DM507/DS814 Algoritmer og datastrukturer

Forår 2022

Projekt, del II

Java version

Institut for matematik og datalogi
Syddansk Universitet

31. marts, 2022

Dette projekt udleveres i tre dele. Hver del har sin deadline, således at arbejdet strækkes over hele semesteret. Deadline for del II er onsdag den 20. april kl. 23:59. De tre dele I/II/III er ikke lige store, men har omfang omtrent fordelt i forholdet 15/30/55. Projektet skal besvares i grupper af størrelse to eller tre.

Mål

Det overordnede mål for projektet i DM507 er træning i at overføre kursets viden om algoritmer og datastrukturer til programmering. Projektet og den skriftlige eksamen komplementerer hinanden, og projektet er ikke ment som en forberedelse til den skriftlige eksamen.

Det konkrete mål for del II af projektet er først at implementere datastrukturen *ordered dictionary* (ordnet ordbog), og derefter bruge den til at sortere tal. Arbejdet vil også virke som forberedelse til del III af projektet.

Opgaver

Opgave 1

Kort sagt er opgaven at overføre bogens pseudo-kode for ubalancerede søgetræer til en Java-klasse.

Krav i opgave 1

Du skal i Java lave en klasse `DictBinTree`, som tilbyder (`implements`) følgende interface (koden udleveres):

```
import java.util.ArrayList;

public interface Dict {
    public boolean search(int k);
    public void insert(int k);
    public ArrayList<Integer> orderedTraversal();
}
```

Nøgler er af typen `int`, og der er ikke yderligere data tilknyttet en nøgle (dvs. vi bruger *ikke* `Element` fra del I her). Metoden `search(k)` returnerer blot en boolean, der angiver om nøglen `k` er i træet. Metoden `insert(k)` indsætter nøglen `k` i træet. Metoden `orderedTraversal()` returnerer en kopi af træets nøgler i en `ArrayList` af `Integer` i sorteret orden (fremfor at printe dem på skærmen som i bogens pseudo-kode).

Implementationen skal laves ved hjælp af strukturen *binært søgetræ*, som beskrevet i Cormen et al. kapitel 12. Som det fremgår af interfacet `Dict` skal der kun implementeres indsættelse (pseudo-kode side 294), søgning (pseudo-kode side 290 eller 291), og inorder gennemløb (pseudo-kode side 288). Implementationen *skal* basere sig på denne pseudo-kode. Træet skal ikke holdes balanceret (der skal ikke bruges metoder fra kapitel 13).

Implementationen skal være i form af en Java-klasse ved navn `DictBinTree`. Klassen skal implementere interfacet `Dict`. Klassen skal have én constructor-metode ved navn `DictBinTree()`, som opretter en ny, tom dictionary.

Du skal definere en separat Java-klasse `BinNode` til at repræsentere knuder i træer (du skal *ikke* bruge en array-struktur til at repræsentere træet, sådan som for en heap i del I af projektet). Et objekt af typen `BinNode` skal indeholde referencer til to andre `BinNode`'s (dens venste barn og højre børn), med værdi `null` hvis pågældende barn ikke findes. Det skal også indeholde en nøgle, men behøver i dette projekt ikke at indeholde en reference til en forælder.

Et objekt af typen `DictBinTree` skal indeholder en reference til den `BinNode`, som er rod i træet (hvis træet er tomt, har denne reference værdi `null`). Det vil sige, at et `DictBinTree` objekt og dets reference til roden svarer til T og $T.root$ fra Cormen et al. (figur 10.9 side 247 samt kapitel 12).

For hvert træ er der således præcis ét objekt af typen `DictBinTree` og nul eller flere objekter af typen `BinNode`. Nye objekter af typen `BinNode` oprettes under `insert` (ved brug af `new` i Java, som for alle objekter).

Der vil være behov for at implementere metoder udover dem i interfacet, til internt brug i klassen. F.eks. vil der for rekursive funktioner skulle laves *to* udgaver: den offentlige funktion fra interfacet, samt en funktion som gør det virkelige arbejde (og svarer til pseudo-koden). Den første er ikke rekursiv, men kalder blot den anden og tilføjer i kaldet parametre med relevante værdier (f.eks. at knuden, som der kaldes på, er træets rod). For funktioner baseret på en løkke kan disse parameter blot oprettes inden løkken går i gang. I inorder gennemløb vil der være brug for at sende en `ArrayList` med som argument. Når pseudo-koden for inorder gennemløb vil udskrive en knudes element, skal man i stedet tilføje elementet til enden af `ArrayList`'en med `add()` (uden index argument). Bemærk at parametrene i interfacet ovenfor er anderledes end i bogens pseudo-kode. Dette skyldes, at implementationsdetaljer (såsom at der findes `BinNode`'er i dictionary'en) ikke skal være synlige for brugere af datastrukturen.

Husk at teste alle tre metoder fra `Dict` grundigt for din `DictBinTree` klasse (herunder test på tomme træer, test af indsættelse af ens nøgler, samt test af søgning efter både eksistende og ikke-eksisterende nøgler) inden du går videre til næste opgave.

Opgave 2

Du skal implementere en sorteringsalgoritme kaldet Treesort baseret på metoderne i interfacet `Dict`. Algoritmen består i at lave gentagne `insert`'s i en dictionary, efterfulgt af et kald til `orderedTraversal`. Tallene i den returnerede `ArrayList` skal så blot skrives ud.

Krav i opgave 2

Algoritmen skal implementeres i et program kaldet `Treesort.java`. Dette program skal bruge din ovenfor udviklede klasse `DictBinTree`.

Præcis som det udleverede program `PQSort` fra del I af projektet skal `Treesort` læse fra standard input (der som default er tastaturet), og skrive til standard output (der som default er skærmen). Dette skal (præcis som i programmet `PQSort`) gøres som følger: Input læses via klassen `Scanner` fra biblioteket `java.util` og bruger dens metode `nextInt()` til at indlæse tallene. Output laves via `System.out.println()`. Input til `Treesort` er en sekvens af `char`'s bestående af heltal adskilt af newlines, og programmet skriver som output tallene i sorteret orden, adskilt af newlines. Som eksempel skal `Treesort` kunne kaldes således i en kommandoprompt:

```
java Treesort
34
```

```
645
-45
1
34
0
Control-D
```

(Control-D angiver slut på data under Linux og Mac, under Windows brug Ctrl-Z og derefter Enter) og skal så give følgende output på skærmen:

```
-45
0
1
34
34
645
```

Ved hjælp af *redirection*[1] af standard input og output kan man i en kommandoprompt anvende *samme* program også på filer således:

```
java Treesort   < inputfile   > outputfile
```

Som test af `Treesort` kan man køre det på testfilerne fra del I.

En vigtig grund til, at du skal afprøve ovenstående metode (med redirection i en kommandoprompt), er at programmerne skal kunne testes automatisk efter aflevering. Man må af samme grund heller ikke i sin kildekode have `package` statements, eller organisere sin kode i en folderstruktur. Dette sker ofte automatisk hvis man bruger en IDE som Eclipse, NetBeans eller IntelliJ under udvikling af koden. I så fald må man fjerne `package` statements og folderstruktur inden aflevering (og derefter igen teste funktionaliteten, herunder redirection, i en kommandoprompt).

Formalia

Du skal kun aflevere dine filer med Java kildekode. Filerne skal indeholde en fornuftig mængde kommentarer om, hvad koden gør. De skal også indeholde navnene og SDU-logins på gruppens medlemmer.

Filerne skal enten afleveres som individuelle filer eller som ét `zip`-arkiv (med alle filer på topniveau, dvs. uden nogen directory struktur).

[1] Læs evt. om redirection på William Shotts' website (med flere detaljer her), Wikipedia eller Unix Power Tools.

Filen skal afleveres elektronisk i itslearning i mappen Projekt under faneblad Ressourcer. Afleveringsmodulet er også sat ind i en itslearning plan i kurset.

Under afleveringen skal man erklære gruppen ved at angive alle medlemmernes navne. Man skal kun aflevere én gang per gruppe. Bemærk at man under aflevering kan oprette midlertidige "drafts", men man kan kun aflevere *én gang*.

Aflever materialet senest:

Onsdag den 20. april kl. 23:59.

Bemærk at aflevering af andres kode eller tekst, hvad enten kopieret fra medstuderende, fra nettet, eller på andre måder, er eksamenssnyd, og vil blive behandlet særdeles alvorligt efter gældende SDU regler. Man lærer desuden heller ikke noget. Kort sagt: kun personer, hvis navne er nævnt i den afleverede fil, må have bidraget til koden.