Ugeseddel: Binære søgetræer

Carsten Witt*

5. maj 2022

Om denne uge

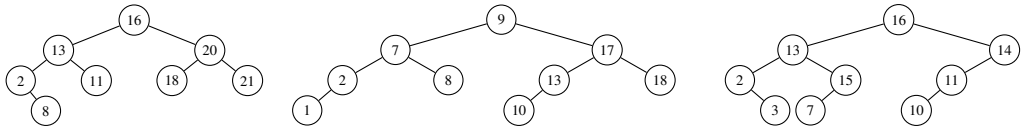
Materialer *Introduction to Algorithms*, Cormen, Leiserson, Rivest og Stein (CLRS): kap. 12 på nær 12.4. Se også ugeplan på hjemmesiden.

Opgaver

Opgaver markeret med † er tilgængelige i CodeJudge og opgaver markeret med * er særligt udfordrende.

1 Håndkøring og egenskaber

1.1 [w] Hvilke af følgende træer er et binært søgetræ?



1.2 [w] Hvor findes elementet med hhv. mindste og største nøgle i et binært søgetræ?

1.3 [w] Givet nøglemængden $\{1, 4, 5, 10, 16, 17, 21\}$, tegn binære søgetræer af højde $2, 3, 4, 5$ og 6 indeholdende disse nøgler.

1.4 [w] Angiv præorder-, inorder- og postordersekvensen af nøgler for det midterste træ i 1.1.

1.5 Sammenlign hoborden og søgetræsinvarianten.

1.6 Skriv pseudokode for en iterativ version af inordergennemløb.

1.7 Vis at når en knude v har 2 børn, så har dens successor ikke noget venstre barn og dens predecessor intet højre barn.
Antag at alle nøgler er unikke. *Hint:* lav et modstridsbevis.

2 Blade og højde

Lad T være et binært træ med n knuder og rod v.

2.1 Giv en rekursiv algoritme der givet v beregner antallet af blade i T. Skriv pseudokode for din løsning.

2.2 Giv en rekursiv algoritme der givet v beregner højden af T. Skriv pseudokode for din løsning.

2.3 [†] Implementér din algoritme til at beregne højden.

*baseret på Philip Billes materiale

3 Mere Rekursion på Træer (gammel eksamensopgave)

Denne opgave handler om rekursion og rodfæstede binære træer. Hver knude x har felterne $x.left$, $x.right$ og $x.parent$. Felterne peger hhv. på knudens venstre barn, højre barn og forælderen. For rodknuden r gælder $r.parent = null$. Hver knude x i træet har derudover et felt $x.size$, der indeholder et heltal. Betragt følgende algoritme:

```
ZERO(x)
if x ≠ null then
  x.size = 0
  ZERO(x.left)
  ZERO(x.right)
end if
```

1. Analyser køretiden af $\textsc{Zero}(x)$ som en funktion af n, hvor x er rodknuden i et træ med n knuder.
2. Lad $T(x)$ være deltræet med x som rod og lad $|T(x)|$ angive antallet af knuder i $T(x)$. Giv en algoritme, $\textsc{InitSize}(x)$, der givet en rodknude x af et træ sætter $y.size$ til $|T(y)|$ for alle knuder y i træet. Skriv din algoritme i pseudokode og analyser køretiden af din algoritme som en funktion af n, hvor n er antallet af knuder i træet.
3. Vi siger at kanten mellem en knude x og dens barn y, (x, y), er *rød*, hvis $|T(x)| \geq |T(y)|$. Giv en rekursiv algoritme, $\textsc{RedEdge}(x)$, der givet en rodknude udregner antallet af røde kanter i træet. Skriv din algoritme i pseudokode og analyser køretiden af din algoritme som en funktion af n, hvor n er antallet af knuder i træet.
4. Analyser og angiv i O-notation det maximale antal af røde kanter på enhver sti fra roden af træet til et blad.

4 Gennemløb af binære søgetræer

4.1 Giv en algoritme der givet et binært træ T med nøgler i hver knude afgør om T overholder søgetræsinvarianten.

4.2 Giv en algoritme der givet et binært søgetræ T konstruerer et omvendt binært søgetræ T^{R}. Træet T^{R} skal være et binært træ med de samme nøgler som T. For enhver knude v i T^{R} skal der gælde at nøglerne i venstre deltræ er $\geq v$ og nøglerne i højre deltræ er $\leq v$.

4.3 [$*$] Giv en algoritme der, givet to binære søgetræer T_1 og T_2, konstruerer et enkelt binært søgetræ over elementerne fra både T_1 og T_2.

5 Perfekt balancerede binære søgetræer

Lad $A[0..n-1]$ være et sorteret array af $n = 2^{h+1} - 1$ forskellige tal. Giv en sekvens af indsættelser af tallene i A i et binært søgetræ T således at T bliver et komplet binært træ af højde h.

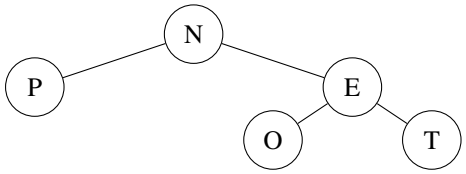
6 Preordergennemløb

[†] Implementér en rekursiv algoritme til at lave preordergennemløb af et binært træ.

7 Endnu mere rekursion på træer (gammel eksamensopgave)

Denne opgave handler om rekursion og rodfæstede binære træer. Hver knude x i træet har felter $x.left$, $x.right$, og $x.parent$. Såfremt rodknuden for et træ er *null*, er træet tomt. Felterne hhvs. $x.left$, $x.right$, $x.parent$ peger på knudens venstre og højre barn og forældren og er muligvis *null*, hvis knuden ikke eksisterer. I særdeleshed gælder r.parent = *null* for rodknuden r. Hver knude x i træet har derudover et felt $x.label$, der indeholder et bogstav. Betragt følgende algoritme:

```
PrintTree(x)
if x ≠ null then
   PRINT(x.label)
   if x.left ≠ null then
      PrintTree(x.left)
   end if
   if x.right ≠ null then
      PrintTree(x.right)
   end if
end if
```



Lad x være rodknuden for træet i ovenstående figur. Et kald til algoritmen PrintTree(x) vil da udskrive sekvensen `NPEOT`.

7.1 Du skal ændre algoritmen PrintTree således at det rekursive gennemløb ved kaldet til PrintTree(x) for rodknuden x i træet fra figur 1 udskriver sekvensen `NETOP` i stedet. Argumentér for at din algoritme er korrekt.

7.2 En intern knude i et træ er en knude, som ikke er et blad. Konstruer en algoritme Intern(x) der givet en rodknude x for et træ, returnerer antallet af interne knuder i træet. Skriv pseudokode for din algoritme. Angiv tidskompleksiteten/køretiden af din løsning i O-notation som funktion af n, hvor n er antallet af knuder i træet. Begrund dit svar.

7.3 Et træ har en R-rodvej hvis der er en vej i træet fra roden x til et blad hvor alle knuderne v på vejen fra x til bladet har v.label = `R`. Konstruér en rekursiv algoritme RRodvej(x) der returnerer tallet 1 hvis træet med roden x har en R-rodvej. Hvis en sådan vej ikke eksisterer, skal algoritmen returnere 0. Skriv pseudokode for din algoritme og argumentér for at den er korrekt. Angiv dernæst tidskompleksiteten/køretiden af din løsning i O-notation. Begrund dit svar.