ZADÁNÍ SEMESTRÁLNÍ PRÁCE
VÝPOČET PODOBNOSTI N-ÁRNÍCH STROMŮ

Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která jako vstup načte ze souborů, předaných jako parametry na příkazové řádce, dva obecně n-ární stromy (formát vstupních souborů upřesněn dále v textu) dané výčtem uzlů a jejich potomků. **Cílem aplikace je tyto dva stromy porovnat pomocí metriky *Tree Edit Distance* a vypsat jejich minimální vzájemnou vzdálenost ve smyslu této metriky.**

Aplikace bude přijímat z příkazové řádky dva parametry: Oba představují jména vstupních souborů s popisem tvaru porovnávaných stromů. Program se bude spouštět příkazem

`ted.exe`[2] ⟨`tree1`⟩ ⟨`tree2`⟩↵

Symboly ⟨`tree1`⟩ a ⟨`tree2`⟩ zastupují jména vstupních souborů s popisem stromů. Žádné implicitní předpoklady o příponách či umístění souborů neexistují – jak umístění, tak jméno a přípona souboru mohou být zcela libovolné. Váš program tedy může být během testování spuštěn například takto:

```
...\>ted.exe tree01.dat tree02.dat↵
```

Výsledkem činnosti programu bude výpis vzdálenosti mezi stromy specifikovanými předanými vstupními soubory. Vzdálenost mezi stromy je minimální cena sad operací (mazání uzlu, editace, vložení nového uzlu) nutných k převedení jednoho stromu na tvar druhého. Každá operace nad stromem má jasně definovanou svoji cenu. Výsledná vzdálenost musí respektovat zadání a nastavení cen pro každou operaci se stromy (viz Vzdálenost stromů).

Pokud nebudou na příkazové řádce uvedeny dva argumenty, vypište chybové hlášení a stručný návod k použití programu v angličtině podle běžných zvyklostí (viz např. ukázková semestrální práce na webu předmětu Programování v jazyce C).

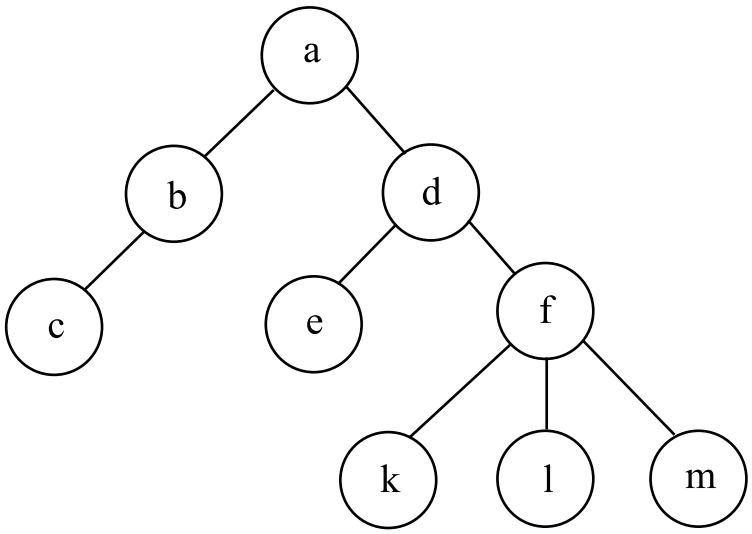
Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `makefile` a pro Windows `makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

Specifikace vstupu programu

Vstupem programu jsou pouze dva parametry – názvy souborů na příkazové řádce – interakce s uživatelem pomocí klávesnice či myši v průběhu práce programu se neočekává. Soubory s popisem tvaru stromu jsou – bez ohledu na případnou příponu – prosté textové soubory v kódování ASCII.

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10) a s běžnými distribucemi Linuxu (např. Ubuntu, Mint, OpenSUSE, Debian, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 10 Buster s jádrem verze 4.19.0-11-amd64 a s překladačem gcc 8.3.0.

[2] Přípona `.exe` je povinná i při sestavení v Linuxu, zejm. při automatické kontrole validačním systémem.



Obrázek 1: Grafické znázornění stromu.

V informatice je strom široce využívanou strukturou – je reprezentován uzly, přičemž každý uzel má právě jednoho rodiče (pokud není tzv. kořenem) a může mít $\langle 0, n \rangle$ potomků (následovníků). Uzel bez potomků se nazývá listem stromu.

Vstupní data obou stromů jsou uložena v textovém souboru. Každý řádek představuje uzel stromu s výčtem jeho potomků. Je zapsán ve formátu: "⟨uzel⟩`:`⟨potomek1⟩`,`⟨potomek2⟩`,...,`CR LF". První údaj tedy představuje pojmenování uzlu (může být prakticky libovolné), pak následuje dvojtečka a za ní výčet názvů potomků daného uzlu. Potomci ve výčtu jsou odděleni čárkami. Každý řádek je pak ukončen znaky konce řádku CR LF (tedy „windowsovským" způsobem). Vstupní soubor předpokládejte v kódování prostým 7-bitovým ASCII, tzn. bez znaků národních abeced. Pokud by se v pojmenování uzlů vyskytovaly akcentované znaky s ASCII hodnotou vyšší než 127, nemusíte jejich správnou interpretaci nijak ošetřovat (prostě načtěte příslušný byte ze souboru a netrapte se tím, zda má reprezentovat písmeno '`š`' nebo '`ą`'). Obrázek 1 zachycuje podobu stromu reprezentovaného následujícím popisem:

```
a:b,d CR LF
b:c CR LF
d:e,f CR LF
f:k,l,m CR LF
```

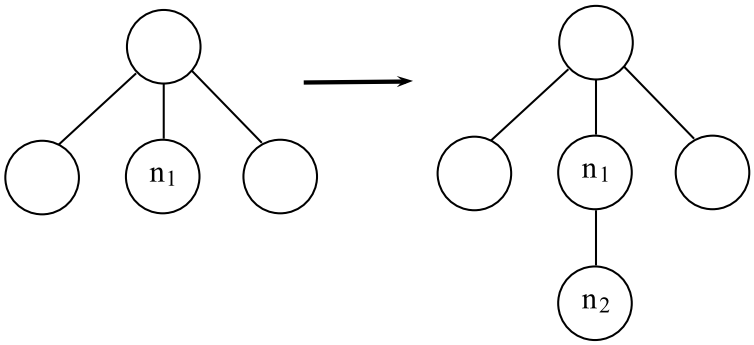
První prvek z prvního řádku textového souboru bude vždy kořenem stromu. Vstupní soubor nemá pevnou velikost a během testování může mít naprosto různý počet uzlů, pro reprezentaci stromů proto použijte **vhodně navrženou dynamickou strukturu**.

Vzdálenost stromů

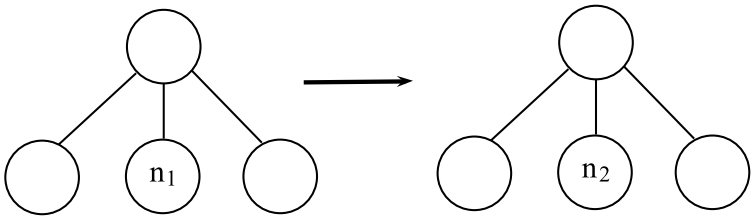
V kódu semestrální práce je tedy třeba implementovat výpočet již zmíněné vzdálenosti mezi dvěma stromy (T_1 a T_2). Vzdálenost vyjadřuje minimální množinu operací, které se musí vykonat k převodu stromů T_1 a T_2 do jednoho stejného stromu (viz obr. 5). Porovnávání struktur stromů potřebujeme v řadě disciplín: při porovnávání textů (např. vzdálenost dvou řetězců – více v předmětu Programovací techniky), v biologii, analýze obrazu, automatické analýze správnosti teorému, při optimalizaci kompilovaného kódu, ...

Množina operací (úkonů), které lze aplikovat na strom, vypadá takto:

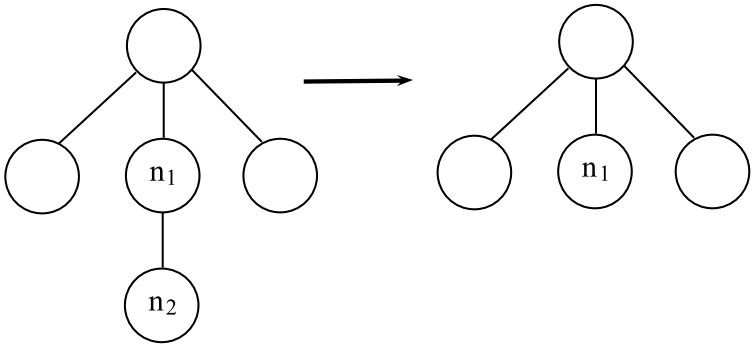
- Operace ***relabel*** (budeme dále značit písmenem '**r**') znamená změnu pojmenování uzlu ve stromu (viz obr. 3).
- Operace ***delete*** (dále značíme '**d**') značí vymazání uzlu, jenž není kořenem. Vymazání uzlu v, jenž má rodiče v' ve stromu T znamená, že následníci vymazaného uzlu v se stanou následníky rodiče tohoto uzlu (tedy uzlu v'). Potomci (následníci) jsou vloženi na místo za posledního následníka (má-li nějakého) uzlu v' v pořadí zleva doprava (viz obr. 4).
- Operace ***insert*** (označujeme '**i**') značí komplement k operaci **d**. Vložení uzlu v jako následníka uzlu v' ve stromě T znamená vytvoření nového rodiče v, jenž převezme následníky (nebyl-li uzel v' listem) uzlu v' ve stromě T (viz obr. 2).



Obrázek 2: Ukázka operace *insert*.



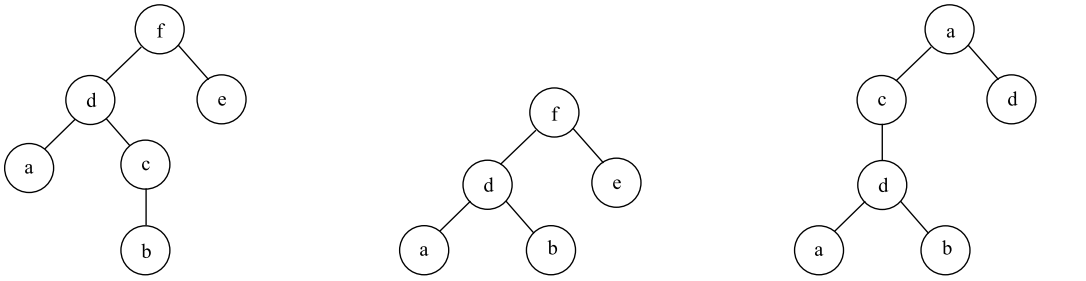
Obrázek 3: Ukázka operace *relabel*.



Obrázek 4: Ukázka operace *delete*.

Mějme definovanou cenovou funkci $\text{cost}(\cdot) \in \mathbb{N}$ nad každou operací: Operace *insert* bude mít cenu stanovenou jako $\text{cost}(\mathbf{i}) = 2$, operace *relabel* potom $\text{cost}(\mathbf{r}) = 1$ a *delete* má cenovou funkci

$\text{cost}(\mathbf{d}) = 2$. Vzdálenost stromů S mezi stromem T_1 a T_2 je sekvence operací měnící strom T_1 v T_2. Vzdálenost S neboli cena za převod T_1 v T_2 je vypočtena jako suma cen operací nutných k převodu jednoho daného stromu na druhý. Optimální vzdálenost $\delta(T_1, T_2)$ je posloupností takových operací, které převedou strom T_1 na strom T_2 s minimální cenou.



(a) Strom T_1 (b) Odstranění uzlu c (c) Vložení nového uzlu a přejmenování uzlů f, e

Obrázek 5: Ukázka transformace stromu T_1 na strom T_2.

Váš program by měl umožňovat **výpočet podobnosti**, tj. vyřešení a ohodnocení převodu stromů libovolné velikosti. Naprogramujte řešení tak, aby byla velikost stromu shora omezena jen dostupnou pamětí a výkonem použité výpočetní techniky.

Pokud reprezentace stromu nebude platná (tj. nedodržení definice stromu, cyklus, apod.), program bude reagovat chybovým hlášením (viz Specifikace výstupu programu) a ukončením činnosti.

Řešitelnost úlohy

Tato úloha je za předpokladu správně zadaných dat a splnění definice stromu **vždy řešitelná**.

Specifikace výstupu programu

Výstup programu bude směrován pouze na konzoli. Výstupem (v případě, že nenastala chyba) bude řádek s výpisem vzdálenosti mezi stromy (pokud řešení proběhlo bezchybně) následovaný znaky konce řádky '`\n`', na další řádce bude následovat výpis množina operací nad daným stromem, které je třeba vykonat k převedení stromu T_1 (daného prvním vstupním souborem) z jeho původní reprezentace do podoby shodné se stromem T_2 (daným druhým vstupním souborem). Každá operace bude oddělena čárkou (například `d,i,r,r`), po dokončení výpisu množiny operací bude řádek ukončen znakem '`\n`'. Posloupnost operací může být libovolná, kontrolován bude pouze počet jednotlivých použitých operací a ověření optimální vzdálenosti mezi stromy (jako suma cen daných operací). Pokud existuje více rozdílných množin operací se stejnou optimální vzdáleností, vypište i tyto množiny. Výpis každé další takové množiny bude opět ukončen znakem konec řádky '`\n`'.

Chybové stavy

Odhalí-li program chybu (zejména při analýze vstupních souborů), nechť reaguje výpisem jedné z chybových hlášek uvedených v tabulce. Mějte, prosím, na paměti, že z důvodu automatické kontroly odevzdávaného díla je nezbytně nutné **přesně dodržet** formát chybových hlášek.

Chybové hlášení	Význam, popis chybového stavu
`ERR#1: Missing argument!`	Na příkazové řádce nebyl(y) programu předán(y) parametr(y) specifikující vstupní soubor(y).
`ERR#2: Malformed input!`	Zápis reprezentace stromu v textovém souboru není korektní.
`ERR#3: Out of memory!`	Není k dispozici dostatek operační paměti.
`ERR#4: Cannot continue!`	Nespecifická chyba během výpočtu.

Hodnoty chyb od 5 výše můžete využít pro vlastní potřeby, ovšem dodržte uvedené formátování, tj. velkými písmeny zkratka `ERR`, následovaná bez mezer znakem '`#`' a číslem chyby – za číslem budiž pak dvojtečka a mezera a pak stručný popis chyby ukončený vykřičníkem '`!`'.

Číselný kód chyby nechť program předá také operačnímu systému prostřednictvím návratové hodnoty z funkce `int main(·)`.

Ukázka spuštění a výstupu

```
...\>ted.exe data_strom1.txt data_strom2.txt↵
6
d,i,r,r
```

Užitečné techniky a odkazy

Uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy víceméně standardní, lze k nim nalézt velké množství dokumentace:

1. Tree Edit Distance (`https://grfia.dlsi.ua.es/ml/algorithms/references/editsurvey_bille.pdf`),
2. n-ární stromy,
3. dynamické programování.

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli. Prostudujte materiály na `https://www.cs.haifa.ac.il/~oren/Publications/TEDinTALG.pdf`, `https://www.researchgate.net/publication/221313911_Analysis_of_Tree_Edit_Distance_Algorithms`, příp. další podobné, Google jich najde spousty.