DM507 Algoritmer og datastrukturer

Forår 2010

Projekt, del II

Institut for matematik og datalogi
Syddansk Universitet

19. marts, 2010

Dette projekt udleveres i tre dele. Hver del har sin deadline, således at afleveringerne, og dermed arbejdet, strækkes over hele semesteret. Projektet skal besvares individuelt. Deadline for del II er tirsdag den 20. april.

Mål

Målet for del II af projektet er at implementere rød-sorte træer, samt bevise en øvre grænse for mængden af rebalancering. Med hensyn til updates skal der kun ses på indsættelser, ikke sletninger.

Rød-sorte træer

Rød-sorte træer er grundigt beskrevet i Cormen et al., kapitel 13. Bemærk at pseudo-koden i dette kapitel er baseret på en implementation med en sentinel-knude til at repræsentere tomme undertræer (samt rodens forælder), som illustreret i Figur 13.1 (b) på side 310. Denne knude er altid sort. Vi antager også denne implementation her.

Opgaver

Opgave 1

Der skal laves en Java-implementation, baseret på bogens pseudo-kode, af rød-sorte træer, som indeholder metoderne SEARCH (pseudo-kode side 290 eller 291), INSERT (pseudo-kode side 315 (og 316, 313)), samt INORDER-TRAVERSAL (pseudo-kode side 288).

Det antages at nøgler er af typen `int` (så man ikke behøver bruge f.eks. generics i Java), og at elementer blot består af nøgler (der er ikke yderligere

data associeret til en nøgle). Implementationen skal være i form af en Java-klasse, som kan bruges af andre programmer. Klassen skal hedde `RBT`, og skal implementere flg. interface:

```
public interface RBTree {
    public boolean search(int k);
    public void insert(int k);
    public int[] inorderTraversal();
    public boolean isRedBlack();
}
```

Metoden `inorderTraversal()` returnerer elementerne i et array (i sorteret orden) fremfor at printe på skærmen som i bogens pseudo-kode. Metoden `isRedBlack()` checker om et givet `RBTree` overholder (de vigtigste af) kravene 1–5 på side 308. Denne metode er bla. anvendelig til fejlfinding under implementationsprocessen. Metoden skal baseres på flg. pseudo-kode:

```
IsRedBlack(T)
    return {T.root.color == black and
                BHeight(T, T.root) ≥ 0 and
                (not TwoReds(T, T.root))}

BHeight(T, v)
    if v == T.nil
        return 0
    else
        h1 = BHeight(T, v.left)
        h2 = BHeight(T, v.right)
        if h1 ≠ h2 or h1 == −1
            return −1
        elseif v.color == black
            return h1 + 1
        else
            return h1


TwoReds(T, v)
    if v == T.nil
        return false
    elseif {v.color == red and
                (v.left.color == red or v.right.color == red)}
        return true
    else
        return TwoReds(T, v.left) or TwoReds(T, v.right)
```

Du skal i rapporten argumentere for køretiden af denne pseudo-kode, og for at IsRedBlack(T) returnerer **true** hvis og kun hvis T opfylder kravene 2, 4, og 5 på side 308 i Cormen et al. (det antages at krav 1 *er* opfyldt, dvs. at der kun bruges farverne sort og rød, og krav 3 bliver automatisk opfyldt via brugen af en sentinel-knude med farven sort).

Opgave 2

Vi ønsker i denne opgave at bevise, at hvis man starter med et tomt rød-sort træ og laver n indsættelser, da er den samlede mængde rebalanceringsarbejde (dvs. arbejde lavet af pseudo-koden side 316) $O(n)$. Denne viden vil vise sig brugbar i del III af projektet.

For et rød-sort træ T lader vi $\mathrm{R}(T)$ være antallet af sorte knuder som har to røde børn. For eksempel er $\mathrm{R}(T) = 1$ for træet i Figur 13.1 (side 310) i Cormen et al.. Vi skal se på hvordan $\mathrm{R}(T)$ udvikler sig, når T ændrer form under indsættelser og efterfølgende rebalanceringer.

En indsættelse består i at erstatte et tomt undertræ med en ny knude, og derefter udføre pseudo-koden på side 316. Som det fremgår af diskussionen i afsnit 13.3 af Cormen et al. vil while-løkken i denne pseudo-kode løbe nul eller flere gange gennem Case 1, og derefter højst een gang gennem Case 2 og højst een gang gennem Case 3, hvorefter den stopper.

1. Argumentér for at selve indsættelsen (at erstatte et tomt undertræ med en ny knude, uden rebalancering) højst kan øge $\mathrm{R}(T)$ med een.
2. Argumentér for at under rebalancering vil Case 2 ikke ændre $\mathrm{R}(T)$ (brug Figur 13.6 og den tilhørende figurtekst).
3. Argumentér for at under rebalancering kan Case 3 højst øge $\mathrm{R}(T)$ med een (brug Figur 13.6 og den tilhørende figurtekst).
4. Argumentér for at hvis en rebalancering starter med k gange Case 1, da vil disse sænke $\mathrm{R}(T)$ med mindst $k - 1$ (brug Figur 13.5 og den tilhørende figurtekst).

Vi ser nu på situationen hvor man starter med et tomt rød-sort træ T_{start} og laver n indsættelser, resulterende i et træ T_{slut}, og ønsker at vurderere den samlede mængde rebalanceringsarbejde.

4. Argumentér for at der højst n gange i alt udføres Case 3.
5. Lad k_i betegne det antal gange Case 1 udføres ved rebalancering efter den i'te indsættelse. Bemærk at $\mathrm{R}(T_{\text{start}}) = 0$, og at $0 \leq \mathrm{R}(T_{\text{slut}})$.

Argumentér for at

$$0 \leq \mathrm{R}(T_{\mathrm{slut}}) \leq \mathrm{R}(T_{\mathrm{start}}) + 2n - \sum_{i=1}^{n}(k_i - 1)$$

ved at bruge tidligere viste udsagn.

6. Argumentér for at det heraf følger at $\sum_{i=1}^{n} k_i \leq 3n$.
7. Argumentér for at den samlede mængde rebalanceringsarbejde under de n indsættelser er $O(n)$.

Formalia

Lav en rapport, som indeholder dine svar på opgave 1 og 2 ovenfor. Koden for opgave 1 skal være passende kommenteret, skal inkluderes i rapporten som bilag, og eventuelle ikke-trivielle aspekter af implementeringen skal diskuteres i rapportens hoveddel. Der skal afleveres rapporten i pdf-format, samt Java-implementationen som separate filer (dvs. udover deres inklusion på tryk i rapporten).

Materialet afleveres med `aflever` kommandoen i en shell på Imadas Linux-system: Lav et directory som indeholder ovenstående filer, flyt ned i dette, og udfør kommandoen `aflever DM507`. Dette vil kopiere indholdet af dette directory til et sted i systemet, som underviseren kan tilgå. En email vil blive sendt til ens studentermail som kvittering. Man kan bruge kommandoen flere gange, senere anvendelser overskriver materialet fra tidligere anvendelser.

Aflever materialet senest:

Tirsdag den 20. april, 2010, kl. 23:59.

PS: Bemærk at aflevering af andres kode, f.eks. hentet fra nettet, er eksamenssnyd, og vil blive behandlet som sådan. Man lærer desuden heller ikke noget.