Datastructuren: Opdracht 2
Binair Zoeken en Binaire Zoekbomen

Deadline:
13 Oktober 2013 om 23:59:59

De onderwerpen van deze opdracht zijn het binaire zoekalgoritme[1] en de binaire zoekboom[2]. Het is de bedoeling twee datastructuren te implementeren: een array dat ten alle tijde gesorteerd is en een binaire zoekboom.

1 Het Binaire Zoekalgoritme

Het eerste deel van deze opdracht gaat over het binaire zoekalgoritme. Het is de bedoeling een array te implementeren die ten alle tijden gesorteerd is van klein naar groot. Elk toegevoegd element hoort maar één enkele keer voor te komen in de datastructuur. De datastructuur heeft voor elk element in de array een frequentieteller die bijhoudt hoe vaak een element is "toegevoegd". De datastructuur is dus een array met de volgende aangepaste operaties:

- Een toevoegprocedure die een element toevoegt aan het gesorteerde array of de bijbehorende frequentieteller ophoogt. Hierbij moet de juiste plaats van toevoegen of ophogen in het array gevonden worden door middel van *binair zoeken*.
- Een verwijderprocedure die van een element het aantal in het array verlaagt of deze bij een resulterend aantal van 0 uit het array verwijdert. Ook hierbij moet het zoeken naar de positie gedaan worden door middel van *binair zoeken*.

De array mag statisch zijn en een maximum grootte hebben (bijv. 30). Het is ook toegestaan een dynamisch array met variabele grootte te maken.

2 Binaire Zoekbomen

Het tweede deel van deze opdracht gaat over het implementeren van een binaire zoekboom. De binaire zoekboom heeft *tenminste* de volgende operaties:

- `insert()` Voor het toevoegen van elementen aan de binaire zoekboom. Een element komt maar één keer voor in de boom.
- `search()` Voor het zoeken van elementen in de binaire zoekboom. Deze funtie hoort `true` terug te geven waneer het element in de boom zit, `false` zo niet.
- `toDot()` Om de binaire zoekboom om te zetten in DOT-notatie.

Het DOT-bestand kan worden gebruikt om de boom makkelijk te visualiseren in een programma als GraphViz, welke geïnstalleerd staat op de Linux werkomgeving, bijvoorbeeld met het commando `dotty <filename>`. Er zijn ook Windows uitvoeringen beschikbaar. Voor informatie over de structuur van een DOT-bestand, zie de officiële handleiding [1] of Wikipedia [2].

[1] Hoorcollege van 18 September 2013
[2] Hoorcollege van 25 September 2013

3 Opgaven

Binair Zoeken De correcte werking van de gesorteerde array moet worden aangetoond met de volgende procedures:

- Genereer 50 willekeurige integers in het bereik $[0, 40]$, druk deze integers af op het scherm (waarden en frequenties), voeg deze integers toe aan het array en druk de array af op het scherm.
- Genereer 10 nieuwe willekeurige integers in het bereik $[0, 40]$, druk deze integers af op het scherm (waarden en frequenties), probeer deze 10 integers te verwijderen uit het array en druk het array af op het scherm.

Binaire Zoekbomen De correcte werking van de binaire zoekboom moet worden aangetoond met de volgende procedures:

- Genereer 50 willekeurige integers in het bereik $[0, 50]$ druk deze integers af op het scherm, voeg deze integers toe aan de binaire zoekboom en exporteer deze boom vervolgens naar een DOT-bestand.

4 Eisen

De opdracht wordt beoordeeld op werking, keuzes van de implementatie, netheid van programmeren en verdere navolging van de eisen die in deze opdracht worden gesteld. De implementatie moet gebruik maken van *templates* en `const`-*correctness*. De code moet foutloos te compileren zijn met g++[3]. Er mag geen gebruik worden gemaakt van de C++ STL datastructuren en algoritmes[4].

Zorg dat beide datastructuren op een nette manier zijn geïmplementeerd en kunnen worden hergebruikt!

Inleveren Er moet een *Makefile* en een tekstbestand meegeleverd worden. Het tekstbestand bevat de namen en studentnummers van de makers, instructies voor het compileren en voor het draaien van de verschillende opdrachten, een duidelijke uitleg over de invoer en uitvoer hiervan en eventueel ander commentaar. De broncode, *Makefile* en het tekstbestand moeten samen ingepakt worden in een archief, zoals zip/gzip/rar/tgz. Stuur geen *executables* mee! Stuur ook geen DOT-bestanden mee!

De uitwerking dient uiterlijk op zondag 13 Oktober 2013 om 23:59:59 via email opgestuurd te worden naar Jan van Rijn (j.n.van.rijn@liacs.leidenuniv.nl).

Referenties

[1] E. R. Gansner, E. Koutsofios, and S. North. Drawing Graphs with *dot*. `http://www.graphviz.org/pdf/dotguide.pdf`.

[2] Wikipedia. Dot (graph description language). `http://en.wikipedia.org/wiki/DOT_language`.

[3] De g++ compiler die geïnstaleerd staat op de LIACS Linux werkplekken wordt als referentie gebruikt. Er hoort gecompileerd te worden met tenminste de `-Wall -Wextra` flags.

[4] De standaard `#includes` zoals `<iostream>`,`<cstdlib>`,`<cstdio>`,`<ctime>` zijn wel toegestaan. Bij twijfel, vraag het aan de assistenten.