Gevorderde Algoritmen en Datastructuren

Prakticumopdracht 1: Snel Zoeken

November 2010

Doel van deze opdracht is: ervaring opdoen met het analyseren en vergelijken van zoekalgoritmen.

De opdracht sluit aan bij Chapter 3 van het leerboek *Algorithm Design* van Goodrich en Tamassia. Het is de bedoeling om twee programma's te ontwikkelen, waarmee je kunt bepalen welke datastructuur beter is om een doorzoekbaar geordend lexicon (*searchable ordered dictionary*) te implementeren: een conventionele binaire zoekboom, of een van de volgende alternatieve datastructuren: AVL-boom, (2,4)-boom, *splay tree*, *skip list*. De programma's dienen een tekstbestand woord voor woord te lezen, waarbij elk *nieuw* woord wordt toegevoegd aan de datastructuur. Elk woord komt dus slechts éénmaal voor in de datastructuur.

Onderdelen Implementatie

1. Je kunt het onderstaande stuk Java-code gebruiken om een bestand te openen en daar woord voor woord uit te lezen. Hierin is `s` een `String` met een bestandsnaam:

```
try{
    Reader r = new BufferedReader (new FileReader(s));
    StreamTokenizer st = new StreamTokenizer(r);
    st.lowerCaseMode(true);
    st.whitespaceChars(',','.');
    st.nextToken();
    while (st.ttype != StreamTokenizer.TT_EOF) {
        if (st.ttype==StreamTokenizer.TT_WORD) {
           // het gelezen token staat in st.sval, doe er iets mee
        }
        st.nextToken();
    }
} catch(Exception e) {
   e.printStackTrace();
}
```

2. Schrijf een programma (in Java) dat een doorzoekbaar geordend lexicon implementeert met een binaire zoekboom. Voeg de woorden één voor één toe en zorg dat elk woord slechts één maal in de boom voorkomt. Het verwijderen van woorden uit de zoekboom hoeft dus niet geïmplementeerd te worden.

3. Kies een van de volgende vier alternatieve datastructuren: AVL-boom, (2,4)-boom, *splay tree*, *skip list*. Schrijf een tweede programma dat hetzelfde doet als het vorige, maar met de gekozen alternatieve datastructuur ipv. de binaire zoekboom.

Performance-analyse

Analyseer de beide programma's op twee manieren:

1. tel het aantal stringvergelijkingen dat nodig is om een bestand te verwerken,
2. tel het aantal toekenningen (assignments) in de stukken code die worden uitgevoerd.

Vergeet hierbij de vergelijkingen van en de toekenningen aan tijdelijke variabelen niet. Neem hierbij aan dat een stringvergelijking en een toekenning constante tijd kosten.
Voer de analyse uit op de gegeven tekstbestanden text1, text2, text3, text4 en text5 (op de Nestorpagina Prakticum).

Rapportage

Schrijf een verslag met een korte uitleg van de implementatie en voeg de code toe in een appendix. Beschrijf ook hoe je code gebruikt dient te worden. Neem de resultaten van de analyse op en geef daarbij een korte discussie.

Deadline

De deadline voor het inleveren van het verslag en de code is **vrijdag 17 december 2010, 17.00 uur**. Voor elke werkdag (of gedeelte daarvan) dat het verslag te laat ingeleverd is, gaat er een punt van het cijfer af.

Extra uitdagingen

De opdracht kan oa. op de volgende wijzen verzwaard worden:

- kies *twee* of meer alternatieve datastructuren en vergelijk die met elkaar;
- implementeer ook het *verwijderen* van woorden en verwerk dat in de performance-analyse.

Deze en andere verzwaringen kunnen het cijfer verhogen. Het maximum blijft overigens 10.