Programovací Techniky

Samostatná práce

2016/2017

Vypracoval Dubina, Kubů

Zadání samostatné práce

Zadání samostatné práce :

Vytvořte program, který načte text ze souboru a ze všech unikátních slov vyskytujících se v textu vytvoří slovník. Slovník bude možné vyexportovat ve vhodně zvoleném formátu do souboru a znovu použít při dalším spuštění programu. Program dále umožní napsat nebo načíst libovolný text a v něm vyhledat zadané slovo. Pokud se slovo bude vyskytovat v prohledávaném textu, tak program vypíše počet výskytů a uvede u všech výskytů počáteční a koncový index, kde se v textu slovo nachází. Pro prohledávání textu využijte algoritmu komprimované trie. Pokud se zadané slovo v textu nenachází, tak vypište maximálně 10 nejbližších slov ze slovníku. K porovnání vzdáleností mezi hledaným slovem a slovy ze slovníku použijte Levensteinovu metriku probíranou na přednáškách. Uživatel by měl mít také možnost přidat hledané slovo do slovníku, pokud se v něm nenachází. Uživatelské rozhraní programu může být grafické i konzolové. Program bude mít ošetřeny všechny vstupy a zdrojový kód projde validací nástrojem PMD. Kód programu bude okomentovaný javadoc komentáři. Struktura dokumentace je uvedena níže v tomto dokumentu.

Odkaz na zadání samostatné práce:  
<http://home.zcu.cz/~prokop/SP_zadani_2016.pdf>

Analýza problematiky

Úkolem naší semestrální práce bylo vytvořit program, který ze zadaného textu vytvoří slovník unikátních slov, ve kterém bude možné slova vyhledávat.  
Jestliže se hledané slovo v textu nachází, tak uživatel dostane zprávu o všech jeho výskytech v textu včetně indexu. Pokud se hledané slovo nachází pouze ve slovníku, tak uživatel dostane zprávu o tom, že slovo bylo nalezeno ve slovníku v opačném případě se uživateli zobrazí hláška o nenalezení slova, vypsání deseti nejbližších slov a možnost vyhledávané slovo uložit do slovníku.  
Pokud byl slovník již vytvořen, tak se po zapnutí programu automaticky načte a uživatel jej nemusí opětovně vytvářet.

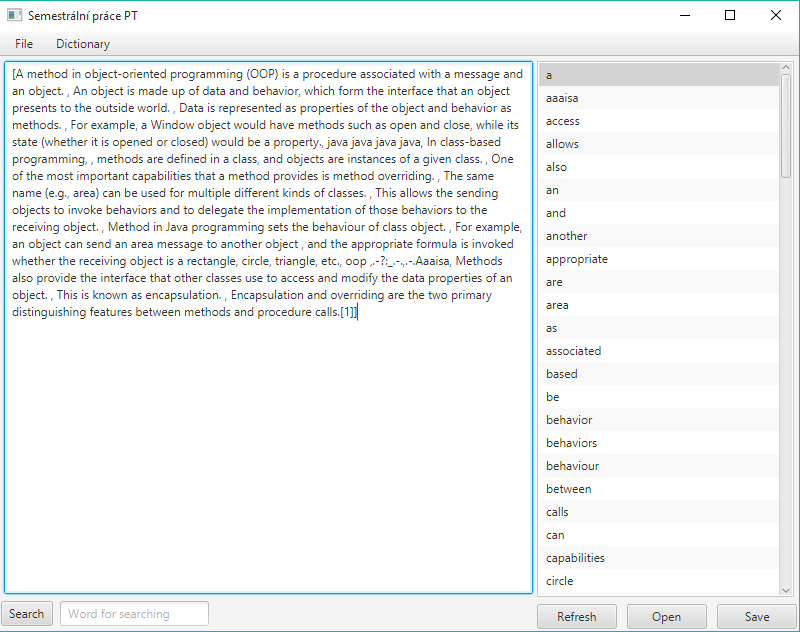
Trie

Jako datová struktura pro uložení jednotlivých slov ze slovníků nám byla přidělena stromová struktura Trie, která dále byla komprimována z důvodu menší časové náročnosti na prohledávání samotné trie.   
Hlavním rozdílem trie od BVS je to, že v BVS se podle hodnoty uzlu rozhodujeme na jakou větev popř. další uzel sestoupíme. Narozdíl o toho v trii v každý uzel, může obsahovat všechny podřetězce , kterými může řetězec pokračovat.   
Například pro trii nad anglickou abecedou písmen každý uzel, může mít až 26 potomků, protože přesně tolik má anglická abeceda písmen.   
Kořen trie je asociavaný s prázdným řetězcem a ani při komprimaci se nepřepisuje.   
Obvyklé použití trie je pro strukturování textových řetězců.

Komprimovaná trie   
 Komprimace trie se používá pro zrychlení vyhledávání v trii.   
Komprimace se provádí tím způsobem, že spojujeme uzly do sebe, ale pouze za určitých podmínek to můžeme provést. Jednou z podmínek je to, že uzel musí mít pouze jednoho potomka, neboli se daný uzel dále nevětvý na více uzlů.

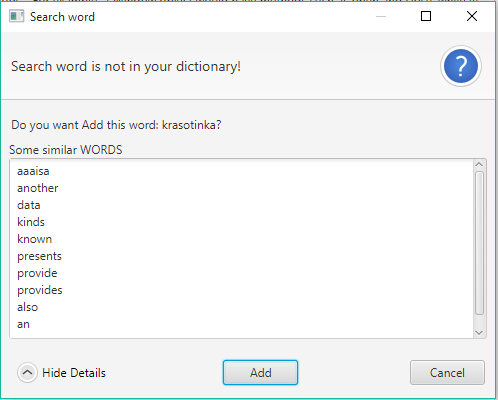
Levenshteinova vzdálenost  
 Je vzdálenost dvou řetězců definovaná jako minimální počet operací vkládání, mazání a substituce takových, aby po jejich provedení byly zadané řetězce totožné.  
Například Levenshteinova vzdálenost mezi řetězcem "KAMIL" a řetezcem "OMYL" je 3. (Změnilo se písmeno A a I a písmeno K bylo smazáno) Levenshteinova vzdálenost tedy vyjadřuje podobnost (resp. rozdílnost) dvou řetězců. Občas se nazývá "změnová vzdálenost" nebo také "editační vzdálenost" (edit distance)

Uživatelská dokumentace



Obrázek 1

Na obrázku 1 vidíme pohled na celé uživatelské rozhraní.   
V levé části vidíme textové pole, do kterého je možné nahrát textový soubor a nebo napsat libovolný text.   
Pod textovým polem je tlačítko Search ( hledej ), které vyhledá slovo zadané ve vedlejším textovém rámečku.   
V pravé části obrazovky je seznam slov, které tvoří slovník. Tyto slova jsou unikátní a seřazená podle abecedy. Pod seznamem slov se nachází 3 tlačítka pro manipulaci se slovníkem a to :   
Refresh – toto tlačítko aktualizuje informace, které se nachází ve slovníku.   
Open – pomocé tohoto tlačítka se nám otevře dialogové okno, ve kterém nás to vyzve, aby jsme vybrali textový soubor, ve kterém máme uložený slovnik.   
Save – tlačítko save slouží pro uložení slovníku do souboru, který se defaultně jmenuje Dictionary.txt a je uložen do složdy s programem.



Obrázek 2

Na obrázku číslo 2 vidíme vyskakovací okno, které po vyhledání neexistujícího slova, informuje uživatele o tom, že slovo ve slovníku neexistuje a vypíše až 10 nejbližších slov podle Levenštejnovi metody, k našemu hledanému slovu.   
V tomto okně máme na výběr ze dvou tlačítek:   
Add – zapříčiní přidání slova do slovníku .  
Cancel – ukončí dialogové okno a námi hledané slovo se do slovníku nezapíše.

Závěr

Celkové zhodnocení semestrální práce.

Semestrální práce z mého pohledu byla zajisté zajímavá a během programování jsme narazili na pár nehezkých problémů, nad kterýmy jsme se museli nekolikrát zamyslel a opakovaně testovat, debugovat a přepisovat daný úsek problémového kódu.   
Semestrální práce byla úkolem dvojic, a tak jsme na semestrální práci pracovali vzdáleně a pro koordinaci jsme používali internetový repozitář GitHub a voice chat, ale také jsme na ní pracovali společně ve škole.   
Veliké plus vidím to, že jsem si konečně ujasnil fungování a programování stromů a obohacení svých znalostí o datovou strukturu Komprimované trie a Levenštejnovi metriky.   
Nad skutečným přínosem aplikace běžnému uživateli, by se dalo diskutovat, ale k procvičení výše zmiňovaných struktur tato semestrální práce posloužila velmi dobře.