Trie

Tema 1

Responsabili: Rareș Tăerel și Ștefan Rușeți

1. Introducere

Scopul acestei teme este de a vă familiariza cu programarea orientată pe obiecte și cu limbajul Java. În același timp, aveți ocazia de a cunoaște o structură de date folosită în practică.

Un Trie [1] este o structură de arbore optimizată pentru stocarea de șiruri de caractere. Această structură este foarte utilizată în special pentru implementarea unei funcționalități de autocomplete similară cu cea a motorului de căutare Google sau cu autopredicția de pe telefonul mobil.

2. Structura unui Trie

La baza unui trie, este o structură de arbore optimizată pentru stocarea șirurilor de caractere. În cazul trie-ului, cheile nu sunt identificate prin informația dintr-un singur nod, ci prin drumul de la rădăcină până la un anumit nod. Astfel, fiecare nod are un număr de fii egal cu dimensiunea alfabetului folosit, iar fiecare muchie către un fiu este etichetată cu litera corespunzătoare din alfabet. Toți descendenții unui nod au un prefix comun asociat cu nodul respectiv iar rădăcina va fi totdeauna asociată cu un șir de caractere vid.

În figura de mai jos (Figura 1) este ilustrată starea trie-ului din exemplul de rulare.

Pentru simplitate grafică, am eliminat din desen muchiile irelevante și am reprezentat ca informație într-un nod doar numărul de cuvinte care corespund cheii nodului respectiv.

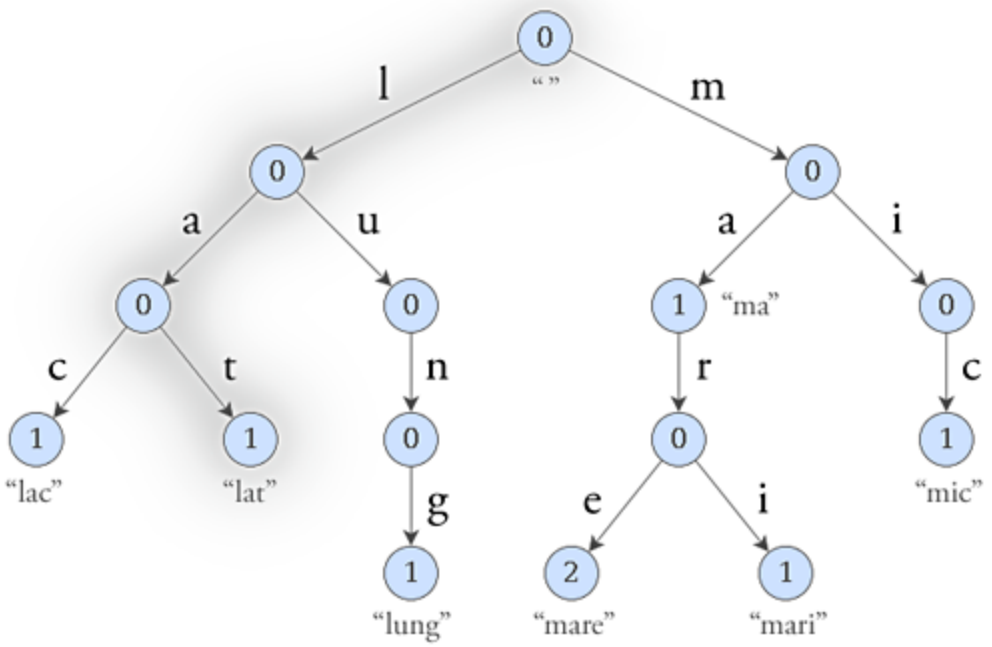

Figura 1

Pornind de la rădăcină (căreia i se asociază şirul vid), fiecărui nod i se asociază (va memora informații suplimentare) cheia al cărei nume se obține prin concatenarea etichetelor tuturor muchiilor de pe drumul de la rădăcina trie-ului până în nodul respectiv (în desen este evidențiat modul în care putem identifica nodul a cărui cheie este "lat").

Acest mod de gestionare a listei de cuvinte permite executarea fiecărei operații în timp O(L), unde L este lungimea unui şir de caractere. Spațiul de memorie folosit depinde de structura cuvintelor din listă, mai exact de prefixele lor comune. Totuși, el nu va depăși O(LungTot * |∑|), unde Sigma este alfabetul folosit (in cazul de față |∑|=26).

3. Cerințe specific

Vor exista două implementări ale trie-ului. Este obligatoriu implementarea interfeței AbstractTrie.java din scheletul temei, deoarece ne dorim o implementare cât mai generală. De asemenea, fiecare element din trie-uri va implementa interfața TrieElement.java ce conține o metodă toCharArray(). La adăugarea unui cuvânt într-un trie, nu se va folosi cuvântul efectiv ci rezultatul metodei toCharArray(). Exclusiv la nivelul acestei metode se va face diferențierea între cele două implementări ale trie-urilor cerute în cadrul temei.

Pentru primul trie va trebui să implementați toCharArray() astfel încât să ignore diferența între litere mari şi mici.

Pentru cel de-al doilea trie va trebui să implementați toCharArray() astfel încât să ignore caracterele '-', '_', '(', ')'.

În afară de cele două interfețe, în pachetul test aveți câteva clase pe care le puteți folosi pentru citirea comenzilor din fișier și scrierea rezultatelor.

Pe lângă implementarea efectivă, va trebui să generați și un javadoc pentru clasele create, folosind comentarii în cod [2] și generarea automată oferită de Netbeans / Eclipse, precum și un readme în care să explicați deciziile luate în implementarea temei și problemele întâmpinate.

Tema va fi testată pe **VMChecker** (va apărea în curând), astfel încât va trebui să aveți și un makefile cu o regula run care să ruleze clasa și care conține metoda ”main”, pentru a ușura testarea automată.

4. Testare

Fișierul de intrare se va numi *trie.in* iar cel de ieșire *trie.out.*

trie.in	***trie.out***
Linie cu cuvinte despărțite prin spațiu n op w_{11} op w_{12} op w_{13} op w_{1n} m op w_{21} op w_{22} op w_{23} op w_{2m}	rezultat_op_ w_{11} rezultat_op_ w_{12} rezultat_op_ w_{13} rezultat_op_ w_{1n} rezultat_op_ w_{21} rezultat_op_ w_{22} rezultat_op_ w_{23} rezultat_op_ w_{2m}

n – numărul de teste pentru primul trie

m – numărul de teste pentru cel de-al doilea trie

w – șir de caractere

op w – operația w ce primește ca parametru șirul de caractere w

Codificarea operațiilor este următoarea:

0 w – adaugă o apariție a cuvântului w în listă

1 w – șterge o apariție a cuvântului w din listă

2 w – tipărește numărul de apariții ale cuvântului w în listă

3 w – tipărește toate cuvintele unice care încep cu prefixul w, în ordine lexicografică (în cazul în care un cuvânt apare de mai multe ori, se afișează cel mai mic din punct de vedere lexicografic), despărțite prin spațiu

Exemplu:

trie.in	***trie.out***
lac lAt Lung l_a_c l-a-t lun)g 12 0 MA 0 mAre 0 MarI 0 mic 1 lat 0 lat 0 MaRe 2 mare 3 la 2 lac 2 latitudine 3 ma 12 0 m(a 0 ma(r)e 0 ma_ri 0 (mic) 1 lat 0 lat 0 mare 2 mare 3 la 2 lac 2 latitudine 3 ma	2 lac lat 1 0 MA MaRe MarI 2 l_a_c lat 2 0 m(a ma(r)e ma_ri

Explicații:

- Pentru primul set de de 12 teste se folosește funcția toCharArray() ce ignoră diferența dintre literele mari și mici iar pentru al doilea, funcția ce ignoră caracterele: ‘-‘, ‘_’, ‘(‘, ‘)’.

- Pentru prima comanda ”3 ma” există în trie 2 intrări echivalente, ”MaRe” și ”mAre”, însă se afișează ”MaRe” pentru ca este mai mic din punct de vedere lexicografic
- O situație mai special se întâmplă în cazul primei comenzi ”3 la”: deși ”lAt” este mai mic decât ”lat”, al doilea este cel afișat, deoarece prima variantă fusese eliminată din trie odată cu comanda ”1 lat”. În general, în cazul unui remove, se decrementează doar numărul de apariții ale elementului, neavând de unde ști exact care dintre elementele echivalente din trie trebuie eliminate. Elementul este eliminat de tot abia când contorul ajunge la zero.

5. Constrângeri

- Pentru toate operațiile, cuvântul w este format din caractere alfanumerice plus caracterele ‘-‘, ‘_’, ‘!’, ‘?’, ‘(‘, ‘)’
- Pentru realizarea temei trebuie folosită o versiune de Java nu mai recentă de Java 7, aceasta fiind versiunea care rulează pe VMChecker.
- Limita de timp pentru fiecare test în parte va fi publicată mai târziu, odată cu postarea temei pe VMChecker

6. Punctaj

Punctajul pe tema va consta din:

- 70% teste
- 10% coding style
- 20% readme + JavaDoc

7. Resurse

[1] Wikipedia Trie – http://en.wikipedia.org/wiki/Trie

[2] Definirea comentariilor în cod –

http://www.oracle.com/technetwork/articles/java/index-137868.html