v editačom zobrazení s $x_1, \ldots, x_{3n}$, čiže budú tvoriť presné pokrytie.

Lema 2.6.4 *Ak S obsahuje presné pokrytie X, platí $\delta(T_1, T_2) = 3m - 2n$.*

Dôkaz. Zostrojíme editačné zobrazenie s cenou $3m - 2n$ (vid. obr. 2.4). Nech $S' \subseteq S$ je presné pokrytie množiny X. V editačnom zobrazení zmažeme vrcholy so symbolom P, ktoré sú v pokrytí S'. Zároveň spárujeme príslušné vrcholy z množiny S' s vrcholmi množiny X. Synov vrcholov z $S - S'$ premenenujeme na y a spárujeme nadradené vrcholy so symbolmi P. Celkovo teda zmažeme $|S'|$ vrcholov a premenujeme $3|S - S'|$ vrcholov. Preto platí $\delta(T_1, T_2) \leq 3m - 2n$. Z dôkazu predošlej lemy vieme, že platí $l_d + l_i + l_m \geq 3m - 2n + l_i$, a preto menšie editačné zobrazenie nemôže existovať.

Z predchádzajúcich dvoch liem vyplýva, že problém presného pokrytia trojprvkovými množinami, ktorý je NP úplný, vieme rozhodovať na základe výpočtu vzdialenosti neusporiadaných stromov.

Kapitola 3

Obmedzená editačná vzdialenosť stromov

V tejto kapitole definujeme obmedzenú editačnú vzdialenosť. Táto vzdialenosť bola prvýkrát predstavená v práci [Zha95]. Obmedzená editačná vzdialenosť vznikne obmedzením editačného zobrazenia editačnej vzdialenosti predstavenej v prvej kapitole. Základná idea obmedzenia bude zobrazovať rôzne podstromy stromu T_1 do rôznych podstromov stromu T_2.

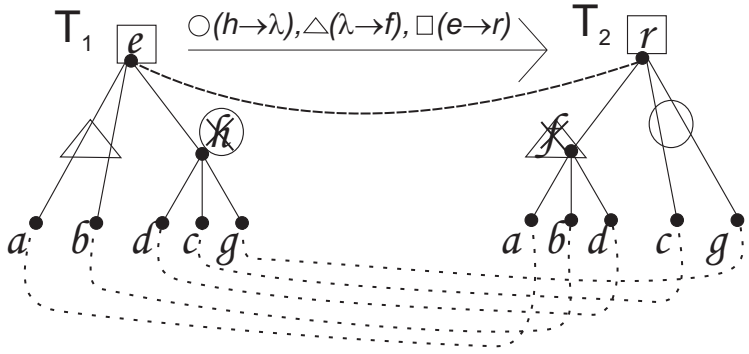
Pri niektorých aplikáciach problému vzdialenosti stromov budeme požadovať takéto obmedzenie [TT88]. V ostatných aplikáciach má obmedzená editačná vzdialenosť výhodu v existencii polynomiálneho algoritmu pre neusporiadané stromy a rýchlejšieho algoritmu pre usporiadané stromy.

Na začiatku kapitoly definujeme obmedzenú editačnú vzdialenosť dvoch stromov. Potom ukážeme algoritmus [Zha95] pre usporiadané stromy s rozborom jeho časovej zložitosti. Na koniec kapitoly ukážeme zmeny v algoritme, ktoré nám umožnia vypočítať vzdialenosť neusporiadaných stromov.

Označenie 3.0.5 *Najnižšieho spoločného predka vrcholov v_1, v_2 budeme označovať* $\mathrm{lca}(v_1,v_2)$

Definícia 3.0.6 *Obmedzené editačné zobrazenie je usporiadaná trojica (M_c, T_1, T_2), kde $M_c \subseteq T_1 \times T_2$ a pre všetky páry $(v_1, w_1), (v_2, w_2) \in M_c$ platí:*

1. *$v_1 = v_2$ práve vtedy, keď $w_1 = w_2$ (jedno-jednoznačnosť)*
2. *v_1 je predchodca v_2 práve vtedy, keď w_1 je predchodca w_2 (zachovanie predchodcov)*
3. *v_1 je naľavo od v_2 práve vtedy, keď w_1 je naľavo w_2 (zachovanie usporiadania súrodencov)*



Obr. 3.1: Editacné zobrazenie, ktoré nie je obmedzeným editacným zobrazením.

4. *Nech* $(v_3, w_3) \in M_c$. *Vrchol* $\mathrm{lca}(v_1, v_2)$ *je priamy predchodca* v_3 *práve vtedy, keď vrchol* $\mathrm{lca}(w_1, w_2)$ *je priamy predchodca* w_3.

Prvé tri podmienky sú totožné s podmienkami editačého zobrazenia. Štvrtá podmienka vytvára obmedzenia pre podstromy, ktoré dostaneme nájdením najnižšieho spoločného predka dvoch vrcholov z editačného zobrazenia. Obmedzenie hovorí, že vrcholy týchto podstromov môžu byť v zobrazení len navzájom. Toto obmedzenie je podobné podmienke zachovania predchodcov s tým rozdielom, že platí aj pre vrcholy, ktoré sa v zobrazení nemusia nachádzať.

Na obrázku 3.1 je príklad editačného zobrazenia, ktoré nie je obmedzeným editacným zobrazením. Platí totiž, že vrchol $e = \mathrm{lca}(a, d)$ je priamy predchodca vrcholu g v strome T_1, ale v strome T_2 vrchol $f = \mathrm{lca}(a, d)$ nie je priamy predchodca vrcholu g .

V celej tejto kapitole sa budeme venovať obmedzenému editačnému zobrazeniu, preto ho budeme často nazývať skrátene zobrazenie. Pretože obmedzené editačné zobrazenie je zároveň editačným zobrazením, pre jeho cenu platí Definícia 1.3.2, teda platí:

$$\gamma(M_c) = \sum_{(v,w)\in M_c} \gamma(v, w) + \sum_{v\in D} \gamma(v, \lambda) + \sum_{w\in I} \gamma(\lambda, w)$$

Definícia 3.0.7 *[Zha95] Obmedzená editačná vzdialenosť dvoch stromov je:*

$$\delta_c(T_1, T_2) = \min\{\gamma(M_c) | (M_c, T_1, T_2) \textit{ je obmedzené editačné zobrazenie}\}$$

Lema 3.0.8 *Nech* T_1, T_2 *sú stromy, potom platí:* $\delta(T_1, T_2) \leq \delta_c(T_1, T_2)$.

Dôkaz. Pretože obmedzené editačné zobrazenie spĺňa podmienky editačného zobrazenia, platí že obmedzená editačná vzdialenosť musí mať aspoň takú cenu ako editačná vzdialenosť.

3.1 Algoritmus pre usporiadané stromy

V tejto podkapitole predstavíme algoritmus z [Zha95] na výpočet obmedzenej editačnej vzdialenosti dvoch usporiadaných stromov. Tento algoritmus bude založený na dvoch rekurzívnych lemách. Prvá bude počítať vzdialenosť dvoch usporiadaných stromov. Druhá sa bude venovať vzdialenosti dvoch usporiadaných lesov. Výsledný algoritmus bude založený na kombinácii týchto dvoch liem.

Lema 3.1.1 *Nech v, w sú korene stromov T_1, T_2. Nech vrchol v má synov $v_1, \ldots, v_i$ a vrchol w má synov $w_1, \ldots, w_j$.*

$$
\begin{aligned}
\delta(\theta,\theta) &= 0 \\
\delta_c(T_1(v),\theta) &= \delta_c(F_1(v),\theta) + \gamma(v \to \lambda) \\
\delta_c(\theta,T_2(w)) &= \delta_c(\theta,F_2(w)) + \gamma(w \to \lambda) \\
\delta_c(T_1(v),T_2(w)) &= \min \begin{cases} \delta_c(\theta,T_2(w)) + \min_{1\le t\le j}\{\delta_c(T_1(v),T_2(w_t)) - \delta_c(\theta,T_2(w_t))\} \\ \delta_c(T_1(v),\theta) + \min_{1\le t\le i}\{\delta_c(T_1(v_t),T_2(w)) - \delta_c(T_1(v_t),\theta)\} \\ \delta(F_1(v),F_2(w)) + \gamma(v \to w) \end{cases}
\end{aligned}
$$

Dôkaz. Prvé tri rovnosti sú triviálne. Pre dôkaz štvrtej rovnosti stačí rozobrať všetky možnosti obmedzeného editačného zobrazenia pre korene v, w.

(a) Vrchol v tvorí pár s vrcholom u stromu T_2 a vrchol w nemá pár v zobrazení. Nech w_t je predchodca vrcholu u. Potom vieme, že všetky vrcholy stromu $T_2 - T_2(w_t)$ môžeme zmazať vďaka podmienke zachovania predchodcov z obmedzeného editačného zobrazenia. V rovnosti je táto možnosť zapísaná zmazaním celého stromu $T_2(w)$ a odpočítaním ceny za zmazanie stromu $T_2(w_t)$.

(b) Vrchol w má pár s vrcholom u stromu T_1 a vrchol v nemá pár v obmedzenom editačnom zobrazení. Táto možnosť je symetrická s prvou, ale pre strom T_2.

(c) Vrcholy v a w majú obidva pár v zobrazení. Aby sme neporušili podmienku zachovania predchodcov, musia tieto vrcholy tvoriť spolu pár.

(d) Obidva vrcholy v, w nemajú páry v zobrazení. Túto možosť nemusíme uvažovať, lebo $\gamma(v \to \lambda) + \gamma(\lambda \to w) \ge \gamma(v \to w)$. Teda zobrazenie, kde obidva vrcholy zmažeme, má väčšiu cenu ako zobrazenie, kde tieto vrcholy spárujeme.

Definícia 3.1.2 *Nech vrchol v má synov $v_1, \ldots, v_i$ a vrchol w má synov $w_1, \ldots, w_j$. Označením $RM(v, w)$ nazveme obmedzené editačné zobrazenie $(M_c, F_1(v), F_2(w))$, pre ktoré navyše platí:*

Ak $(v', w') \in RM(v, w)$, pričom $v' \in T_1(v_s), w' \in T_2(w_t)$, potom pre ľubovoľné $(v'', w'') \in RM(v, w)$ platí $v'' \in T_1(v_s)$ práve vtedy, keď $w'' \in T_2(w_t)$.

Toto označenie nám hovorí, že ak máme obmedzené editačné zobrazenie $RM(v, w)$, tak sa v ňom rôzne podstromy lesa $F_1(v)$ zobrazujú do rôznych podstromov lesa $F_2(w)$. Toto zobrazenie bude kľúčové pre výpočet obmedzenej editačnej vzdialenosti dvoch lesov.

Lema 3.1.3 *Nech vrchol v má synov $v_1, \ldots, v_i$ a vrchol w má synov $w_1, \ldots, w_j$. Potom pre lesy $F_1(v), F_2(w)$ platí:*

$$\begin{aligned}
\delta_c(F_1(v), \theta) &= \sum_{k=1}^{i} \delta_c(T_1(v_k), \theta) \\
\delta_c(\theta, F_2(w)) &= \sum_{k=1}^{j} \delta_c(\theta, T_2(w_k)) \\
\delta_c(F_1(v), F_2(w)) &= \min \begin{cases} \delta_c(\theta, F_2(w)) + \min_{1 \le t \le j}\{\delta_c(F_1(v), F_2(w_t)) - \delta_c(\theta, F_2(w_t))\} \\ \delta_c(F_1(v), \theta) + \min_{1 \le t \le i}\{\delta_c(F_1(v_t), F_2(w)) - \delta_c(F_1(v_t), \theta)\} \\ min_{RM(v,w)}\gamma(RM(v, w)) \end{cases}
\end{aligned}$$

Dôkaz. Prvé dve rovnosti sú triviálne. Pre dôkaz tretej rovnosti rozoberieme všetky možnosti, ktoré môžu pri obmedzenom editačnom zobrazení nastať.

(a) Všetky podstromy $F_1(v)$ sa zobrazia do jedného podstromu $T_2(w_t)$ lesa $F_2(w)$. Ak sa vrchol w_t v zobrazení nenachádza, stačí vypočítať $\delta_c(F_1(v), F_2(w_t))$ a ostatné vrcholy zmazať. Ak je vrchol w_t v zobrazení, potom toto zobrazenie je zahrnuté v $RM(v, w)$.

(b) Všetky podstromy lesa $F_2(w)$ sa zobrazia do jedného podstromu lesa $F_1(v)$. Táto možnosť je symetrická s prvou možnosťou, ale pre les $F_2(w)$.

(c) V zobrazení sú aspoň dva podstromy lesa $F_1(v)$ a aspoň dva podstromy lesa $F_2(w)$. Ukážeme, že takéto zobrazenie spĺňa podmienky $RM(v, w)$. Musíme ukázať, že každý podstrom lesa $F_1(v)$ sa zobrazí do maximálne jedného podstromu lesa $F_2(w)$ a naopak. To dokážeme sporom.

Predpokladajme, že existuje podstrom lesa $F_1(v)$, ktorý sa zobrazí do dvoch podstromov. Možnosť, že existuje podstrom lesa $F_2(w)$, ktorý sa zobrazí do dvoch podstromov

lesa $F_1(v)$, je symetrická. Teda existuje podstrom $T_1(v_t)$, ktorý sa zobrazí do podstromov $T_2(w_x), T_2(w_y)$. V tomto podstrome sa nachádzajú vrcholy $v_{t_x}, v_{t_y} \in T_1(v_t)$, ktoré tvoria páry s vrcholmi $w_{x_t} \in T_2(w_x), w_{y_t} \in T_2(w_y)$, teda platí $(v_{t_x}, w_{x_t}), (v_{t_y}, w_{y_t}) \in M_c$. Keďže podstromy $T_2(w_x)$ a $T_2(w_y)$ sú rôzne, platí $\text{lca}(w_{x_t}, w_{y_t}) = w$. Keďže uvažujeme len o obmedzených editačných zobrazeniach platí, že všetky[1] vrcholy stromu $T_2(\text{lca}(w_{x_t}, w_{y_t})) = T_2(w)$ môžu byť zobrazené len v podstrome $T_1(\text{lca}(v_{t_x}, v_{t_y}))$. Keďže $\text{lca}(v_{t_x}, v_{t_y})$ sa nachádza v podstrome $T_1(v_t)$, v zobrazení sa nachádza len jeden podstrom lesa $F_1(v)$, konkrétne podstrom $T_1(v_t)$. To je spor s prepokladom, že v zobrazení sú aspoň dva podstromy lesa $F_1(v)$.

Teraz ukážeme, ako vypočítať $min_{RM(v,w)}\gamma(RM(v,w))$. Vieme, že obmedzené editačné zobrazenie $RM(v,w)$ zobrazuje rôzne podstromy do rôznych podstromov. Preto, ak hľadáme minimálne zobrazenie, stačí zistiť, ktoré podstromy máme vzájomne zobraziť a ktoré zmazať. Na vyriešenie tohoto podproblému využijeme vzdialenosť na slovách. Podstrom $T_1(v_x)$ si predstavíme ako písmeno v_x a podstrom $T_2(w_y)$ ako písmeno w_y. Nech vrchol v má synov $v_1, \ldots, v_i$ a vrchol w má synov $w_1, \ldots, w_j$. Potom les $F_1(v)$ bude tvoriť slovo $s_1 = v_1.\ldots.v_i$ a les $F_2(w)$ bude predstavovať slovo $s_2 = w_1.\ldots.w_j$.

Môžeme si všimnúť, že editačná vzdialenosť slov s_1 a s_2 je ekvivalentná s obmedzeným editačným zobrazením $RM(v,w)$. Ak zmažeme písmeno v slove s_1, to je totožné so zmazaním príslušného podstromu lesa $F_1(v)$. Zmazanie písmena zo slova s_2 predstavuje zmazanie príslušného podstromu z lesa $F_2(w)$. A nakoniec zmena písmena predstavuje vzájomné zobrazenie príslušných podstromov.

Lema 3.1.4 *Nech vrchol v má synov $v_1, \ldots, v_i$ a vrchol w má synov $w_1, \ldots, w_j$. Potom pre lesy $F_1(v), F_2(w)$ platí:*

$$\min_{RM(v,w)} RM(v,w) = d_E(s_1, s_2), \text{ pričom } s_1 = v_1.v_2 \ldots v_i, s_2 = w_1.w_2 \ldots w_j,$$

kde d_E je editačná vzdialenosť na slovách nad abecedou $\Sigma = \{v_1, \ldots, v_i, w_1, \ldots, w_j\}$, pričom funkciu Ψ ohodnocujúcu jedotlivé operácie na písmenách definujeme takto:

$$\begin{aligned}
\Psi(\binom{v_x}{\varepsilon}) &= \delta_c(T_1(v_x), \theta), \text{ pre } 1 \leq x \leq i \\
\Psi(\binom{\varepsilon}{w_y}) &= \delta_c(\theta, T_2(w_y)), \text{ pre } 1 \leq y \leq j \\
\Psi(\binom{v_x}{w_y}) &= \delta_c(T_1(v_x), T_2(w_y)), \text{ pre } 1 \leq x \leq i, 1 \leq y \leq j
\end{aligned}$$

[1] Samozrejme okrem koreňa, teda okrem vrchola w.

Veta 3.1.5 *[Nán08] Editačná vzdialenosť $d_E(s_1, s_2)$ sa dá vypočítať v čase $O(|s_1||s_2|)$.*

Teraz už máme všetko na zostrojenie algoritmu na výpočet obmedzenej editačnej vzdialenosti dvoch stromov $\delta_c(T_1, T_2)$. Stačí priamo aplikovať Lemy 3.1.1,3.1.3 a 3.1.4. Dosiahnuteľné podproblémy, ktoré musíme vypočítať sú $\delta_c(F_1(v), F_2(w))$ a $\delta_c(T_1(v), T_2(w))$ pre všetky vrcholy $v \in T_1, w \in T_2$. Keďže na vyriešenie podproblému $\delta_c(F_1(v), F_2)(\delta_c(T_1(v), T_2))$ nám stačí vedieť výsledky vzdialenosti pre všetkých jeho synov, môžeme rekurzívny algoritmus prepísať na dynamický s výpočtom vzdialenosti od listov po korene stromov.

Časová zložitosť pre výpočet podproblému $\delta_c(T_1(v), T_2(w))$ je $O(n_i+n_j)$, kde n_i je počet synov v a n_j je počet synov w. Časová zložitosť pre výpočet podproblému $\delta_c(F_1(v), F_2(w))$ je $O(n_i n_j + n_i + n_j)$. Preto pre celkovú zložitosť algoritmu platí

$$\sum_{v\in T_1}\sum_{w\in T_2} O(n_i n_j + 2n_i + 2n_j) \leq \sum_{v\in T_1}\sum_{w\in T_2} O(n_i n_j) \leq O\Big(\sum_{v\in T_1} n_i \sum_{w\in T_2} n_j\Big) \leq O(|T_1||T_2|)$$

Veta 3.1.6 *[Zha95] Problém výpočtu obmedzenej editačnej vzdialenosti dvoch usporiadaných stromov T_1 a T_2 sa dá vyriešiť v čase $O(|T_1||T_2|)$ a pamäti $O(|T_1||T_2|)$.*

3.2 Algoritmus pre neusporiadané stromy

Pre neusporiadané stromy je definovaná vzdialenosť rovnako ako pre stromy usporiadané. Jediný rozdiel je, že z obmedzeného editačného zobrazenia vynecháme podmienku zachovania usporiadania súrodencov, keďže takéto usporiadanie medzi súrodencami neexistuje. Pri konštrukcii algoritmu predstavenom v [Zha96] môžeme využiť Lemy 3.1.1 a 3.1.3, ktoré platia aj pre neusporiadané stromy. Ostáva ukázať jedinú vec, a to spôsob nájdenia minimálneho obmedzeného editačného zobrazenia $\min_{RM(v,w)} \gamma(RM(v, w))$, zavedeného v Definícii 3.1.2.

Pre zopakovanie, obmedzené editačné zobrazenie $RM(v, w)$ je zobrazenie medzi lesmi $F_1(v), F_2(w)$, pre ktoré platí, že rôzne podstromy sa zobrazujú do rôznych podstromov. Pri usporiadaných stromoch sme si zjednodušili pohľad na podstrom ako na písmeno, čiže les predstavoval jedno slovo. Na slovách predstavujúcich lesy sme vypočítali editačnú vzdialenosť, ktorá bolá rovnaká ako cena minimálneho obmedzeného editačného zobrazenia $RM(v, w)$. Pre neusporiadané lesy použijeme podobnú konštrukciu.

Podstrom lesa bude predstavovať vrchol úplného ohodnoteného bipartitného grafu. Jednotlivé partície budú predstavovať lesy $F_1(v)$ a $F_2(w)$ a cena hrany bude predstavo-

vať vzdialenosť príslušných podstromov. Aby sme hranou mohli reprezenovať aj zmazanie podstromu do každej partície navyše pridáme vrcholy predstavujúce prázdne podstromy.

Definícia 3.2.1 *Nech $I = \{v_1, \ldots, v_i\}$ je množina synov vrcholu v, $J = \{w_1, \ldots, w_j\}$ je množina synov vrchola w, $A = \{a_1, \ldots, a_j\}$,$B = \{b_1, \ldots, b_i\}$, $L = I \cup A, R = J \cup B$. Úplný ohodnotený bipartitný graf $G_{v,w}$ zostrojíme z lesov $F_1(v), F_2(w)$ takto:*

(i) $V = L \cup R$

(ii) $E = L \times R$

(iii) $w(x,y) = \delta_c(T_1(x), T_2(y))$ *ak* $x \in I, y \in J$
$w(x,y) = \delta_c(T_1(x), \theta)$ *ak* $x \in I, y \in B$
$w(x,y) = \delta_c(\theta, T_2(y))$ *ak* $x \in A, y \in J$
$w(x,y) = 0$ *ak* $x \in A, y \in B$

Perfektné párovanie na grafe $G_{v,w}$ je párovanie, ktoré pokrýva všetky vrcholy grafu $G_{v,w}$.

Označenie 3.2.2 *Perfektné párovanie na grafe $G_{v,w}$ budeme označovať $MM(v,w)$.*

Kedže $G_{v,w}$ je úplný bipartitný graf a obe partície majú rovnakú veľkosť, v tomto grafe existuje perfektné párovanie.

Definícia 3.2.3 *Cenu párovania definujeme ako:*

$$\gamma(MM(v,w)) = \sum_{(x,y) \in MM(v,w)} w(x,y)$$

.

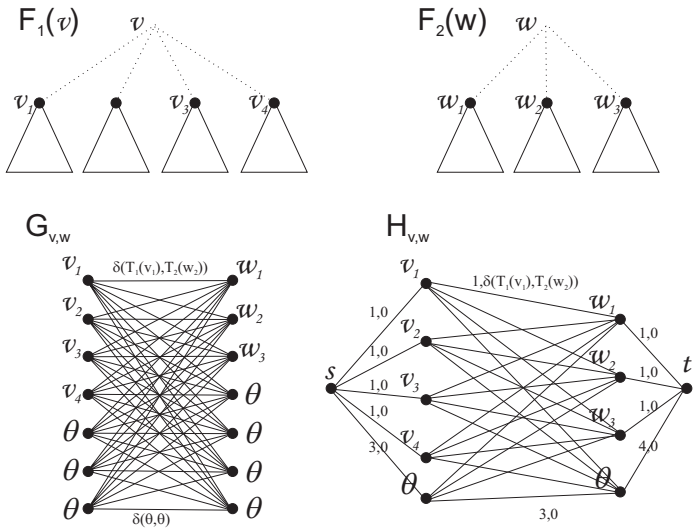
Lema 3.2.4 *Pre minimálne obmedzené editačné zobrazenie $RM(v,w)$ platí:*

$$\min_{RM(v,w)} \gamma(RM(v,w)) = \min_{MM(v,w)} \gamma(MM(v,w))$$

.

Dôkaz. Z perfektného párovania vieme priamo vytvoriť editačné zobrazenie. Páry z perfektného párovania $MM(v,w)$ nám dávajú informáciu, ktorý podstrom máme zmazať, a ktoré podstromy máme zobraziť na seba a teda:

$$\min_{RM(v,w)} \gamma(RM(v,w)) \leq \min_{MM(v,w)} \gamma(MM(v,w))$$



Obr. 3.2: Príkad grafov $G_{v,w}, H_{v,w}$ zostrojených z lesov $F(v)$ a $F(w)$.

Naopak, ak máme obmedzené editačné zobrazenie $RM(v, w)$, vieme ktoré podstromy sa zobrazia na seba a ktoré budú zmazané, preto môžeme jednoducho vytvoriť perfektné párovania, teda:

$$\min_{RM(v,w)} \gamma(RM(v,w)) \geq \min_{MM(v,w)} \gamma(MM(v,w))$$

Ak teda chceme vypočítať $\min_{RM(v,w)} \gamma(RM(v,w))$, zostrojíme graf $G_{v,w}$ a na neho môžeme použiť algoritmus na nájdenie maximálneho párovania s minimálnou cenou. Nevýhodou je, že polovica vrcholov grafu $G_{v,w}$ predstavuje prázdne podstromy. Preto môžeme miesto hľadania párovania v grafe $G_{v,w}$ použiť algoritmus na hľadanie maximálneho toku v grafe $H_{v,w}$, pričom množstvo prázdnych vrcholov nahradíme väčšou kapacitou hrany k jednému prázdnemu vrcholu pre každú partíciu. Príklad zostrojenia grafov $G_{v,w}$ a $H_{v,w}$ je na obrázku 3.2. Presnú konštrukciu grafu $H_{v,w}$ a rozbor časovej zložitosti algoritmu je možné nájsť v práci [Zha96].

Veta 3.2.5 *[Zha96] Problém výpočtu obmedzenej editačnej vzdialenosti dvoch neusporiadaných stromov T_1 a T_2 sa dá vyriešiť v čase $O(|T_1||T_2|D_1D_2\log(D_1 + D_2))$ a pamäti $O(|T_1||T_2|)$, kde D_1, D_2 je maximálny stupeň vrcholov v strome T_1, T_2.*

Kapitola 4

Záver

V tejto práci sme sa zaoberali problémom vzdialenosti stromov. Jedným z prínosov našej práce je, že sme známe algoritmy prezentovali jednotným spôsobom.

V prvej kapitole sme definovali vzdialenosť stromov. Druhá kapitola sa venuje algoritmom na výpočet editačnej vzdialenosti stromov. Táto vzdialenosť je základnou vzdialenosťou definovanou na stromoch. Postupne sme predstavili štyri algoritmy, ktoré riešia problém editačnej vzdialenosti dvoch usporiadaných stromov. Všetky predstavené algoritmy patria do triedy algoritmov stratégie pokrytia. Demainov algoritmus popísaný v podkapitole 2.5 dosahuje optimálnu časovú zložitosť pre algoritmy z triedy stratégie pokrytia. V podkapitole 2.6 sme ukázali dôkaz, že pre neusporiadané stromy je problém výpočtu ich editačnej vzdialenosti NP úplný.

V tretej kapitole práce sme sa venovali obmedzenej editačnej vzdialenosti. Popísali sme algoritmus z [Zha95], ktorý počíta obmedzenú editačnú vzdialenosť dvoch usporiadaných stromov. Neskôr sme ukázali zmeny v algoritme z [Zha96], ktoré umožnia počítať obmedzenú editačnú vzdialenosť dvoch neusporiadaných stromov. Hlavnou výhodou obmedzenej editačnej vzdialenosti je menšia časová náročnosť algoritmu na výpočet vzdialenosti usporiadaných stromov a existencia polynomiálneho algortimu na výpočet vzidalenosti neusporiadaných stromov.

Problematika vzdialenosti stromov je ešte stále skúmaným problémom. V literatúre sa môžeme stretnúť s novými definíciami vzdialeností [LST01], ako aj s problémami súvisiacimi so vzdialenosťou stromov. Keďže spodná časová hranica pre algoritmy riešiace problém editačnej vzdialenosti usporiadaných stromov je dokázaná len pre algoritmy z triedy stratégie pokrytia, môžeme v literatúre nájsť algoritmi využivajúce iné postupy (napr. rýchle násobenie matíc[Che01]). Tieto algoritmy však zatiaľ nedosahujú nižšiu časovú zložitosť ako tu predstavené algoritmy.

Literatúra

[BAS90] Kaizhong Zhang Bruce A. Shapiro. Comparing multiple rna secondary structures using tree comparisons. *Computer Applications in the Biosciences*, 6(4):309–318, 1990.

[Bil05] Philip Bille. A survey on tree edit distance and related problems. *Theor. Comput. Sci.*, 337(1-3):217–239, 2005.

[Che01] Weimin Chen. New algorithm for ordered tree-to-tree correction problem. *J. Algorithms*, 40(2):135–158, 2001.

[DMRW07] Erik D. Demaine, Shay Mozes, Benjamin Rossman, and Oren Weimann. An optimal decomposition algorithm for tree edit distance. In *In Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP*, pages 146–157, 2007.

[DTI03] Serge Dulucq, Hélene Touzet, and Labri Universit Bordeaux I. Analysis of tree edit distance algorithms. In *In Proceedings of the 14th annual symposium on Combinatorial Pattern Matching (CPM*, pages 83–95. Springer-Verlag, 2003.

[HT84] Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.*, 13(2):338–355, 1984.

[Kle98] Philip N. Klein. Computing the edit-distance between unrooted ordered trees. pages 91–102. Springer-Verlag, 1998.

[LST01] Chin Lung Lu, Zheng-Yao Su, and Chuan Yi Tang. A new measure of edit distance between labeled trees. In *COCOON '01: Proceedings of the 7th Annual International Conference on Computing and Combinatorics*, pages 338–348, London, UK, 2001. Springer-Verlag.

[Nán08] Michal Nánási. *Vzdialenosti jazykov.* Bachelor Thesis, Comenius University, 2008.

[Shi91] Frank Y. Shih. Object representation and recognition using mathematical morphology model. *Journal of Systems Integration*, 1(2):235–256, 1991.

[Tai79] Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.

[TT88] E. Tanaka and K. Tanaka. The tree-to-tree editing problem. *IJPRAI*, 2:221–240, 1988.

[ZCZ03] Zhongping Zhang, Rong Li Shunliang Cao, and Yangyong Zhu. Similarity metric for xml documents. In *In Proc. of Workshop on Knowledge and Experience Management*, 2003.

[Zha95] K.Z. Zhang. Algorithms for the constrained editing distance between ordered labeled trees and related problems. 28(3):463–474, March 1995.

[Zha96] Kaizhong Zhang. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3):205–222, 1996.

[ZS89] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.

[ZWS95] Kaizhong Zhang, Jason T. L. Wang, and Dennis Shasha. On the editing distance between undirected acyclic graphs and related problems. In *International Journal of Foundations of Computer Science*, pages 395–407. Springer-Verlag, 1995.